

Olga De Troyer Claudia Bauzer Medeiros
Roland Billen Pierre Hallot
Alkis Simitsis Hans Van Mingroot (Eds.)

LNCS 6999

Advances in Conceptual Modeling

Recent Developments and New Directions

ER 2011 Workshops FP-UML, MoRE-BI,
Onto-CoM, SeCoGIS, Variability@ER, WISM
Brussels, Belgium, October/November 2011, 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

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

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

Olga De Troyer Claudia Bauzer Medeiros
Roland Billen Pierre Hallot
Alkis Simitsis Hans Van Mingroot (Eds.)

Advances in Conceptual Modeling

Recent Developments and New Directions

ER 2011 Workshops FP-UML, MoRE-BI,
Onto-CoM, SeCoGIS, Variability@ER, WISM
Brussels, Belgium, October 31 - November 3, 2011
Proceedings



Springer

Volume Editors

Olga De Troyer

Vrije Universiteit Brussel, Department of Computer Science, Brussel, Belgium

E-mail: olga.detroyer@vub.ac.be

Claudia Bauzer Medeiros

University of Campinas, Institute of Computing, Campinas, SP, Brazil

E-mail: cmbm@ic.unicamp.br

Roland Billen

Pierre Hallot

Université de Liège, Geomatics Unit, Liège, Belgium

E-mail: {rbillen; p.hallot}@ulg.ac.be

Alkis Simitsis

Hewlett-Packard Laboratories, Palo Alto, CA, USA

E-mail: alkis@hp.com

Hans Van Mingroot

Business Development and Academic Relations Belgium-Luxembourg

Brussel, Belgium

E-mail: vanmingroot@be.ibm.com

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-24573-2

e-ISBN 978-3-642-24574-9

DOI 10.1007/978-3-642-24574-9

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011937539

CR Subject Classification (1998): H.4, H.3, I.2, D.2, C.2, H.5

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

© Springer-Verlag Berlin Heidelberg 2011

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

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

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

Printed on acid-free paper

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

Preface to ER 2011 Workshops, Posters, Demonstrations, and Industrial Papers

This book contains the proceedings of the workshops associated with the 30th International Conference on Conceptual Modeling (ER 2011), as well as the papers associated with the Posters and Demonstrations Session, and the Industrial Track of ER 2011.

As always, the aim of the workshops was to give researchers and participants a forum to present and discuss cutting edge research in conceptual modeling, and to pose some of the challenges that arise when applying conceptual modeling in less traditional areas. The workshops deal with theories, techniques, languages, methods, and tools for conceptual modeling and span a wide range of domains including web information systems, geographical information systems, business intelligence, software variability management, and ontologies. Some of the workshops were organized for the first time but others have a longer tradition. Six workshops were selected and organized after a call for workshop proposals:

- **WISM**: Web Information Systems Modeling
- **MORE-BI**: Modeling and Reasoning for Business Intelligence
- **Variability@ER**: Software Variability Management
- **Onto.Com**: Ontologies and Conceptual Modeling
- **SeCoGIS**: Semantic and Conceptual Issues in GIS
- **FP-UML**: Foundations and Practices of UML

In all, 31 workshop papers were accepted from a total of 88 submitted, making an overall acceptance rate of 35%.

For the posters and demonstrations session, nine contributions were selected. The aim of this session was to give researchers a highly interactive forum to show research work in progress and to demonstrate new and innovative applications.

The industrial track is the forum for presentations on innovative commercial software, systems, and services for all facets of conceptual modeling methodologies and technologies. For the 2011 edition, 6 papers were accepted from a total of 11 submitted. The papers present emerging industrial applications of conceptual modeling, as well as business intelligence applications.

Setting up workshops, a poster and demonstration session, and an industrial track requires a lot of effort and involves many people. We would like to thank the different Workshop Chairs and their Program Committees, the Program Committee of the posters and demonstrations, and the Program Committee of the industrial track for their diligence in selecting the papers in this volume.

We would also like to thank the main ER 2011 conference committees, particularly the Conference Co-chairs, Esteban Zimányi and Jean-Luc Hainaut, the Conference Program Co-chairs, Manfred Jeusfeld, Lois Delcambre, and Tok Wang Ling, and the Webmaster, Boris Verhaegen, for their support and for putting the program together.

July 2011

Olga De Troyer
Claudia Bauzer Medeiros
Roland Billen
Pierre Hallot
Alkis Simitsis
Hans Van Mingroot

ER 2011 Workshops, Posters, Demonstrations, and Industrial Track Organization

Workshop Co-chairs

Olga De Troyer
Claudia Bauzer Medeiros

Vrije Universiteit Brussel, Belgium
University of Campinas, Brazil

WISM 2011

WISM 2011 was organized by the Econometric Institute, Erasmus University Rotterdam, Netherlands; the Department of Computer Science, Delft University of Technology, Netherlands; and the Department of Computer Science, Namur University, Belgium.

Program Committee

Workshop Co-chairs

Flavius Frasinicar

Erasmus University Rotterdam,
The Netherlands

Geert-Jan Houben

Delft University of Technology,
The Netherlands

Philippe Thiran

Namur University, Belgium

Program Committee

Djamal Benslimane

University of Lyon 1, France

Sven Casteleyn

Polytechnic University of Valencia, Spain

Richard Chbeir

Bourgogne University, France

Olga De Troyer

Vrije Universiteit Brussel, Belgium

Roberto De Virgilio

Università di Roma Tre, Italy

Oscar Diaz

University of the Basque Country, Spain

Flavius Frasinicar

Erasmus University Rotterdam,
The Netherlands

Martin Gaedke

Chemnitz University of Technology, Germany

Irene Garrigos

Universidad de Alicante, Spain

Hyoil Han

LeMoyne-Owen College, USA

Geert-Jan Houben

Delft University of Technology,
The Netherlands

Zakaria Maamar

Zayed University, UAE

Michael Mrissa

University of Lyon 1, France

Moira Norrie

ETH Zurich, Switzerland

VIII Organization

Oscar Pastor
Dimitris Plexousakis
Jose Palazzo Moreira
de Oliveira
Azzurra Ragone
Hajo Reijers

Davide Rossi
Philippe Thiran
A Min Tjoa
Riccardo Torlone
Lorna Uden
Erik Wilde

Polytechnic University of Valencia, Spain
University of Crete, Greece

UFRGS, Brazil
Technical University of Bari, Italy
Eindhoven University of Technology,
The Netherlands
University of Bologna, Italy
Namur University, Belgium
Technical University of Vienna, Austria
Università di Roma Tre, Italy
Staffordshire University, UK
UC Berkeley, USA

External Reviewers

J.A. Aguilar
A. Bikakis
F. Valverde

MORE-BI 2011

Program Committee

Workshop Co-chairs

Ivan Jureta FNRS and Louvain School of Management,
University of Namur, Belgium
Stéphane Faulkner Louvain School of Management, University of
Namur, Belgium
Esteban Zimányi Université Libre de Bruxelles, Belgium

Steering Committee

Ivan Jureta FNRS and Louvain School of Management,
University of Namur, Belgium
Stéphane Faulkner Louvain School of Management,
University of Namur, Belgium
Esteban Zimányi Université Libre de Bruxelles, Belgium
Marie-Aude Aufaure Ecole Centrale Paris, France
Carson Woo Sauder School of Business, Canada

Program Committee

Alberto Abelló Universitat Politècnica de Catalunya, Spain
Daniele Barone University of Toronto, Canada
Ladjel Bellatreche Ecole Nationale Supérieure de Mécanique
et d'Aérotechnique, France

Sandro Bimonte	Cemagref, France
Farid Cerbah	Dassault Aviation, France
Dalila Chiadmi	Ecole Mohammadia d'Ingénieurs, Morocco
Alfredo Cuzzocrea	University of Calabria, Italy
Olivier Corby	INRIA, France
Marin Dimitrov	Ontotext, Bulgaria
Jérôme Euzenat	INRIA Grenoble Rhône-Alpes, France
Cécile Favre	Université Lyon 2, France
Xavier Franch	Universitat Politècnica de Catalunya, Spain
Octavio Glorio	University of Alicante, Spain
Matteo Golfarelli	University of Bologna, Italy
Gregor Hackenbroich	SAP, Germany
Dimitris Karagiannis	University of Vienna, Austria
Vijay Khatrı	Indiana University, USA
Isabelle Linden	University of Namur, Belgium
Patrick Marcel	Université François Rabelais de Tours, France
Maryvonne Miquel	Institut National des Sciences Appliquées de Lyon, France
Jose-Norberto Mazón	University of Alicante, Spain
John Mylopoulos	University of Trento, Italy
Carlos Ordonez	University of Houston, USA
Jeffrey Parsons	Memorial University of Newfoundland, Canada
Anna Perini	Fondazione Bruno Kessler, Italy
Stefano Rizzi	University of Bologna, Italy
Catherine Roussey	Université de Lyon, France
Anne Tchounikine	Institut National des Sciences Appliquées de Lyon, France
Maguelonne Teisseire	UMR TETIS, France
Juan-Carlos Trujillo Mondéjar	University of Alicante, Spain
Robert Wrembel	Poznan University of Technology, Poland

Variability@ER 2011

Program Committee

Workshop Co-chairs

Iris Reinhartz-Berger	University of Haifa, Israel
Arnon Sturm	Ben Gurion University of the Negev, Israel
Kim Mens	Université catholique de Louvain, Belgium

Program Committee

Felix Bachmann	SEI, USA
David Benavides	University of Seville, Spain
Jan Bosch	Intuit, Mountain View, USA
Paul Clements	SEI, USA

Anthony Cleve	University of Namur FUNDP, Belgium
Olga De Troyer	Vrije Universiteit Brussel, Belgium
Ulrich Eisenecker	Leipzig University, Germany
Øystein Haugen	University of Oslo, Norway
Brice Morin	University of Oslo, Norway
Linda Northrop	SEI, USA
Gilles Perrouin	University of Namur FUNDP, Belgium
Frank van der Linden	Philips, The Netherlands

Onto.Com 2011

Program Committee

Workshop Co-chairs

Giancarlo Guizzardi	Federal University of Espirito Santo, Brazil
Oscar Pastor	Polytechnic University of Valencia, Spain
Yair Wand	University of British Columbia, Canada

Program Committee

Alessandro Artale	Free University of Bolzano, Italy
Alex Borgida	Rutgers University, USA
Andreas Opdahl	University of Bergen, Norway
Bert Bredeweg	University of Amsterdam, The Netherlands
Brian Henderson-Sellers	University of Technology Sydney, Australia
Carson Woo	University of British Columbia, Canada
Chris Partridge	BORO Solutions, UK
Claudio Masolo	Laboratory for Applied Ontology (ISTC-CNR), Italy
Colin Atkinson	University of Mannheim, Germany
David Embley	Brigham Young University, USA
Dragan Gašević	Athabasca University, Canada
Fred Fonseca	Penn State University, USA
Gerd Wagner	Brandenburg University of Technology, Germany
Giancarlo Guizzardi	Federal University of Espirito Santo, Brazil
Heinrich Herre	University of Leipzig, Germany
Heinrich Mayr	University of Klagenfurt, Austria
Jean-Marie Favre	University of Grenoble, France
Jeffrey Parsons	Memorial University of Newfoundland, Canada
Joerg Evermann	Memorial University of Newfoundland, Canada
John Mylopoulos	University of Trento, Italy
Jose Palazzo M. de Oliveira	Federal University of Rio Grande do Sul, Brazil
Leo Orbst	MITRE Corporation, USA
Matthew West	Information Junction, UK
Michael Rosemann	University of Queensland, Australia

Nicola Guarino	Laboratory for Applied Ontology (ISTC-CNR), Italy
Oscar Pastor	Polytechnic University of Valencia, Spain
Palash Bera	Texas A&M International University, USA
Peter Green	University of Queensland, Australia
Peter Rittgen	University College Boras, Sweden
Pnina Soffer	University of Haifa, Israel
Richard Dapoigny	University of Savoie, France
Simon Milton	University of Melbourne, Australia
Stephen Liddle	Brigham Young University, USA
Vadim Ermolayev	Zaporozhye National University, Ukraine
Veda Storey	Georgia State University, USA
Vijay Khatri	Indiana University, USA
Yair Wand	University of British Columbia, Canada
Wolfgang Hesse	University of Marburg, Germany

SeCoGIS 2011

Program Committee

Workshop Co-chairs

Roland Billen	Université de Liège, Belgium
Esteban Zimányi	Université Libre de Bruxelles, Belgium
Pierre Hallot	Université de Liège, Belgium

Steering Committee

Claudia Bauzer Medeiros	University of Campinas, Brazil
Michela Bertolotto	University College Dublin, Ireland
Jean Brodeur	Natural Resources Canada
Christophe Claramunt	Naval Academy Research Institute, France
Christelle Vangenot	Université de Genève, Switzerland
Esteban Zimányi	Université Libre de Bruxelles, Belgium

Program Committee

Alia Abdelmoty	Cardiff University, UK
Gennady Andrienko	Fraunhofer Institute IAIS, Germany
Natalia Andrienko	IAIS Fraunhofer, Germany
David Bennett	The University of Iowa, USA
Michela Bertolotto	University College Dublin, Ireland
Roland Billen	Université de Liège, Belgium
Patrice Boursier	University of La Rochelle, France
Jean Brodeur	Natural Resources Canada
Bénédicte Bucher	Institut Géographique National, France
Yvan Bédard	Laval University, Canada

Ricardo Rodrigues Ciferri	UFSCar, Brazil
Christophe Claramunt	Naval Academy Research Institute, France
Eliseo Clementini	University of L'Aquila, Italy
Maria Luisa Damiani	University of Milan, Italy
Clodoveu Davis	Federal University of Minas Gerais, Brazil
Fernando Ferri	IRPPS-CNR, Italy
Jugurta Lisboa Filho	Universidade Federal de Viçosa, Brazil
Anders Friis-Christensen	Denmark
Pierre Hallot	Université de Liège, Belgium
Bo Huang	The Chinese University of Hong Kong, Hong Kong, China
Marinos Kavouras	NTUA, Greece
Ki-Joune Li	Pusan National University, South Korea
Thérèse Libourel	Université Montpellier II, France
Miguel Luaces	University of A Coruña, Spain
Jose Macedo	Federal University of Ceara, Brazil
Antonio Miguel Vieira Monteiro	INPE - National Institute for Space Research, Brazil
Pedro Rafael Muro Medrano	University of Zaragoza, Spain
Dimitris Papadias	HKUST, Hong Kong, China
Dieter Pfoser	IMIS/ATHENA, Greece
Markus Schneider	University of Florida, USA
Sylvie Servigne-Martin	LIRIS, INSA-Lyon, France
Emmanuel Stefanakis	Harokopio University of Athens, Greece
Kathleen Stewart-Hornsby	University of Iowa, USA
Andrea Rodriguez Tastets	University of Concepcion, Chile
Kerry Taylor	CSIRO ICT Centre, Australia
Peter Van Oosterom	Delft University of Technology, OTB, GIS Technology, The Netherlands
Christelle Vangenot	University of Geneva, Switzerland
Nancy Wiegand	University of Wisconsin, USA
Stephan Winter	The University of Melbourne, Australia
Esteban Zimányi	Université Libre de Bruxelles, Belgium

FP-UML 2011

Program Committee

Workshop Co-chairs

Guido Geerts	University of Delaware, USA
Matti Rossi	Aalto University, Finland

Steering Committee

Juan Trujillo	University of Alicante, Spain
Il-Yeol Song	Drexel University, USA

Jeff Parsons
Andreas L. Opdahl

Memorial University of Newfoundland, Canada
University of Bergen, Norway

Program Committee

Doo-Hwan Bae	EECS Dept. KAIST, South Korea
Michael Blaha	OMT Associates Inc., USA
Cristina Cachero	University of Alicante, Spain
Gill Dobbie	University of Auckland, New Zealand
Dirk Draheim	Inst. of Computer Science, Freie Univ. Berlin, Germany
Eduardo Fernandez	University of Castilla La Mancha, Spain
Frederik Gailly	Vrije Universiteit Brussel, Belgium
Paolo Giorgini	University of Trento, Italy
Jaime Gómez	University of Alicante, Spain
Peter Green	University of Queensland, Australia
Manfred Jeusfeld	Tilburg University, The Netherlands
Ludwik Kuzniarz	Blekinge Institute of Technology, Sweden
Jens Lechtenböcker	University of Munster, Germany
Pericles Loucopoulos	University of Manchester, UK
Kalle Lyytinen	Case Western Reserve University, USA
Hui Ma	Massey University, New Zealand
Antoni Olive	Polytechnic University of Catalonia, Spain
Oscar Pastor	Polytechnic University of Valencia, Spain
Witold Pedrycz	University of Alberta, Canada
Mario Piattini	University of Castilla La Mancha, Spain
Colette Rolland	Université de Paris, France
Manuel Serrano	University of Castilla La Mancha, Spain
Keng Siau	University of Nebraska-Lincoln, USA
Bernhard Thalheim	Universitaet zu Kiel, Germany
A Min Tjoa	Technical University of Vienna, Austria
Ambrosio Toval	University of Murcia, Spain
Panos Vassiliadis	University of Ioannina, Greece
Harry (Jiannan) Wang	University of Delaware, USA

Posters and Demonstrations

Program Committee

Co-chairs

Roland Billen	Université de Liège, Belgium
Pierre Hallot	Université de Liège, Belgium

Program Committee

Renata Araújo	Universidade Federal do Estado do Rio de Janeiro, Brazil
Michael Blaha	OMT Associates, USA
Irene Garrigos	University of Alicante, Spain
Pierre Geurts	Université de Liège, Belgium
Sergio Lifschitz	Pontificia Universidade Católica do Rio de Janeiro, Brazil
Jose-Norberto Mazon	Universidad de Alicante, Spain
Sergio Lujan Mora	Universidad de Alicante, Spain
German Shegalov	Oracle, USA
Alkis Simitsis	HP Labs, USA
David Taniar	Monash University, Australia

Industrial Track

Program Committee

Co-chairs

Alkis Simitsis	HP Labs, USA
Hans Van Mingroot	IBM, Belgium

Program Committee

Phil Bernstein	Microsoft Research, USA
Umeshwar Dayal	HP Labs, USA
Howard Ho	IBM Research, USA
Neoklis Polizotis	UCSC, USA
Erik Proper	Public Research Centre - Henri Tudor, Luxembourg
Sabri Skhiri	Euranova, Belgium
Jan Verelst	University of Antwerp, Belgium)

Table of Contents

WISM 2011 - The Eighth International Workshop on Web Information Systems Modeling

Preface to WISM 2011	1
<i>Flavius Frasincar, Geert-Jan Houben, and Philippe Thiran</i>	

Social Networks and Data Interoperability in Web Information Systems

Academic Social Networks	2
<i>Jose Palazzo Moreira de Oliveira, Giseli Rabello Lopes, and Mirella Moura Moro</i>	

Conceptual Modelling for Web Information Systems: What Semantics Can Be Shared?	4
<i>Simon McGinnes</i>	

Requirements Analysis, User Interaction, and Service Composition in Web Information Systems

A Goal-Oriented Approach for Optimizing Non-functional Requirements in Web Applications	14
<i>José Alfonso Aguilar, Irene Garrigós, and Jose-Norberto Mazón</i>	

Yet Another BPEL Extension for User Interactions	24
<i>Mohamed Boukhebouze, Waldemar Pires Ferreira Neto, and Lim Erbin</i>	

Semantics-Enabled Web API Organization and Recommendation	34
<i>Devis Bianchini, Valeria De Antonellis, and Michele Melchiori</i>	

MORE-BI 2011 - The First International Workshop on Modeling and Reasoning for Business Intelligence

Preface to MORE-BI 2011	44
<i>Ivan J. Jureta, Stéphane Faulkner, and Esteban Zimányi</i>	

Formal Concept Analysis for Qualitative Data Analysis over Triple Stores	45
<i>Frithjof Dau and Barış Sertkaya</i>	

Semantic Cockpit: An Ontology-Driven, Interactive Business Intelligence Tool for Comparative Data Analysis	55
<i>Bernd Neumayr, Michael Schrefl, and Konrad Linner</i>	
A Model-Driven Approach for Enforcing Summarizability in Multidimensional Modeling	65
<i>Jose-Norberto Mazón, Jens Lechtenbörger, and Juan Trujillo</i>	
Repairing Dimension Hierarchies under Inconsistent Reclassification	75
<i>Mónica Caniupán and Alejandro Vaisman</i>	
GrHyMM: A Graph-Oriented Hybrid Multidimensional Model	86
<i>Francesco Di Tria, Ezio Lefons, and Filippo Tangorra</i>	
Ontologies and Functional Dependencies for Data Integration and Reconciliation	98
<i>Abdelghani Bakhtouchi, Ladjel Bellatreche, and Yamine Ait-Ameur</i>	
A Comprehensive Framework on Multidimensional Modeling	108
<i>Oscar Romero and Alberto Abelló</i>	

Variability@ER’11 - Workshop on Software Variability Management

Preface to Variability@ER’11	118
<i>Iris Reinhartz-Berger, Arnon Sturm, and Kim Mens</i>	
ISO Initiatives on Software Product Line Engineering: Vision and Current Status: Invited Talk for Variability@ER2011	119
<i>Timo K. Käkölä</i>	
Feature Modeling Tools: Evaluation and Lessons Learned	120
<i>Mohammed El Dammagh and Olga De Troyer</i>	
Service Variability Patterns	130
<i>Ateeq Khan, Christian Kästner, Veit Köppen, and Gunter Saake</i>	
An Overview of Techniques for Detecting Software Variability Concepts in Source Code	141
<i>Angela Lozano</i>	
Variability in Multi-tenant Environments: Architectural Design Patterns from Industry	151
<i>Jaap Kabbedijk and Slinger Jansen</i>	

Onto.Com 2011 - International Workshop on Ontologies and Conceptual Modeling

Preface to Onto.Com 2011	161
<i>Giancarlo Guizzardi, Oscar Pastor, and Yair Wand</i>	
Experimental Evaluation of an Ontology-Driven Enterprise Modeling Language	163
<i>Frederik Gailly and Geert Poels</i>	
Levels for Conceptual Modeling	173
<i>Claudio Masolo</i>	
Principled Pragmatism: A Guide to the Adaptation of Ideas from Philosophical Disciplines to Conceptual Modeling	183
<i>David W. Embley, Stephen W. Liddle, and Deryle W. Lonsdale</i>	
Ontological Usage Schemes: A Working Proposal for the Ontological Foundation of Language Use	193
<i>Frank Loebe</i>	
Gene Ontology Based Automated Annotation: Why It Isn't Working ...	203
<i>Matthijs van der Kroon and Ana M. Levin</i>	
Formal Ontologies, Exemplars, Prototypes	210
<i>Marcello Friscone and Antonio Lieto</i>	
Unintended Consequences of Class-Based Ontological Commitment	220
<i>Roman Lukyanenko and Jeffrey Parsons</i>	

SeCoGIS 2011 - The Fifth International Workshop on Semantic and Conceptual Issues in GIS

Preface to SeCoGIS 2011	230
<i>Esteban Zimányi, Roland Billen, and Pierre Hallot</i>	
Referring Expressions in Location Based Services: The Case of the 'Opposite' Relation	231
<i>Phil Bartie, Femke Reitsma, Eliseo Clementini, and Simon Kingham</i>	
Cognitive Adequacy of Topological Consistency Measures	241
<i>Nieves R. Brisaboa, Miguel R. Luaces, and M. Andrea Rodríguez</i>	
The Neighborhood Configuration Model: A Framework to Distinguish Topological Relationships between Complex Volumes	251
<i>Tao Chen and Markus Schneider</i>	
Reasoning with Complements	261
<i>Max J. Egenhofer</i>	

Towards Modeling Dynamic Behavior with Integrated Qualitative Spatial Relations 271
Stefan Mitsch, Werner Retschitzegger, and Wieland Schwinger

Transforming Conceptual Spatiotemporal Model into Object Model with Semantic Keeping 281
Chamseddine Zaki, Myriam Servières, and Guillaume Moreau

FP-UML - The Seventh International Workshop on Foundations and Practices of UML

Preface to FP-UML 2011 291
Guido L. Geerts and Matti Rossi

On Automated Generation of Associations in Conceptual Database Model 292
Drazen Brdjanin and Slavko Maric

Specification and Utilization of Core Assets: Feature-Oriented vs. UML-Based Methods 302
Iris Reinhartz-Berger and Arava Tsoury

Actor-eUML for Concurrent Programming 312
Kevin Marth and Shangping Ren

Posters and Demonstrations

Preface to the Posters and Demonstrations 322
Roland Billen and Pierre Hallot

An Eclipse Plugin for Validating Names in UML Conceptual Schemas 323
David Aguilera, Raúl García-Ranea, Cristina Gómez, and Antoni Olivé

KEYRY: A Keyword-Based Search Engine over Relational Databases Based on a Hidden Markov Model 328
Sonia Bergamaschi, Francesco Guerra, Silvia Rota, and Yannis Velegakis

VirtualEMF: A Model Virtualization Tool 332
Cauê Clasen, Frédéric Jouault, and Jordi Cabot

Towards a Model-Driven Framework for Web Usage Warehouse Development	336
<i>Paul Hernández, Octavio Glorio, Irene Garrigós, and Jose-Norberto Mazón</i>	
CRESCO: Construction of Evidence Repositories for Managing Standards Compliance	338
<i>Rajwinder Kaur Panesar-Walawege, Torbjørn Skyberg Knutsen, Mehrdad Sabetzadeh, and Lionel Briand</i>	
Modeling Approach for Business Networks with an Integration and Business Perspective	343
<i>Daniel Ritter and Ankur Bhatt</i>	
Mosto: Generating SPARQL Executable Mappings between Ontologies	345
<i>Carlos R. Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo</i>	
The CSTL Processor: A Tool for Automated Conceptual Schema Testing	349
<i>Albert Tort, Antoni Olivé, and Maria-Ribera Sancho</i>	
A Tool for Filtering Large Conceptual Schemas	353
<i>Antonio Villegas, Maria-Ribera Sancho, and Antoni Olivé</i>	
Industrial Track	
Preface to the Industrial Track	357
<i>Alkis Simitsis and Hans Van Mingroot</i>	
Business Intelligence	
QBX: A CASE Tool for Data Mart Design	358
<i>Antonino Battaglia, Matteo Golfarelli, and Stefano Rizzi</i>	
The Meta-Morphing Model Used in TARGIT BI Suite	364
<i>Morten Middelfart and Torben Bach Pedersen</i>	
Tool Support for Technology Scouting Using Online Sources	371
<i>Elena Tsiporkova and Tom Tourwé</i>	
Industrial Applications of Conceptual Modeling	
Governance Issues on Heavy Models in an Industrial Context	377
<i>Sabri Skhiri, Marc Delbaere, Yves Bontemps, Gregoire de Hemptinne, and Nam-Luc Tran</i>	

High Quality Technical Documentation for Large Industrial Plants Using an Enterprise Engineering and Conceptual Modeling Based Software Solution	383
<i>Steven J.H. van Kervel</i>	
Publishing Open Data and Services for the Flemish Research Information Space.....	389
<i>Christophe Debruyne, Pieter De Leenheer, Peter Spyns, Geert van Grootel, and Stijn Christiaens</i>	
Author Index	395

Preface to WISM 2011

The international workshop on Web Information Systems Modeling (WISM) aims to uncover the most recent developments in the field of model-driven design of Web Information Systems (WIS). This is the eighth edition of the workshop, after seven successful editions organized in Vancouver (2010), Amsterdam (2009), Barcelona (2008), Trondheim (2007), Luxembourg (2006), Sydney (2005), and Riga (2004).

The extended abstract of the invited talk by de Oliveira et al. proposes concepts for Social Network Analysis to be applied for online Academic Social Networks. More specifically, it investigates the network analysis, work dissemination, and partner recommendation in an academic context. The social network based on co-author relationships among researchers is used as an example.

The first paper by McGinnes proposes research directions for handling data exchange between heterogeneous WIS. These research directions are developed around the idea of building a basic ontology containing a simple set of shared concepts. The author argues that knowing the types of the shared concepts is not enough to ensure WIS interoperability, as WIS need be able to programmatically handle these types.

The second paper by Aguilar et al. suggests a solution for selecting the appropriate WIS requirements in order to maximize the user satisfaction with respect to a prioritized list of non-functional requirements. The approach is based on detecting the Pareto front of candidate solutions and selecting the ones that ensure the fulfillment of the highly ranked non-functional requirements. For demonstration purposes the authors use the requirements of an online Conference Management System represented using the i^* framework.

The third paper by Boukhebouze et al. proposes UI-BPEL, an extension of BPEL that allows the modeling of user interaction (providing, selecting, and getting data by the user) in Web service composition. The main goal of UI-BPEL is to allow the generation of user interfaces based on the described user interactions. This process is composed of two steps: the generation of an abstract user interface independent of any interaction modality and computing platform, and the generation of a concrete user interface based on the abstract user interface that is adapted to the user specific context.

The fourth paper by Bianchini et al. describes an approach for rapid WIS development by composing existing Web APIs. The composition is based on selection patterns that exploit functional similarities and coupling criteria. For this purpose Web APIs are annotated using existing lightweight semantic models extended with semantically annotated events.

We do hope that the previous synopsis has triggered the reader's interest to have a closer look at the workshop proceedings. Last, we would also like to thank all the authors, reviewers, and participants for their input and support for the workshop.

July 2011

Flavius FrasinCAR
Geert-Jan Houben
Philippe Thiran

Academic Social Networks

Jose Palazzo Moreira de Oliveira¹, Giseli Rabello Lopes¹,
and Mirella Moura Moro^{2,*}

¹ Universidade Federal do Rio Grande do Sul - UFRGS
Porto Alegre, Brazil

{palazzo, grlopes}@inf.ufrgs.br

² Universidade Federal de Minas Gerais - UFMG
Belo Horizonte, Brazil
mirella@dcc.ufmg.br

Abstract. The growth of Web 2.0 encouraged the consideration not only of technological and content aspects but also the social interactions and its relational aspects. Researches in the academic context also have followed this trend. Methods and applications have been proposed and adapted to consider “social aspects” by different modes. In this paper, we present an overview of our publications focusing on Academic Social Networks including proposals of analysis, dissemination and recommendation for this context.

Keywords: Social Networks, Research, Collaboration.

1 Challenges and Research

In the Web 2.0 perspective not only the technological and content aspects but also the social interactions and its relational aspects must be considered. In this scenario, emerged applications with social aspects including Web-based communities and Social Networks. The Social Network Analysis (SNA) is based on the assumption that the relationship’s importance between interaction elements is a central point for analysis. The increasing interest in SNA research is encouraged by the popularization of online social networks. Our example of SNA concepts application is to model an Academic Social Network representing collaboration among researchers by the use of co-author relationships. Following, we present an overview of our publications focusing on Academic Social Networks including models for analysis, dissemination and recommendation in this context.

In a first paper [2] we introduced a set of challenges for developing a dissemination service over a Web collaborative network. Specific metrics were defined for working on a co-authorship network. As a case study, we build such a network using those metrics and compare it to a manually built one. Specifically, once we build a collaborative network and verify its quality, the overall effectiveness of the dissemination services will also be improved. A dissemination service is established by data producers and consumers. Specifically, consumers subscribe to

* This research is partially supported by CNPq (Brazil) and is part of the InWeb project.

the service by defining a profile, which is usually composed of different queries. As the producers introduce new data, usually through messages, the dissemination service evaluates each message against the profiles. Once there is a match between a profile and a message, the service sends that message to the profile's consumer. The results are evaluated by the access patterns and users accordance with the quality of disseminated papers and, more important, by the increase in the cooperation pattern among inter-institutional researchers.

In a second paper [1] we presented a Collaboration Recommendation on Academic Social Networks. In the academic context, scientific research works are often performed through collaboration and cooperation between researchers and research groups. Researchers work in various subjects and in several research areas. Identifying new partners to execute joint research and analyzing the level of cooperation of the current partners can be very complex tasks. Recommendation of new collaborations may be a valuable tool for reinforcing and discovering such partners. In this work, we have presented the details of an innovative approach to recommend scientific collaborations on the context of Social Networks. Specifically, we introduce the architecture for such approach and the metrics involved in recommending collaborations. A case study to validate the approach, using researchers associated to the InWeb project, is also presented.

Finally we developed an analysis [3] including a new relationship weighted approach. These weights aim to measure the importance of the relational ties between actors and are important to be considered in an Academic Social Network. In our case study, we demonstrated the possibility of uses of the Gini coefficient to analyze weighted Social Networks. Gini coefficient is applied to measure the homogeneity level of the collaboration, i.e., if only a few researchers keep a good level of collaboration or if all researchers of the network, indeed, are contributing for the group.

The developed researches demonstrate the importance of applying the concepts of Social Networks to the analysis and quality improvement of academic groups. Additionally, we proposed new measures to adequately quantify the groups' quality for generating ranks.

References

1. Lopes, G.R., Moro, M.M., Wives, L.K., de Oliveira, J.P.M.: Collaboration recommendation on academic social networks. In: Trujillo, J., Dobbie, G., Kangassalo, H., Hartmann, S., Kirchberg, M., Rossi, M., Reinhartz-Berger, I., Zimányi, E., Frasin-car, F. (eds.) ER 2010. LNCS, vol. 6413, pp. 190–199. Springer, Heidelberg (2010)
2. Lopes, G.R., Moro, M.M., Wives, L.K., de Oliveira, J.P.M.: Cooperative authorship social network. In: Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management, Buenos Aires, Argentina, May 17-20. CEUR Workshop Proceedings, vol. 619 (2010), CEUR-WS.org
3. Lopes, G.R., da Silva, R., de Oliveira, J.P.M.: Applying gini coefficient to quantify scientific collaboration in researchers network. In: Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011, pp. 1–68. ACM, New York (2011)

Conceptual Modelling for Web Information Systems: What Semantics Can Be Shared?

Simon McGinnes

School of Computer Science and Statistics, Trinity College Dublin, Ireland
Simon.McGinnes@tcd.ie

Abstract. There is an increasing need to allow software applications to exchange data, which usually requires the negotiation of meanings between incompatible conceptual models. Theoretically, the concepts in one application can be mapped to those in another, but this can be challenging in practice. The problem is more fundamental than “information plumbing”; it requires reconciliation between alternative and possibly conflicting ways of viewing the world. Ontologies and the Semantic Web have been proposed as potential solutions to the information exchange problem. This research investigates from first principles what kinds of data exchange are possible, with the aim of analyzing the issue in a useful way for the developers and users of web-based information systems. The analysis suggests particular means of facilitating data exchange involving the use of a simple set of shared basic-level categories.

Keywords: web information systems, ontologies, data models, conceptual modelling, Semantic Web.

1 Introduction

Now is an important moment in human development. Thanks to the internet, for the first time in history we have the capacity to share data on a massive scale, to move information in digital form more or less anywhere we want at the press of a button. But with this new ability comes the need to think in new ways about information exchange. Our conventional view of sharing information developed in the low bandwidth world of conversation, storytelling, books and newspapers. Trying to apply the same ideas to the internet may risk creating the potential for confusion. Recently, efforts have been made to automate the sharing of information using technologies such as the Semantic Web, microformats and web services. As we begin to use these technologies it is important to be clear about what we mean when we talk about sharing information and what kinds of information can feasibly be shared.

This position paper aims to clarify these questions in a useful way for the developers and users of web-based information systems. The discussion is necessarily rather philosophical, but it is made as practical as possible because sharing data is inherently a practical issue. In fact, the sole reason for sharing data is to facilitate action. Viewing shared data as passive information is to miss the point; the significance of sharing data is in the potential it creates for action.

1.1 The Need for Data Sharing

There is a greater need for data sharing within and particularly between organizations than ever before. Historically, most organizations have used a portfolio of information systems for different purposes, so corporate data has been locked into a number of separate, mutually-incompatible data structures. Organizations need their systems to work together, but sharing data between heterogeneous applications is rarely straightforward. In an attempt to achieve integration, many organizations use enterprise software applications, which address requirements across a range of business processes. However, the adoption of enterprise software products tends to lock organizations into a single supplier and prevents the selection of best-of-breed solutions. Both approaches—the “application portfolio” approach, and the adoption of enterprise software—can be expensive, and neither is ideal.

The situation is exacerbated when it comes to sharing data between organizations, or between organizations and individuals. There is at present no universal inter-organizational equivalent to the enterprise software solution. Therefore organizations have little choice but to make their mutually-incompatible applications work together. Historically, two factors have presented a barrier to interoperation. One is physical incompatibility: if two systems are physically disconnected then they cannot communicate. The Internet and related technologies have largely solved that problem. However, as soon as applications can physically exchange data then the need for semantic compatibility becomes paramount. This problem is less easily solved.

For example, consider two information systems which need to interoperate: (a) a sales order processing system containing data about product types, suppliers, customers, orders and employees, and (b) an accounting system with data on transactions, debtors, creditors and accounts. Although the two sets of concepts describe the same real-world phenomena (people, organizations and business transactions), the systems conceptualize the data very differently. The systems are conceptually incompatible, even though they store data about the same things. The incompatibility presents a barrier to data exchange between the applications. Unless a programmer crafts a suitable custom interface between the two applications, which translates between the two ways of conceptualizing the underlying business entities, it will be difficult to share data between them; the applications are built around concepts that do not map to one another in any straightforward way.

Of course, the problem is not confined to this trivial example of sales and accounting systems. It is rife, because most application software is structured around idiosyncratic, domain-specific concepts. This seemingly-benign design practice, which has long been the norm in information systems engineering, guarantees that different programs will tend to be semantically incompatible. But, regardless of the pros and cons of using ad hoc concepts in application design, we need to find ways of integrating conceptually-incompatible systems. This is where technologies such as ontologies and the Semantic Web offer some hope.

1.2 Structure and Terminology

The paper is structured as follows. Section 2 explores from first principles the ways in which data exchange between semantically-incompatible systems can and cannot be

achieved. Section 3 discusses ontologies, the Semantic Web and related technologies in the light of this analysis. Section 4 outlines some possible changes to design practice suggested by this analysis, which might facilitate semantic interoperability. Section 5 concludes with a summary of findings and limitations, and suggests directions for further work.

Two commonly-used terms are avoided in this paper, because their ambiguity could contribute to confusion. The first is *information*; this is a term from everyday language which has been co-opted in IS/IT with a variety of meanings. It can refer to essentially the same thing as data, or to any digital signal, or to text, or to facts that have particular significance. The second term is *semantics*; this term has been adopted by particular academic communities, with the result that its meaning is rather blurred. Because the meanings of both terms are central to the arguments in this paper, we avoid using them altogether and will use other, less ambiguous terms as appropriate.

2 Shared Meanings

When talking about data and information systems it is important to distinguish clearly between real-world things (“non-information resources” in linked data terminology), the signs that represent them, and mental states which corresponding to signs and real-world things. For example, the term “IBM” is a sign, which corresponds to a particular organization known as IBM (a real-world thing). An observer may have an idea of the organization known as IBM, and this is a mental state. This three-way relationship is sometimes encapsulated in the “semiotic triangle” [1].

Information systems store signs (bits, bytes, images, text, etc.) which represent real-world things and mental states. When people communicate, they do so using signs such as words and gestures. In all of these cases, signs are manipulated in the hope of evoking mental states. This much is uncontroversial; in the following discussion, we consider how mental states can be evoked through the use of signs by people and machines. We are interested in knowing how software applications can exchange data despite conceptual incompatibility. In order to understand that, we need to understand how meaning is transmitted. Since humans often try to transmit meaning to one another, it is helpful first to consider how this works.

2.1 How Do People Share Meanings?

Figure 1 illustrates two views of the communication of meaning between individuals. In view (a) meanings are shared between individuals, and the purpose of language is merely to evoke these shared meanings. This view is reminiscent of Jung’s collective unconscious or the concept of the platonic ideal. In view (b) meaning is transmitted directly by language and so there is no particular need for meanings to be shared.

We argue that, at best, both of these views are unhelpful. There is no evidence for view (a); despite centuries of philosophical theorizing, no scientific research has uncovered any mechanism by which meanings might exist “out there” and be shared between individuals. As for view (b), meaning is not something which can flow; it is an experience. When we talk of conveying meaning, this is a figure of speech and not

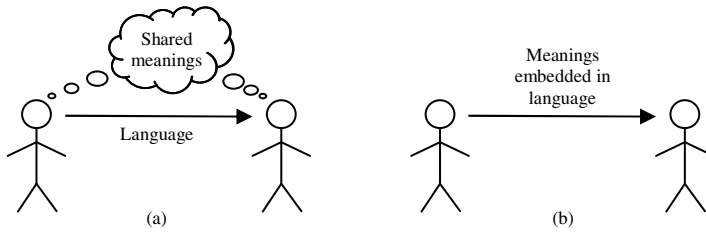


Fig. 1. Models for the communication of meaning between people

to be taken literally. Language cannot literally carry meaning since it consists only of signs: sounds and gestures. The medium in this case is *not* the message. Meaning arises only as an experience in each observer when he or she hears the sounds and observes the gestures. The mental states that are evoked by these perceptions give rise to mental states which we experience as meaning (Figure 2). According to this view, no meanings can be shared; experience and memory are private mental states of each individual. Language, which consists of signs, flows between individuals and evokes the experience of meaning separately in each individual.

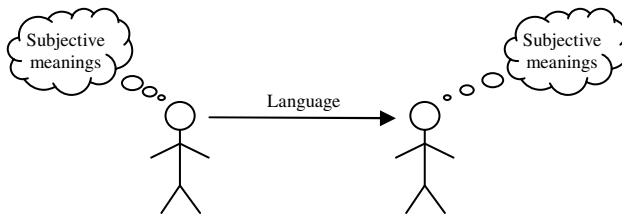


Fig. 2. Improved model for the communication of meaning between people

However, this raises the question of how humans communicate at all, if there are no shared meanings. The answer is that all meaning is subjective, yet we can assume that much of our experience is similar. Humans communicate imperfectly and misunderstanding is common. But we can proceed for much of the time as if we share common meanings, because we are physiologically similar to one another and have similar formative experiences. To give a trivial example, it is a reasonable working assumption that we all mean the same thing by “red”. This assumption is in fact incorrect, because evidence tells us that many people are colorblind and cannot experience red in the same way as non-colorblind individuals. It is also likely that individual variations in the perception of colors exist quite apart from colorblindness. Nevertheless, it is more helpful to assume common meanings than to assume we have no shared experience. The same argument can be extended to many concepts; while there may be disagreement between individuals on the interpretation of particular signals, it is more beneficial to attempt communication than not to.

Unlike computers, people can rely on having a common set of concepts, because we all share the experience of living in the same world (roughly), are members of the same species, and share elements of brain function. Hence our most fundamental

concepts (whether innate or learned) tend to be similar; most people would agree on what a person is, for example. These ideas provide the context for thought and tell us generally how to behave in relation to any given thing or situation that we encounter.

2.2 How Is Meaning Experienced?

A related question is how each individual's subjective meanings are stored and what form they take when consciously experienced. This is an area of academic debate, but one broad trend in our understanding of meaning can be identified. Early theories of cognition postulated that meaning stems from conceptual structures in the mind. In psychology, the spreading activation model is an example of such a theory; nodes representing distinct concepts are connected explicitly, reflecting associative links between ideas [2]. Similar thinking in computer science gave rise to conceptual graphs and the ideas of schemata and frames [3]. There have even been suggestions that the unconscious mind performs computations using symbolic logic [4].

However, despite a great deal of looking, neuroscience has not found evidence for conceptual structures in the mind. It is becoming apparent that meaning arises in a rather different way. Rather than activating concepts, perceptions appear instead to elicit the recall of prior experience in a holistic manner. When we observe a situation or hear language we recall a complex of memories with associated emotional states. To recall is to re-experience, and it is this conscious re-experiencing of prior experiences which we know as meaning.

Recall occurs on the basis of similarity using perceptual feature-matching processes, which operate on multiple levels. This allows us to recall memories because of literal as well as abstract (analogical) similarities—one situation may be quite unlike another in detail and yet we still are reminded, because of more abstract similarities [5]. This suggests that, as a general principle, thinking about a concept does not depend on definitions or on analytical thinking; it involves the retrieval of prior experience on the basis of perceptual similarity. The definition of a concept emerges and becomes crisp only when we try to define it consciously.

Although neuroscience has not found evidence for the existence of conceptual structures, the results of studies (some of which use brain imaging) suggest that the primate brain possesses “hardwired” semantic regions which process information about particular subjects such as people, animals, tools, places and activities [6]. There is debate about the interpretation of these results, but the implication is that the brain automatically segregates (on the basis of feature matching) cognitive processing into certain broad categories. Categorization occurs unconsciously and the categories appear to be innate, not learned. They correspond to concrete, everyday ideas rather than abstractions. We can hypothesize that other concepts—more specialized ideas like *customer* and *account*, for example—are learned, and become associated with the corresponding basic-level innate categories.

2.3 How Do Computers Share Meaning?

To reiterate, we are interested in knowing how software applications can exchange data despite conceptual incompatibility. Software applications cannot be said to experience, at least in the sense meant above, and so they cannot be considered

capable of experiencing meaning in the way that a human does. The closest equivalent to meaning for a software application is when it has been programmed to act on specific types of data; we may then say (figuratively) that the application “understands” that type of data. This figurative understanding consists of two elements: (a) being programmed to accept data with a particular structure, and (b) being programmed to deal with that type of data appropriately. However, we must be clear that the ability to handle each specific type of data requires explicit programming.

So, if two applications “understand” different types of data (and so are conceptually incompatible) how can they exchange their own equivalent of meaning? As before, we consider different models of communication. In Figure 3, view (a) suggests that it is sufficient merely to exchange data. View (b) suggests that metadata should also be included with the data; its function is to explain the structure and purpose of the data, so that the receiving application can process it properly.

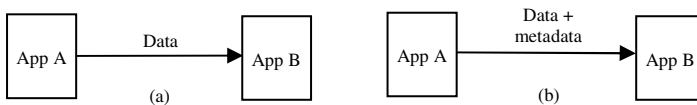


Fig. 3. Models for the communication of meaning between applications

View (a) is clearly insufficient since (to use our earlier example) the accounting system will not recognize data about customers, orders and so on, and will therefore be unable to do anything useful with them. Therefore some explanation of these concepts is needed. However, view (b) is also incomplete, because any metadata must be expressed in terms of concepts that the receiving application understands. The accounting system will understand what to do with data about customers and orders only if the metadata explains the concepts *customer* and *order* in terms of the concepts *account* and *transaction*, or in terms of other constructs which the accounting system understands. That would require the sending application to have knowledge (e.g. of accounts and transactions) which it does not have.

For two conceptually-incompatible applications to exchange meanings, they must both “understand” how to process particular types of data (Figure 4). This requires that both be programmed with the ability to handle those types of data. For that to be the case, both applications must share particular conceptual structures and must contain program code which can deal with the corresponding data. But that would mean that *the applications would no longer be conceptually incompatible*. Of course, this is a contradiction. It means that the only way for conceptually-incompatible applications to exchange data is by becoming conceptually compatible.

This analysis tells us that conceptually-incompatible applications can never exchange data unless they become conceptually compatible first. It makes no difference how complicated the data exchange method is or how much markup is included. When data is exchanged it can be processed meaningfully only if the sending and receiving applications are conceptually compatible and this situation can be achieved only in advance of the exchange, through programming. In the next section, we consider the implications of this finding for data exchange technologies.

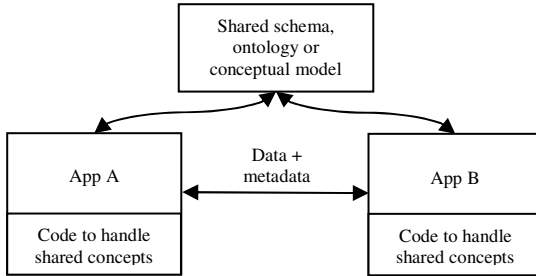


Fig. 4. Improved model for communication of meaning between applications

3 Implications for Data Exchange Technologies

3.1 Ontologies and the Semantic Web

Ontologies have been proposed as a solution to data transfer problems. An ontology is a formal description of concepts and their relationships. Unlike conceptual models, ontologies are generally prepared independently of particular applications. Upper ontologies such as SUMO contain hundreds of thousands of interconnected concepts; they aim to encapsulate broad areas of knowledge in model form. They contain both highly-generic and highly-specific abstractions; for example, SUMO contains the hierarchy *sentient agent* → *cognitive agent* → *human* → *internet user* (the → symbol can be read as “may be a”). Domain ontologies address more limited subject areas. Some researchers are trying to reconcile upper ontologies with domain ontologies; others are developing intermediate, industry-specific reference ontologies. Rather than converging on a single, universal ontology, the field is producing many distinct ontologies, each with particular strengths and scope [7].

The Semantic Web is partially based on ontologies. It envisions an alternative or addition to the conventional web which will carry data using ontology-like structures. According to this vision, programs will be able to interoperate and share data freely, using ontologies to mediate data exchange. The goals for the Semantic Web are ambitious: it will “enable machines to comprehend semantic documents and data” [8]. While a full theoretical treatment of the Semantic Web has yet to be elucidated, the idea is that, if conceptual structures can be defined in sufficient detail and expressed in machine-readable metadata, then software applications will be able to share and use data intelligently and seamlessly.

The implicit assumption is that more detailed and rigorous definition of conceptual structures can create machine comprehension; the computer will “understand” what the data means sufficiently well to be able to process it appropriately. But how machine comprehension can arise from more detailed definition of concepts has yet to be explained. This appears to be a rather grand challenge similar to those previously addressed in artificial intelligence research.

Setting aside the claims about machine comprehension, can ontologies provide a basis for data sharing? In theory if two applications conform to the concepts defined

in a particular ontology then exchange of data becomes easy, because the applications are conceptually compatible. However, it requires that applications be designed from the ground up to conform to the particular ontology, which is an expensive proposition. Also, no standard ontology has emerged. Multiple competing ontologies and microformats exist, each with its own peculiar take on the relevant domains, and it is not particularly easy to use them in combination. If applications are built to match a variety of different ontologies, the risk is that this lack of standardization will perpetuate the present problem of ad hoc conceptual structures and legacy “information islands”.

An alternative approach is to use ontologies as a basis for mapping the conceptual models of applications to one another. Defining conceptual structures in more detail does not in itself convey significance or create machine comprehension. But richer definition of concepts (such as *customer* and *account*, in our example) might help link them to terms in a common ontology. This would allow programs to read the data, but it would not tell programs how to process the data appropriately—unless specific programming were to be done for each concept or term in the ontology. That is also an expensive proposition and probably less likely because ontologies contain many thousands of terms, most of which are irrelevant to any particular application.

To summarize, the development of ontologies may assist in reconciling mutually-incompatible conceptual structures and so allow applications to exchange data. But ontologies are in effect better “information plumbing”; they cannot tell applications how to process any given type of data. Each distinct ontology defines a set of concepts which must be programmed into a software application before the application can process the corresponding data meaningfully. The metadata (in RDF, for example) is useful because it signifies the type of data, but this is useful only inasmuch as the receiving application *already understands* how to deal with data of that type.

3.2 Potential for Basic-Level Categories to Facilitate Interoperability

Because of the fundamental limitation that applications must already share concepts in order to exchange corresponding data, the use of ontologies does not substantially improve our ability to link applications on a large scale or more rapidly. We are still held back by the need to program into each application the methods that it must use to handle data corresponding to each concept. There is no such thing as common sense or general knowledge for a computer application which would tell it how to handle specific types of data.

However, the discussion in Section 2.1 alluded to some results from neuroscience which may be helpful in thinking about this problem. The results suggest that a small number of categories are hardwired in the brain and perhaps innate. These correspond to concepts that people seem to understand intuitively: other people, food, tools, actions and so on. They are neither particularly generic nor particularly abstract but instead couched at a basic or everyday level [9]. One hypothesis is that these common basic-level concepts, together with other common experiences such as emotions and sensations, allow us to think and communicate about more complex ideas.

Applying the same idea to software applications, we could envisage a simple set of shared “innate” categories for software applications to adhere to, providing a basis for

at least some limited data exchange and interoperability, without the need for programming. It is generally a straightforward matter to map an existing conceptual model to a simple set of basic-level categories [10]. For example, data about people could be tagged as such, as could data about places, documents, organizations, and so on. Once an item of data had been tagged according to this set of simple basic-level categories, certain default programming would be applicable. For example, places can be represented as maps; documents can be downloaded and opened. Other possible basic-level categories include systems, physical objects, conceptual objects and categories [11].

Recall that the purchase order processing and accounting systems in our example contained the concepts *supplier* and *creditor*. If both of these concepts were identified with the innate category *organization*, then the two applications could exchange data meaningfully, identifying data in the categories *supplier* and *creditor* as data about organizations. The applications could then treat the data in a way deemed appropriate for processing data about organizations. Similarly, the concepts *purchase* and *transaction* could be identified with the innate category *activity*, allowing exchange and treatment appropriate for activities, and so on. By providing a generic level of programming to suit each shared basic-level category, at least some basic level of default operation would be possible on exchanged data without the two applications possessing complex shared conceptual schemas.

4 Conclusion

Today, each software application is structured around its own set of unique, ad hoc, concepts. This design practice guarantees conceptual incompatibility. There is a need to find alternative design approaches that will allow applications to work with different types of data more flexibly and share data more readily. Yet it is important to respect the uniqueness of each application's conceptual model; a one-size-fits-all conceptual model cannot easily be imposed.

It will become increasingly important for applications to be able to share data automatically, despite the problem of conceptual incompatibility. A way is needed of imbuing applications with the equivalent of common sense or general knowledge, which will allow them to offer a sensible response to new types of data. In ontology research, it is hoped that this will be achieved through more rigorous and more detailed definition of concepts, to ultimately enable a form of machine comprehension. But it is unclear how machine comprehension will be produced by more detailed definition of concepts. As for the less ambitious goal of using ontologies to map between conceptual models, even if ontology use in information systems were to become widespread, a critical mass of organizations would need to adopt a particular ontology before the benefits of standardization could be realized.

Overall, there is a tendency to think in rather non-specific ways about how ontologies and the Semantic Web might permit free exchange of data. Any exchange of data is constrained by the problem of conceptual incompatibility, and this cannot be overcome solely by the inclusion of more complex markup. It requires advance programming so that applications are able to handle the types of data to be exchanged. This cardinal rule constitutes a fundamental limit on conceptual interoperability and

can be stated thus: *applications can meaningfully interoperate with respect to data of a specific type only if they have been programmed in advance to handle data of that type*. When data containing particular concepts is exchanged, applications have to be specifically programmed to handle the relevant concepts, regardless of what mechanism is used to transfer the data or construct the programs, and irrespective of what markup or metadata is included.

In conclusion, this work remains theoretical. However, the relatively limited progress to date on the Semantic Web (in comparison with the worldwide web, for example) may represent evidence of the inherent limitation discussed in this paper. Research is needed into ways of genuinely allowing heterogeneous applications to exchange data in the face of conceptual incompatibility. Whether or not ontologies are involved, the idea of using a simple set of “innate” categories should be tested because it may offer a more practicable approach than attempting to implement large and unwieldy ontologies in software applications.

References

1. Liebenau, J., Backhouse, J.: *Understanding Information: An Introduction*. Macmillan, Basingstoke (1990)
2. Crestani, F.: Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review* 11, 453–482 (1997)
3. Sowa, J.F.: *Conceptual Structures*. Addison-Wesley, Reading (1984)
4. Modell, A.H.: *Imagination and the Meaningful Brain*. The MIT Press, Cambridge (2003)
5. Eysenck, M.W., Keane, M.: *Cognitive Psychology: A Student’s Handbook*. Psychology Press, UK (2005)
6. Mason, M.F., Banfield, J.F., Macrae, C.N.: Thinking About Actions: The Neural Substrates of Person Knowledge. *Cerebral Cortex* 14, 209–214 (2004)
7. Kalfoglou, Y., Hu, B.: *Issues with Evaluating and Using Publicly Available Ontologies* (2006)
8. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284, 28–37 (2001)
9. Pansky, A., Koriati, A.: The Basic-Level Convergence Effect in Memory Distortions. *Psychological Science* 15, 52–59 (2004)
10. McGinnes, S.: *Conceptual Modelling: A Psychological Perspective*. Ph.D Thesis, Department of Information Systems, London School of Economics, University of London (2000)
11. McGinnes, S., Amos, J.: Accelerated Business Concept Modeling: Combining User Interface Design with Object Modeling. In: Harmelen, M.V., Wilson, S. (eds.) *Object Modeling and User Interface Design: Designing Interactive Systems*, pp. 3–36. Addison-Wesley, Boston (2001)

A Goal-Oriented Approach for Optimizing Non-functional Requirements in Web Applications

José Alfonso Aguilar, Irene Garrigós, and Jose-Norberto Mazón

Lucentia-DLSI

University of Alicante, E-03080, San Vicente del Raspeig, Alicante, Spain
{ja.aguilar, igarrigos, jnmazon}@dlsi.ua.es

Abstract. Web design methodologies should be able to provide a requirements analysis stage to consider the large and heterogeneous audience of Web applications. In this stage, non-functional requirements (NFRs) should be addressed in order to improve the quality of the Web application perceived by users. To this aim, different configurations of requirements could be implemented depending on the NFRs preferred by the Web user. Furthermore, prioritizing and making tradeoffs between NFRs is crucial for satisfying the audience of Web applications. Therefore, this work presents an algorithm based on the Pareto optimal approach to evaluate and select the optimal configuration of requirements for a Web application. To do this, the NFRs are maximized according to a priority list provided by the audience. Our approach is illustrated with a running example.

Keywords: non-functional requirements, Web engineering, goal-oriented requirements engineering.

1 Introduction

Unlike traditional stand-alone software, the audience of Web applications is both open and large. Therefore, users may have different goals and preferences and stakeholders (in this context, stakeholders are individuals or organizations who affect or are affected directly or indirectly by the development project in a positive or negative form [9]) should be able to cope with these heterogeneous needs by means of an explicit requirements analysis stage in which functional and non-functional requirements are considered [2].

Functional requirements (FRs) describe the system services, behavior or functions, whereas non-functional requirements (NFRs), also known as quality requirements, specify a constraint in the application to build or in the development process [7]. An effective definition of requirements improves the quality of the final product. Unfortunately, in most of the Web engineering approaches, a complete analysis of requirements is performed considering only FRs, thus leaving aside the NFRs until the implementation stage. We totally agree with [3] the argument that NFRs are a very important issue and must be considered from

the very beginning of the development process, to be analyzed in depth, in order to improve the quality of the Web application perceived by users.

Interestingly, the recent inclusion of goal-oriented techniques in Web requirements engineering [4] offers a better analysis in the requirements stage since requirements are explicitly specified in goal-oriented models in order to support reasoning about organizational objectives, alternatives and implications, thus having a deep understanding about the domain. This has allowed the stakeholders to choose among the design decisions that can be taken to satisfy the goals and evaluate the implementation of certain requirements in particular (including NFRs). However, this is not enough to ensure that the Web application satisfies the real user needs because they do not offer mechanisms to maximize NFRs, i.e., to find a tradeoff between the FRs and NFRs.

Therefore, not paying attention to eliciting, documenting and tracking NFRs makes harder for the stakeholders to take design choices. Consequently, the quality of the Web application perceived by users will be affected negatively. Thus, NFRs need to be addressed to enable the stakeholder to choose among multiple configurations maximizing the NFRs, helping them to take design choices which positively affects the quality of the Web application, i.e., maximizing the NFRs navigability and accessibility, improves the browsing user experience.

Bearing these considerations in mind, this paper presents an algorithm that allows the stakeholder to evaluate the implementation of certain requirements considering the NFRs maximization. The algorithm is based on the Pareto optimal approach [10], which is useful when there are multiple competing and conflicting objectives that need to be balanced. The algorithm thus constructs a group of configurations (called Pareto front) optimizing the NFRs. A configuration only can be considered on the Pareto front, if and only if, it maximizes a NFR and the other ones remain the same, or they are maximized too. From these balanced configurations, the stakeholder can select the final configuration taking into account the different NFRs from the beginning of the development process. The proposal presented in this paper is defined upon our Web engineering method A-OOH (*Adaptive Object Oriented Hypermedia method*) [1] although it can be applied to any other Web engineering approach.

The remainder of this paper is structured as follows: Section 2, presents related work relevant to the context of this work. Section 3, describes the proposal for goal-oriented requirements analysis where is found the contribution of this work and introduces a running example for demonstration purposes. The Pareto algorithm for softgoals maximization and its application (described step by step) is presented in Section 4. Finally, the conclusion and future work is presented in Section 5.

2 Related Work

In our previous work [2], a systematic literature review has been conducted for studying requirement engineering techniques in the development of Web applications. Our findings showed that most of the Web engineering approaches focus on

the analysis and design phases and do not give a comprehensive support to the requirements phase. Furthermore, the NFRs are considered in a isolated form, leaving them out of the analysis stage. In addition, we can also conclude that the most used requirement analysis technique is UML use cases and profiles. On the other side, with regard to approaches that consider NFRs from early stages of the development process, in [8] the authors propose a metamodel for representing usability requirements for Web applications. Moreover, in [3] the authors present the state-of-the-art for NFRs in a MDD (Model-Driven Development), as well as an approach for a MDD process (outside the field of Web engineering). Unfortunately, these works overlook how to maximize the NFRs.

To sum up, there have been many attempts to provide techniques and methods to deal with some aspects of the requirements engineering process for Web applications. Nevertheless, there is still a need for solutions that considers NFRs from beginning of the Web application development process, in order to assure that they will be satisfied at the same time that the functional requirements are met, improving the quality of the Web application perceived by users.

3 Specifying Requirements in Web Engineering

This section briefly describes our proposal to specify requirements in the context of a Web modeling method by using i^* models [6], [1]. As a goal-oriented analysis technique, the i^* framework focuses on the description and evaluation of alternatives and their relationships to the organizational objectives. This proposal supports an automatic derivation of Web conceptual models from a requirements model by means of a set of transformation rules.

Following, we shortly describe an excerpt of the i^* framework which is relevant for the present work. For a further explanation, we refer the reader to [1]. The i^* framework consists of two models: the strategic dependency (SD) model to describe the dependency relationships (represented as \dashv) among various actors in an organizational context, and the strategic rationale (SR) model, used to describe actor interests and concerns and how they might be addressed. The SR model (represented as \odot) provides a detailed way of modeling internal intentional elements and relationships of each actor (\circ). Intentional elements are goals (\circ), tasks (\diamond), resources (\square) and softgoals (\heartsuit). Intentional relationships are means-end links (\dashv) representing alternative ways for fulfilling goals; task-decomposition links (\dashv) representing the necessary elements for a task to be performed; or contribution links ($\overset{\text{help}}{\dashv} \overset{\text{hurt}}{\dashv}$) in order to model how an intentional element contributes to the satisfaction or fulfillment of a softgoal. Possible labels for a contribution link are “Make”, “Some+”, “Help”, “Hurt”, “Some-”, “Break”, “Unknown”, indicating the (positive, negative or unknown) strength of the contribution.

To adapt i^* framework to the Web engineering domain we use the taxonomy of Web requirements presented in [5]. Next, we have used the extension mechanisms of UML to define a profile for using i^* to specific Web domain terminology. Therefore, new stereotypes have been added according to the different kind of Web requirements (NFRs are modeled directly using the i^* softgoal element).

A sample application of the i^* modeling framework for Web domain is shown in Figure 1, which represents the SR model of our running example for the Conference Management System (CMS), the complete specification of the case study can be found at: <http://users.dsic.upv.es/~west/iwwest01>. The purpose of the system is to support the process of submission, evaluation and selection of papers for a conference. It is important to highlight that each element from Figure 1 corresponds to a requirement type from the taxonomy previously mentioned, i.e., the content requirement (Content) from the taxonomy is displayed with the notation “*Resource*” from i^* and the navigational (Navigational) and service (Service) requirements with the symbol “*Task*” from i^* , both with their respective associations (*decomposition-links*). The extract of the CMS example is focused on the selection of the review process. Four actors participate in the CMS, but due to space limitations, for this example only the author, reviewer and system actors were considered. In this case, three actors are detected that depend on each other, namely “*Reviewer*”, “*Author*” and “*Conference Management System*”. The reviewer needs to use the CMS to “*Review paper*”. The author depends on the CMS in order to “*Paper be reviewed*”. These dependencies and the CMS actor are modeled by a SD and SR models in Figure 1.

The goal of the CMS actor is “*Process of review of papers be selected*”. To fulfill this goal, the SR model indicates that one of the two navigational requirements: “*Blind review process*” or “*Normal review process*” should be performed. In this running example, the path to achieve the goal of the CMS actor is by means of the navigational requirement “*Blind review process*”. We can observe in the SR model that some navigational and service requirements are decomposed in other requirements, some of them affects positively or negatively some NFRs (softgoals hereinafter), i.e., the service requirement “*Download paper without authors’ name*” needs the content requirement “*Papers*”, also, affects positively the softgoal “*Privacy be maximized*” and in some negatively form the softgoal “*Obtain more complete info*”. This fact is very important to see how to satisfy the goal “*Process of review of papers be selected*” considering the Web application softgoals.

4 Optimizing NFRs in Web Applications

In this section, we present a proposal (see Figure 2) which provides support for the stakeholder in order to evaluate and decide which functional requirements have to be implemented to improve the Web application functionality while NFRs are maximizing. To do this, we extend the goal-oriented requirement approach for Web engineering, described in Section 3, with the Pareto front algorithm, in order to achieve the optimal configuration based on the softgoals maximization. Also, a set of steps in order to fully satisfy the stakeholder goals have been defined. Therefore, the effect of the implementation of a set of FR can be assessed and the best among several implementation options can be selected by prioritizing the softgoals while still satisfying the goals.

The Pareto front algorithm is useful when there are multiple competing and conflicting objectives [10] that need to be balanced. The Pareto front is a notion

all decision vectors that are not dominated by any other decision vectors form the Pareto optimal set, while the corresponding objective vectors are said to form the Pareto front. Our approach as a running example is described next.

4.1 The Pareto Algorithm as a Running Example

Following the Pareto efficiency, we have defined the following steps to determine the Pareto optimal configuration for requirements implementation while the softgoals are balanced and maximized (see Figure 2). At the same time, these steps are applied in our running example presented in Section 3.

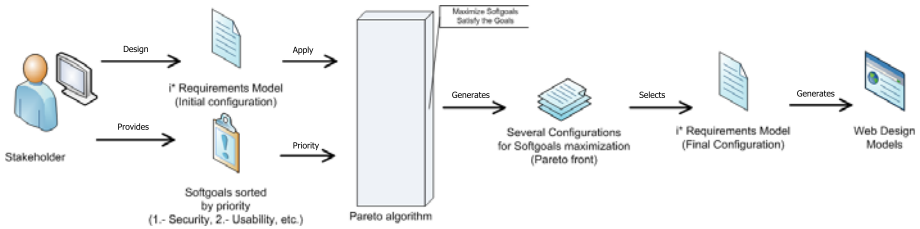


Fig. 2. Overview of the Pareto approach

Step 1. Create the Initial Requirements Model. The stakeholder creates an initial requirements model (IRM) using the i^* framework with which specifies the goals, softgoals and functional requirements (*Navigational, Service, Personalization, Layout and Content*) that the Web application must satisfy. This IRM specifies the requirements that must be implemented as a first prototype of the final Web application. At the same time, the stakeholder defines a list of softgoals sorted by priority, with which specifies the softgoals that the Web application must accomplish. For this running example, the requirements model is the one described in Section 3.

Step 2. Create a Requirements List. The second step consists of developing a list of the requirements (implemented or not) that contribute to any softgoal in the i^* requirements model (see Table 1). This table shows a requirements list and their type of contributions to the softgoals, where “S1” corresponds to softgoal “*Be fair in review*” from requirements model, “S2” to “*Review process easier*”, “S3” represents “*Accurate review process*”, “S4” conforms to “*Privacy be maximized*”, “S5” “*Avoid possible conflicts of interest*” and “S6” it is the “*Obtain more complete info*”.

Step 3. Store Each Possible Requirements Configuration. In this step, each possible implementation (configuration) of N requirements is stored in a decision vector $X_v: \forall v 0 \leq v < 2^N, \forall i \in \{1 \dots N\} X_{v_i} = T_i$, where X_{v_i} is the i th element of X_v , $T_i = I$ if the requirement is implemented and $T_i = N$ if the requirement is not implemented.

Table 1. The requirements contributions to softgoals

Requirements	"S1"	"S2"	"S3"	"S4"	"S5"	"S6"
R1.- "Blind review process"	Help	Break	Hurt	-	Help	-
R2.- "Download papers without authors' name"	-	-	-	Help	-	Some -
R3.- "Normal review process"	Some -	Make	Help	-	-	-
R4.- "Download paper with authors' name"	-	-	-	Some -	Hurt	Help
R5.- "View review process status"	-	-	-	-	-	Help

Step 4. Assign a Weight to Each Contribution from Requirements to Softgoals. The contribution of each requirement (implemented or not) must be quantified. To this aim, the stakeholder creates a matrix by using the following weights to each kind of contribution: $w=0$ if the requirement does not contribute to any softgoal, $w=+1$ if there is a Help contribution link, $w=-1$ if there is a Hurt contribution, $w=+2$ if there is a Some + link, $w=-2$ if the contribution is Some -, $w=+4$ if there is a Make and $w=-4$ if there is a Break contribution link.

Therefore, the matrix is defined, so that each entry W_{ij}^k corresponds to the contribution of the i th requirement to the j th softgoal on the k status (implemented or not): $\forall i \in \{1 \dots N\}, \forall j \in \{1 \dots M\}, \forall k \in \{I, N\} W_{ij}^k = w$, where N is the number of requirements and M is the number of softgoals, and w is defined as previously described.

For computing the objective functions in this running example, the following matrix (II) is defined containing the quantification of each requirement to softgoals, as explained in the previous section. As an example, row 3, (requirement "Normal review process"), column 2 (softgoal "Review process easier"), shows "+4" in the matrix, indicating a "Make" contribution if the requirement is implemented, and on the other side, if the requirement "Blind review process" (row 1) is implemented, column 2 will be indicating "-4" in the matrix, this means a "Break" contribution.

$$M_{ij}^k = \begin{pmatrix} +1 & -4 & -1 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 & 0 & -2 \\ -2 & +4 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -1 & +1 \\ 0 & 0 & 0 & 0 & 0 & +1 \end{pmatrix} \quad (1)$$

Step 5. The Objective Function. For each softgoal j the corresponding objective function F_j with respect to a decision vector X_v is calculated by summing the contributions of all requirements to each softgoal j taking into account the requirements configuration defined in x_v : $\forall j \in \{1 \dots M\}, \forall v_0 \preceq v < 2^N$ $F_j(X_v) = \sum_{j=1}^M W_{ij}^k$, where N is the number of requirements, M is the number of softgoals.

Finally, the sum of all objective functions with respect to a decision vector X_v is computed to obtain the overall fitness of the decision vector X_v : $\forall j \in \{1 \dots N\}, \forall v_0 \preceq v < 2^N$ $\sum_{j=1}^M F_j(X_v)$, where N is the number of requirements and M is the number of softgoals.

Table II shows all possible decision vectors (column 2 to 6, all rows), in other words, all possible requirements configurations, where "I" represents the status "Implemented" and "N" represents "Not implemented". The results of the corresponding objective functions are shown in columns 7 to 12, and the overall

Table 2. The possible requirements to implement or not for the softgoal tradeoff

Configuration	R1	R2	R3	R4	R5	F(S1)	F(S2)	F(S3)	F(S4)	F(S5)	F(S6)	Pareto front
X1	I	I	I	I	I	-1	0	0	-1	0	0	No
X2	I	I	I	I	N	-1	0	0	-1	0	-1	No
X3	I	I	I	N	I	-1	0	0	1	1	-1	Yes
X4	I	I	I	N	N	-1	0	0	1	1	-2	No
X5	I	I	N	I	I	1	-4	-1	-1	0	0	Yes
X6	I	I	N	I	N	1	-4	-1	-1	0	-1	No
X7	I	I	N	N	I	1	-4	-1	1	1	-1	Yes
X8	I	I	N	N	N	1	-4	-1	1	1	-2	No
X9	I	N	I	I	I	-1	0	0	-2	0	2	Yes
X10	I	N	I	I	N	-1	0	0	-2	0	1	No
X11	I	N	I	N	I	-1	0	0	0	1	1	Yes
X12	I	N	I	N	N	-1	0	0	0	1	0	No
X13	I	N	N	I	I	1	-4	-1	-2	0	2	Yes
X14	I	N	N	I	N	1	-4	-1	-2	0	1	No
X15	I	N	N	N	I	1	-4	-1	0	1	1	Yes
X16	I	N	N	N	N	1	-4	-1	0	1	0	No
X17	N	I	I	I	I	-2	4	1	-1	-1	0	Yes
X18	N	I	I	I	N	-2	4	1	-1	-1	-1	No
X19	N	I	I	N	I	-2	4	1	1	0	-1	Yes
X20	N	I	I	N	N	0	0	1	1	0	-2	No
X21	N	I	N	I	I	0	0	0	-1	-1	-1	No
X22	N	I	N	I	N	0	0	0	-1	-1	-1	No
X23	N	I	N	N	I	0	0	0	1	0	-1	Yes
X24	N	I	N	N	N	0	0	0	1	0	-2	No
X25	N	N	I	I	I	-2	4	1	-2	-1	2	Yes
X26	N	N	I	I	N	-2	4	1	-2	-1	1	No
X27	N	N	I	N	I	-2	4	1	0	0	1	Yes
X28	N	N	I	N	N	-2	4	1	0	0	0	No
X29	N	N	N	I	I	0	0	0	-2	-1	2	Yes
X30	N	N	N	I	N	0	0	0	-2	-1	1	No
X31	N	N	N	N	I	0	0	0	0	0	1	Yes
X32	N	N	N	N	N	0	0	0	0	0	0	No

fitness for each decision vector is shown in column 13. Finally, in the last column, we indicate if the corresponding decision vector is in the Pareto front. Grey rows are the Pareto front.

Step 6. Maximize the Softgoals and Still Satisfying the Goals. In this step the stakeholder creates a list of softgoals sorted by priority (the softgoals priority was established in the list from *Step 1* by the stakeholder) and a list of goals that the Web application has to achieve. For this case, the softgoals priority list is shown in Table 3.

Table 3. Softgoals priority list for achieve the goal “Process of review of papers be selected”

Order	Softgoal
1	“S4.- Privacy be maximized”
2	“S2.- Review process easier”
3	“S3.- Accurate review process”
4	“S1.- Be fair in review”
5	“S5.- Avoid possible conflicts os interest”
6	“S6.- Obtain more complete info”

Step 7. Select the Pareto Optimal Configuration. Finally, according to Table 3, the two most important softgoals to maximize are “S4” and “S2”. Therefore, it is necessary to select the final solution according to the priorities established over the softgoals.

First of all, it is necessary to select the configurations that besides being Pareto front satisfy the goal “Process of review of papers be selected”, for this running example, the configurations “X3”, “X7”, “X17” and “X25” are the only ones that satisfy the goal from all the 14 configurations that are Pareto front (see Table 2). Importantly, this step could be done in *Step 5*, i.e., calculating the

objective function for those configurations that satisfy the goals, but not doing it in this form allows us to select between different configurations considering only the softgoals maximization, leaving aside the *goals*, this gives a wider scope to the stakeholder for the final implementation.

The next step consists in selecting from the configurations that are Pareto front and satisfy the goal, the ones that maximize the softgoals according with the list from Table 3. To do this, it is necessary to check all the configurations with the requirements model to select the configurations that allow to achieve the goal (in this case there are two paths, i.e., two means-ends links), these are “X3”, “X7”, “X17” and “X25”. Then, it is necessary to select the best option according to the softgoals to maximize. For the softgoal “S4”, “X3” and “X7” are the configurations which its overall is maximized and, for the softgoal “S2” are “X17” and “X25”.

For this running example, the configuration “X3” is the best option, because according with the priority list, “S4” and “S2” are the softgoals to prioritize. The configurations “X17” and “X25” maximize “S2”, however the contributions of both to softgoal “S4” (which is the number one from the priority list) are -1 and -2 (see Table 2). Furthermore, besides that the configuration “X3” has an overall fitness of $+1$ for “S4” as same as the configuration “X7”, the configuration “X3” has an overall fitness of 0 for “S2” and, “X7” has an overall fitness of -4 for “S2”, resulting more affected that the configuration “X3” (see Table 2), with which indicating that optimizing security comes at a high cost with respect to other softgoals (usability). The rest of solutions of the Pareto front are intermediate configurations that lead us to different tradeoffs.

Finally, the final requirements model (FRM) is the configuration “X3” (see Table 2). Therefore, the requirements “R1.- Blind review process”, “R2.- Download papers without authors name”, “R3.- Normal review process” and “R5.- View review process status” must be implemented in order to maximize the softgoals “S4.- Privacy be maximized” and “S2.- Review process easier”. In “X3” only “R4” is not implemented. These requirements enable alternative paths (means-ends links) to satisfy the goal.

5 Conclusion and Future Work

In this work, we have presented an extension to our goal-oriented requirements analysis approach for the development of Web applications. Our approach allows the stakeholder to evaluate and decide which requirements configuration to implement. To do so, we devised an algorithm based on the Pareto optimal approach that is particularly suited to balance and maximize the conflicting softgoals. Furthermore, it facilitates the evaluation of the obtained (Pareto) optimal solutions and the selection of the final solution taking into account the priorities of softgoals. To do this, it includes weighted contributions according to the importance of softgoals, in order to further help the stakeholder to balance and optimize the different softgoals. Future work consists in the integration of

our goal-oriented approach for requirements analysis and the Pareto algorithm in a MDD solution for the development of Web applications, within the A-OOH approach.

Acknowledgments. This work has been partially supported by the MANTRA project (GV/2011/035) from the University of Alicante, and by the MESOLAP (TIN2010-14860) from the Spanish Ministry of Education and Science. José Alfonso Aguilar is subventioned by CONACYT (Consejo Nacional de Ciencia y Tecnología) Mexico and University of Sinaloa, Mexico.

References

1. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: An MDA Approach for Goal-Oriented Requirement Analysis in Web Engineering. *J. Univ. Comp. Sc.* 16(17), 2475–2494 (2010)
2. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: Web Engineering Approaches for Requirement Analysis- A Systematic Literature Review. In: 6th Web Information Systems and Technologies (WEBIST), vol. 2, pp. 187–190. SciTePress Digital Library, Valencia (2010)
3. Ameller, D., Gutiérrez, F., Cabot, J.: Dealing with Non-Functional Requirements in Model-Driven Development. In: 18th IEEE International Requirements Engineering Conference (RE), pp. 189–198. IEEE, Los Alamitos (2010)
4. Bolchini, D., Paolini, P.: Goal-Driven Requirements Analysis for Hypermedia-Intensive Web Applications. *J. Req. Eng.* 9(2), 85–103 (2004)
5. Escalona, M.J., Koch, N.: Requirements Engineering for Web Applications - A Comparative Study. *J. Web Eng.* 2(3), 193–212 (2004)
6. Garrigós, I., Mazón, J.N., Trujillo, J.: A requirement analysis approach for using i* in web engineering. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 151–165. Springer, Heidelberg (2009)
7. Gupta, C., Singh, Y., Chauhan, D.S.: Dependency Based Process Model for Impact Analysis: A Requirement Engineering Perspective. *J. Comp. App.* 6(6), 28–30 (2010)
8. Molina, F., Toval, A.: Integrating Usability Requirements that can be Evaluated in Design Time into Model-Driven Engineering of Web Information Systems. *J. Adv. Eng. Softw.* 40, 1306–1317 (2009)
9. Sommerville, I.: *Software Engineering*, 6th edn. Addison-Wesley, Reading (2001)
10. Szidarovszky, F., Gershon, M., Duckstein, L.: *Techniques for Multiobjective Decision Making in Systems Management*. Elsevier, Amsterdam (1986)
11. Yu, E.S.K.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE International Symposium on Requirements Engineering (RE), p. 226. IEEE, Washington, DC, USA (1997)

Yet Another BPEL Extension for User Interactions^{*}

Mohamed Boukhebouze, Waldemar Pires Ferreira Neto, and Lim Erbin

PReCISE Research Center, University of Namur, 5000, Belgium
{mohamed.boukhebouze,waldemar.neto,lim.erbin}@fundp.ac.be

Abstract. In this paper, we propose a BPEL extension that deals with the user interactions expression in a Web service composition. This extension defines new data interaction activities to allow user providing, selecting or getting data. Moreover, the extension proposes a new type of interaction events that allow processing of the user interaction. The user interaction specification helps to generate a user interface for the Web service composition which is made by transforming the described user interactions to user interface components.

Keywords: BPEL extension, User Interaction, User Interface.

1 Introduction

A Web service is an autonomous software application that can be described, published, discovered, and invoked across the Web [6] by using a set of XML based standards such as UDDI, WSDL, and SOAP. Web services can be combined together in order to fulfill the user request that a single Web service cannot satisfy [10]. This mechanism is known as Web service composition. A Web service composition consists of several Web services orchestration or Web services choreography processes that deal with a functional need which is unsatisfied by a single Web service [10].

Several initiatives have been conducted to provide languages such as WS-BPEL (Web Services Business Process Execution Language) [5] that allow the description of Web service composition execution. WS-BPEL (BPEL for short) is an XML based standard that provides a syntax to define the Web service composition behavior (control flow) and mechanisms that enable data manipulation (data flow).

This language expresses the Web service composition process in a fully automated way. Users are not able to interact with the Web services until the end of the process execution. For example, users are not able to provide the input to a Web service at runtime, they are not able to cancel the process execution, or they are not able to have some intermediary output from a Web service. However, many Web service composition scenarios require user interactions [3]. These user interactions can be classified into four types [8]:

^{*} Research supported by la Wallonie.

- *Data input interaction* represents the fact that the user provides data to a Web service at runtime. For example, a user provides a licence Number to a car renting Web service;
- *Data output interaction* represents the fact that a Web service composition makes data available to the user. For example, the trip scheduling Web service composition presents the car renting price to the user;
- *Data selection* represents the fact that the user can select a data from a set of data. For example, a user can make a choice between flight or driving as a means of transportation;
- *Interaction event* represents an indicator that a user interaction has been carried out. For example, a cancellation is done by a user during the Web service composition execution (by pressing a button for example).

The difference between the data interaction types (input, output and selection) and the interaction event is that the data interaction types allow changing the data flow of the composition, while an interaction event changes only the control flow of the composition regardless of the data (e.g. the cancelling process is done regardless of the data).

Unfortunately, the BPEL language does not support such types of user interaction. For this reason, BPEL meta-model needs to be extended to express user interactions in a Web service composition. In addition, a user interface for the composition needs to be developed in order to help the user to interact with the Web services at runtime.

In this work, we propose a BPEL extension for the user interactions expression. This extension is called UI-BPEL (User Interaction Business Process Execution Language). UI-BPEL supports the expression of the four types of the user interactions explained above by introducing new BPEL elements: (1) a new BPEL activities (*DataInputUI*, *DataOutputUI*, *DataSelectionUI*) to express the user data interactions; (2) a new type of BPEL event (*InteractionEventUI*) to express the interaction event; (3) an extension of the BPEL's *Pick* and *Scope* activities that support the new *InteractionEventUI*. The main objective of this extension is to allow the generation of a user interface for the Web service composition based on the described user interactions. This generation is performed by transforming the UI-BPEL user interactions elements to specific user interface components. For example, transforming a *DataInputUI* to a text box, transforming a *DataOutputUI* to a label, and transforming a *DataSelectionUI* to a combo box.

UI-BPEL is part of the several initiative efforts of BPEL extension to address the user interaction expression in a Web service composition. An example of such extensions is BPEL4People [3], which introduces user actors into a Web service composition by defining a new type of BPEL activities to specify user tasks. However, this extension focuses only on the user task and does not deal with the design of a user interface for the Web service composition. Another example of BPEL extensions that addresses the user interaction is BPEL4UI (Business Process Execution Language for User Interface) [1]. BPEL4UI extends the *Partner Link* part of BPEL in order to allow the definition of a binding

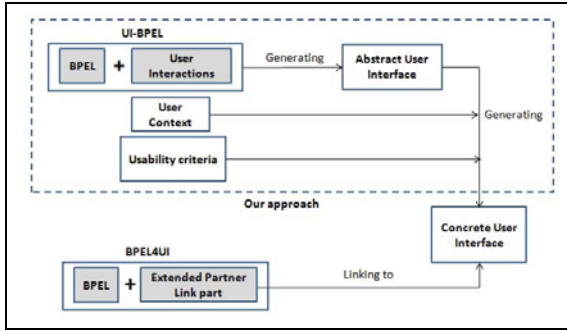


Fig. 1. UI-BPEL Vs. BPEL4UI

between BPEL activities and an existing user interface. This user interface is developed separately from the composition instead of being generated.

Figure 1 shows the difference between our approach (UI-BPEL) and the BPEL4UI approach. The top side of the figure depicts our approach that proposes to extend BPEL with user interactions so that a user interface for the Web service composition can be generated. The generated user interface is described in two abstraction levels: first, we propose to generate, from a UI-BPEL process, an abstract user interface, which is independent of any interaction modality (e.g. graphical input, vocal output) and computing platform (e.g. PC, smart phone) [4]. We then generate a concrete user interface based on the user context and by taking into account a set of user interface usability criteria [7]. This approach is compliant with the existing user interface description languages (like UsiXML [4]), which describe a user interface at different abstraction levels. The bottom side of Figure 1 shows the BPEL4UI approach that proposes to extend the *Partner Link* of BPEL in order to link the composition with an existing concrete user interface (HTML/JavaScript). This approach does not allow the generation of a user interface adapted to the user context (user preference, user environment, and user platform) and the usability criteria (e.g. the interface should respect the size of the device screen).

In the remainder of this paper, we focus on the description of the UI-BPEL. In section 2, we show a Web service composition scenario that requires user interactions. We present, in Section 3, an overview of the UI-BPEL meta-model with an illustrative example. Next, in Section 4, we present an extension of the Eclipse BPEL editor that supports the UI-BPEL. Finally, we conclude this paper with a discussion about the generation of the user interface from the proposed extension and our future works.

2 Scenario

In this section we introduce the scenario of the purchase order process, which requires user interactions. A customer requests a purchase order by providing

the necessary data such as the item, and the customer address (user interaction: data input interaction). Upon receiving a request from the customer, the initial price of the order, initial price of the order is calculated and a shipper is selected simultaneously. The shipper selection needs to be made by the customer who selects a shipper from a list of available shippers (user interaction: data selection). When the two activities are completed, the final price is calculated and a purchase order needs to be presented to the customer (user interaction: data output interaction). If the customer accepts his purchase order, she/he selects a payment method either by cash or by credit card (user interaction: data selection). There is a discount if the customer pays by cash. Next, a bill is shown to the customer (user interaction: data output interaction). Finally, the user the customer could cancel her/his request for an purchase order before receiving the receipt of the payment (user interaction: interaction event).

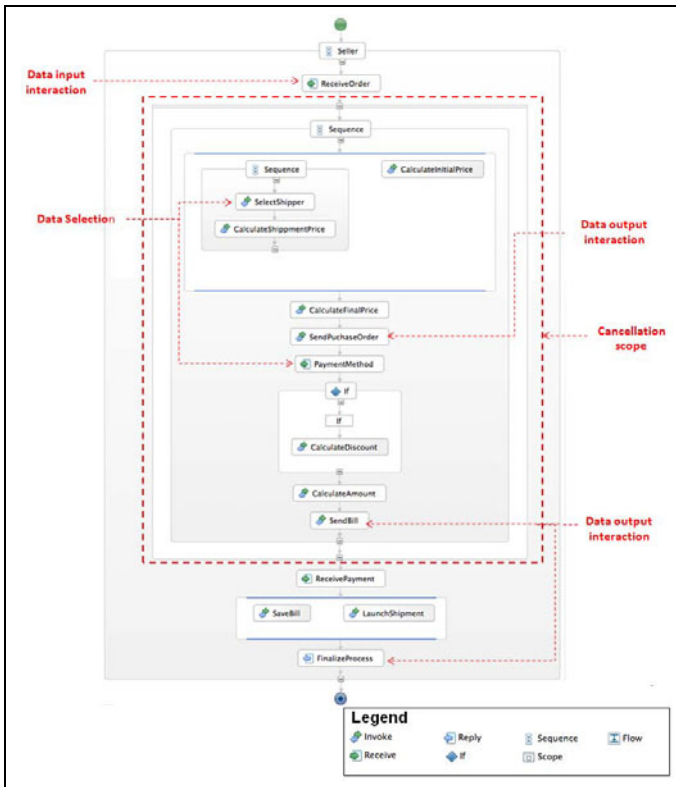


Fig. 2. Purchase Order Process expressed in BPEL

Figure 2 illustrates the purchase order process expressed using the graphical representation of Eclipse BPEL Designer graphical notations [2]. BPEL does not support any type of user interaction required by the scenario. For example, BPEL cannot express the fact that the customer needs data interaction to provide

the order. Moreover, BPEL cannot express the fact that the process can make the purchase order available to the customer using data output interaction. In addition, BPEL cannot describe the data interaction that allows the customer to choose a shipper and a payment method. Finally, BPEL does not support the fact that the customer has the ability to cancel the process before the payment is received.

In the next section, we propose an extension of BPEL that deals with user interactions required by the purchase order process scenario. The user interaction specification helps to generate a user interface for the purchase order process.

3 UI-BPEL

In this section, we present our BPEL extension (called UI-BPEL) that addresses the BPEL user interaction expression issue.

3.1 UI-BPEL Meta-Model

UI-BPEL extends the BPEL [5] by adding new data interaction activities and a new type of interaction events. Figure 3 presents an overview of the UI-BPEL meta-model. It shows the classes that model the user interactions and their connections with some relevant classes of the BPEL meta-model.

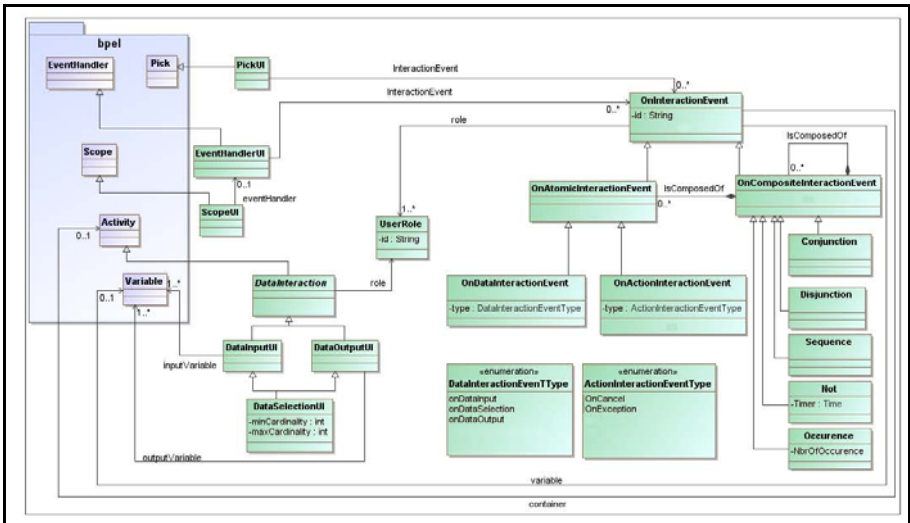


Fig. 3. Overview of the UI-BPEL Meta-model

In the following, we present the main classes of the UI-BPEL meta-model.

- **DataInteraction** is a generalization class that represents the new UI-BPEL activities.

- ***DataInputUI*** is a class that represents the data input activity. This activity is similar to the BPEL *Receive* activity, since it suspends the composition process execution while waiting for an input from the user. However, unlike the BPEL *Receive* activity, the process waits for an event where data is provided, instead of a message from another Web service. This type of event will be explained afterwards.
- ***DataOutputUI*** is a class that represents the data output activity. This activity specifies which variable contains data to be presented to the user.
- ***DataSelectionUI*** is a class that represents the data selection activity. This activity allows the user to select one value (or a subset of values) from a set of values of a specific variable. The number of selectable values can be defined by using the *minCardinality* property (how many values must be selected at least) and/or the *maxCardinality* property (the maximum number of elements have to be selected). Like ***DataInputUI***, the ***DataSelectionUI*** activity suspends the execution of the process until receiving an event of selecting data as we will explain afterwards.

UI-BPEL proposes a new type of event to process the user interaction. These events help to indicate when a user interaction is carried out, so that a specific action can be launched. UI-BPEL defines the user interaction events as following:

- ***OnInteractionEvent*** is a generalization class that represents a new type of BPEL event. Such an event defines the occurrence of a user interaction. An interaction event can be either an atomic event that designates a predefined event *OnAtomicInteractionEvent* or a composite event that combines atomic and/or composite events *OncompositeInteractionEvent*.
- ***OnAtomicInteractionEvent*** is a generalization class that represents the occurrence of a data interaction or an interaction action done by user.
- ***OnDataInteractionEvent*** is a class that indicates the fact that a Data interaction has been done. An *OnDataInteractionEvent* can be:
 - *onDataInput* indicates that new data has been entered by the user. This event allows the process to resume its execution when it was being suspended by a *DataInputUI* activity.
 - *onDataSelection* indicates that some data has been selected by the user. This event allows to resume the process execution when it was being blocked by a *DataSelectionUI* activity.
 - *onDataOutput* indicates that new data has been presented through the user interface. This event is used to notify that the user validates a data output. For example, an *onDataOutput* event can be the confirmation that the user accepts the copyright licence of a Web service.
- ***OnActionInteractionEvent*** is a class that represents the notification of the occurrence of a predefined interaction action. These predefined interaction actions are:
 - *OnCancel* indicates that a cancellation action has been performed by the user.

- *OnException* indicates that an exception has occurred in the user interface (e.g. the User Interface Failure). This event is used by the BPEL *Fault handler* part.

OnActionInteractionEvent is used to handle the cancellation or the exception that can happen within a scope of the Web service composition.

- ***OnCompositeInteractionEvent***: is a class that expresses a combination of atomic and/or composite events. According to [9], a composite event can be:
 - *Disjunction*($e1, e2$): specifies that at least one of the two interaction events is detected;
 - *Conjunction* ($e1, e2$): specifies that interaction events take place without taking into account of their occurrence order;
 - *Occurrence* ($e1, \text{number of occurrence}$): specifies multiple occurrences of the same interaction event;
 - *Sequence* ($e1, e2$): specifies the sequence of interaction events
 - *Not* ($e1, t$): characterizes the fact that an interaction event has not happened at time t .

OnCompositeInteractionEvent can be used to process the user interactions aspect in the Web service composition. For example, if a user does not provide an input data during time t (*Not* (*onDataInput*, t)), the process should be canceled. *OnCompositeInteractionEvent* can also be used to process the grouping of a set of data interactions on the same User interface container. For example, if we group the data input of the customer information, and the data selection of a shipper on a same user interface component, the event *Conjunction* (*on customer information Input*, *on Shipper Selection*) models the fact that the customer information is provided and a shipper Selected. This could resume the execution of a suspended process by the data input and the data selection activities. Note that, grouping the data interaction on a same user interface component is handled by the user interface generation method. The data interaction grouping is not in the scope of this paper.

UI-BPEL also defines some useful new classes, for example:

- ***UserRole***: is a class that represents which user is responsible of *DataInteraction*. *UserRole* is an important element for the user interface generation process since a user interface should be generated for each different role.
- ***PickUI***: is a class that extends the BPEL *Pick* activity in order to take into account *onInteractionEvent* listening.
- ***ScopeUI***: is a class that extends the BPEL *Scope* activity in order to add on its *EventHandler* the *onUserEvent* listening.

3.2 UI-BPEL Example

We now illustrate the UI-BPEL of the purchase order scenario presented in Section 2.

4 UI-BPEL Designer Tool

This section describes the UI-BPEL Designer Tool that is dedicated to edit a UI-BPEL process that conforms to the UI-BPEL Meta-model. The tool is an Eclipse plug-in based on the Eclipse BPEL Designer [2]. The Eclipse BPEL Designer is chosen since it has an extensible framework that allows not only the WS-BPEL 2.0 specification, but also the deployment and the execution of BPEL processes from the tool into a BPEL engine. Figure 5 shows a screenshot of the UI-BPEL Designer Tool.

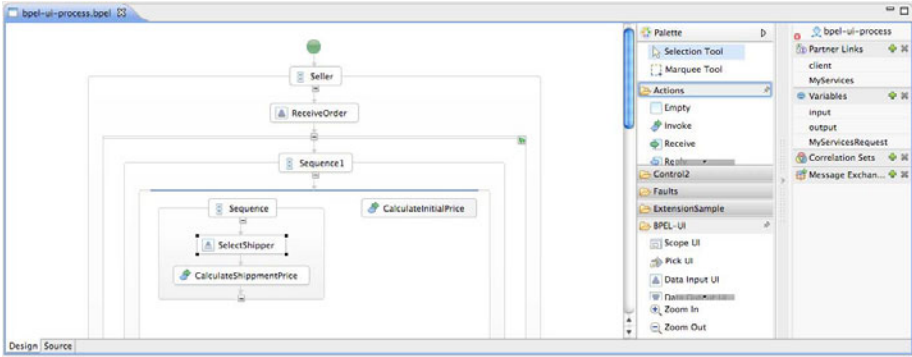


Fig. 5. UI-BPEL Designer Tool

The tool is available at the following address: <http://webapps.fundp.ac.be/wse>

5 Conclusion and Discussions

In this paper, we presented a BPEL extension, called the UI-BPEL. This extension addresses the user interaction in a Web service composition by defining new data interaction activities to allow user providing, selecting or getting data. The extension also proposes new types of interaction events to react when user interaction is carried out. The main goal of UI-BPEL is to allow the generation of a user interface for the Web service composition based on the described user interactions. This generation can be done in two steps:

- **Step1:** an abstract user interface [7] is generated from a UI-BPEL model. This user interface is described independent of any interaction modality (e.g. graphical modal, vocal modal) and computing platform (e.g. PC, smart phone). For this reason, each specified user interaction will be transformed into an abstract component, which can be an input abstract component, a selection abstract component, an output abstract component or a control abstract component. A control abstract component allows to process one or a set of data interactions. It also allows to model action interaction (e.g. cancel action). In addition, the control abstract component is also responsible of triggering the interaction events.

- **Step2:** a concrete user interface [7] is generated from the abstract user interface. The concrete user interface is adapted to a specific user context (user preference, user environment, and user platform). For example, for a visually handicapped person, the output abstract component will be transformed to vocal output. The concrete user interface takes also into account a set of user interface usability criterions. For example, the interface should respect the size of the device screen.

Our future work includes the development of two transformation methods for the two steps of the user interface generation described above.

References

1. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From people to services to ui: Distributed orchestration of user interfaces. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 310–326. Springer, Heidelberg (2010)
2. Eclipse, B.: Project. Eclipse BPEL Designer (2011), <http://www.eclipse.org/bpel/>
3. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., Trickovic, I.: Ws-bpel extension for people-bpel4people. Joint White Paper, IBM and SAP (2005)
4. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A language supporting multi-path development of user interfaces. In: Bastide, R., Palanque, P.A., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
5. OASIS, B.: Web Services Business Process Execution Language (2007)
6. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)
7. Seffah, A., Gulliksen, J., Desmarais, C.M. (eds.): Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle. HCI Series, vol. 8 ch. 6, pp. 109–140. Springer, Heidelberg (2005)
8. Tofan, S., Pradais, A., Buraga, S.: A study regarding the abstract specification of the user interface by using USIXML and UIML languages. Romanian Journal of Human-Computer Interaction (RoCHI 2009), 31–34 (2009)
9. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD 2006, pp. 407–418 (2006)
10. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Trans. Software Eng. 30(5), 311–327 (2004)

Semantics-Enabled Web API Organization and Recommendation

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dipartimento di Ingegneria dell'Informazione
Universita' degli Studi di Brescia, Via Branze, 38, 25123 Brescia
{bianchin,deantone,melchior}@ing.unibs.it

Abstract. The design of Web applications from third-party Web APIs can be shortened by providing a conceptual model that abstracts from implementation aspects of each Web API and supports their selection. In this paper we propose a framework to support easy Web application development. The framework provides Web API organization based on automated matching techniques apt to establish semantic links between them according to properly defined similarity and coupling criteria and Web API selection patterns to support interactive and proactive Web application development according to an exploratory perspective.

1 Introduction

Recently, research attention is being progressively shifting from the development of Web applications from scratch to their composition starting from a huge amount of components independently developed by third parties [7]. These components are made available through their APIs, to serve the development of situational applications. However, integration of Web APIs often means integration of UIs (consider, for instance, the wide use of Google maps in ProgrammableWeb.com, an on-line registry of about 3000 Web APIs). UIs can generate/react to events, that require synchronization with the other Web APIs in the same application. Moreover, Web application development is hampered by the semantic heterogeneity of Web API descriptions (in terms of I/O variables, operations, events) and by their increasing number.

The development of Web applications from third-party Web APIs can be shortened by providing a model that abstracts from implementation aspects of each Web API and supports their selection. While some efforts such as WADL (Web Application Description Language) are being developed for RESTful services, whose aim is to be the counterpart of the WSDL standard for SOAP Web services, there are some tools such as SWEET [9] which guides the providers to give a structured representation of their Web APIs (by using the hRESTS formalism) and add semantics by referencing concepts in publicly available domain ontologies through the MicroWSMO language [8]. This still does not avoid problems during Web application development due to the huge availability of Web APIs. In this paper we propose a framework to support easy Web application development. The framework provides Web API organization based on automated

matching techniques apt to establish semantic links between them according to properly defined similarity and coupling criteria and Web API selection patterns to support interactive and proactive Web application development according to an exploratory perspective.

The relevance of models of components/Web APIs which abstract from implementation details and thus make the composition of the Web application easier has been highlighted in [1,4,5]. In [1] a formal model based on Datalog rules is proposed to capture all the aspects of a mashup component. Mashups are combined into patterns and a notion of inheritance relationship between components is introduced. This model has been used in the MATCHUP system [6], which suggests components to be connected on the basis of recurrent patterns. In [4] an abstract component model and a composition model are proposed, expressed by means of an XML-based language, where components are coupled according to a publish/subscribe mechanism, where events raised from a component are associated to operations of other components. No particular selection strategies are suggested. In [5] a faceted classification of unstructured Web APIs and a ranking algorithm to improve their retrieval are proposed. The classification and searching solution are based on IR techniques. The use of semantics in Web application development has been introduced in [10], where authors propose a novel application of semantic annotation together with a matching algorithm for finding sets of functionally equivalent components. With respect to these approaches, our approach considers different kinds of semantic links between Web APIs, providing a richer organization of available APIs and easy browsing of them during Web application development. Based on this organization, finer selection strategies have been designed and implemented.

The paper is organized as follows: Section 2 introduces a motivating scenario and the proposed framework, whose elements are detailed in Section 3 (Web API semantic descriptor), Section 4 (semantic links) and Section 5, where the selection patterns are presented; Section 6 presents a preliminary implementation and validation of the framework; finally, Section 7 closes the paper and points out future work.

2 Motivating Scenario

Let consider Dante, a Web designer who works for an Italian multimedia and entertainment publishing company and must quickly design a Web application to allow users to find the company shops for promoting coming soon books/DVDs. This is an example of situational application, targeted on the company's specific requirements and potentially useful for short periods of time, that is, until the sales promotion will last. Main tasks of Dante concern the selection and combination of suitable Web APIs which already implement some of the required functionalities, such as Web APIs to obtain information about company shops or Web APIs to visualize on a map the location of the shops which have been found. Dante proceeds step by step by selecting Web APIs and wiring them to obtain the final Web application as quickly as possible. Dante specifies the

Web API he's looking for, for instance in terms of desired categories, operations and inputs/outputs. A list of available API descriptions should be proposed, ranked with respect to the degree of match with categories, operations and I/O names specified by Dante. Less relevant APIs must be filtered out, to properly reduce the search space. When Dante has already selected a Web API, other APIs that could be coupled with the selected one should be proactively suggested to him. The suggested APIs should be ranked with respect to the degree of coupling with the selected one. Coupling can be evaluated on the basis of correspondences between events and operations of Web API descriptions. Let suppose that Dante chooses a Web API that, given the book/DVD title, returns the addresses of company shops where the book/DVD is available. Other Web APIs that can be wired with the selected one, for example Web APIs that visualize points on a map by specifying the addresses to display, can be suggested to Dante. If Dante wants to substitute one of the selected Web APIs, the system should suggest a list of candidate APIs as alternatives, ranked with respect to their similarity with the API to be substituted. API similarity can be evaluated on the basis of similarity between operation names, inputs and outputs of Web API descriptions.

The framework we propose has been designed to support Dante in the described motivating scenario. The framework is based on semantic annotation of Web APIs provided through available tools (e.g., SWEET [9]) and is composed of three main elements.

- *Semantics-enabled Web API model.* A collection of Web API semantic descriptors are extracted from semantically Web APIs. Descriptors abstract from underlying concrete implementations of the Web APIs. Each semantic descriptor has a reference to the URI of the original API.
- *Web API registry model.* Web API semantic descriptors are organized in a semantics-enabled registry through semantic links established by applying automated matching techniques.
- *Selection patterns.* Semantic links are exploited for supporting: (i) proactive suggestion of Web API descriptors ranked with respect to their similarity with the Web application designer's requirements; (ii) interactive support to the designer for the composition of the Web application, according to an exploratory perspective.

3 Semantics-Enabled Web API Model

Basic elements of the semantics-enabled Web API model are shown in Figure 1. The description of Web APIs as found on the Web typically relies on non structured HTML documentation that makes difficult machine processing. Therefore techniques, formats and tools for semantic annotation of Web APIs based on lightweight semantic models have been proposed in [9], where the SWEET tool have been described. The SWEET tool provides support in the identification of elements that characterize a Web API to produce a description according to the hREST language. A set of common elements can be identified in Web API

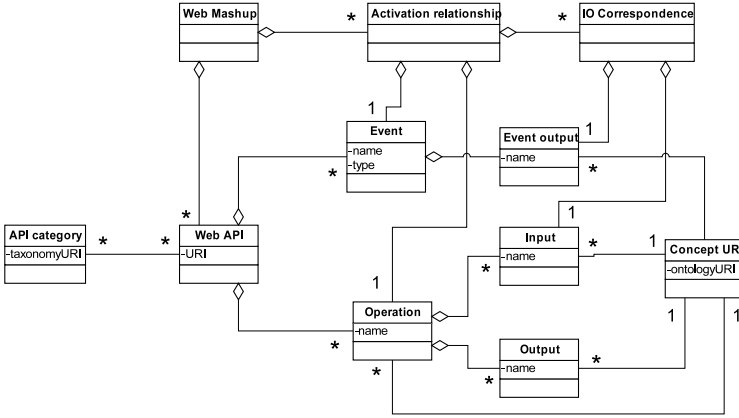


Fig. 1. The semantics-enabled Web API model

descriptions: (i) inputs and outputs; (ii) operations, usually associated with buttons or links on the Web API graphical interface; (iii) events, to model user's interactions with the Web API interface. Therefore, SWEET enables the semantic annotation of the hRESTS structure: (i) APIs are classified with respect to categories, that are taken from standard taxonomies available on the Web (identified by a taxonomy URI); (ii) operation names, inputs and outputs and event outputs are annotated with concepts, extracted from domain ontologies and identified by a concept URI. Domain ontologies are built by domain experts and can be designed ad-hoc for a particular application or can be made available on the Web for general purposes. If a suitable ontology is not available, it must be created first, for example using common editors (such as Protégé OWL). No commitment is made on a particular ontology formalism. Annotation and classification of the Web APIs is performed according to the MicroWSMO [8] notation extended with semantically annotated events.

From each semantically annotated Web API, the following semantic descriptor SD_i is extracted:

$$SD_i = \langle CAT_i, OP_i, EV_i \rangle \quad (1)$$

where: CAT_i is a set of categories, OP_i is a set of operations, EV_i is a set of events. Each operation $op_k \in OP_i$ is described by the operation name op_k , the operation inputs $IN(op_k)$ and the operation outputs $OUT(op_k)$, that are annotated with concepts taken from the domain ontologies. Each event $ev_h \in EV_i$ is described by the set of ontological concepts which annotate the event outputs $OUT_{ev}(ev_h)$ as well. An event of a Web API can be connected to an operation of another Web API in a publish/subscribe-like mechanism. An event-operation pair is represented in the model through an activation relationship, that is equipped with the set of correspondences between event outputs, changed when the event is raised, and inputs of operations triggered by event occurrence (see Figure 1). An activation relationship from an event ev_i^h of a descriptor SD_i and an operation op_j^k of another descriptor SD_j , is defined as follows:

$$act_{hk} = \langle ev_i^h, op_j^k, M_{hk} \rangle \quad (2)$$

where $ev_i^h \in EV_i$, $op_j^k \in OP_j$, M_{hk} is the set of correspondences between ev_i^h outputs and op_j^k inputs. IO correspondences are suggested comparing concepts used to annotate outputs and inputs and evaluating a concept affinity $CAff$ between them. The coefficient $CAff \in [0..1]$ evaluates the similarity between two concepts. To this purpose, we rely on techniques such as those extensively defined in [3]. Here we simply state that $CAff$ is based on WordNet terminological (domain-independent) relationships (e.g., synonymy, hypernymy) between terms used as concept names and on semantic relationships (e.g., equivalence, subsumption) between concepts defined in domain ontologies. The proposed model abstracts from implementation details in Web API description (in fact, Web APIs can be implemented through different technologies and can be accessed according to different messaging formats, such as XML, JSON, RSS items) and enables the design of common selection patterns to support Web application development.

4 Web API Registry Model

We propose a semantics-enabled Web registry model where descriptors are organized according to two kinds of semantic links: (i) *functional similarity links*, set between descriptors that provide similar functionalities, and (ii) *functional coupling links*, set between descriptors that can be included in the same Web application.

A *functional similarity link* between two semantic descriptors SD_i and SD_j is set if their categories are compatible (that is, at least one SD_j category is the same or less specific than one SD_i category) and is formally defined as:

$$\langle SD_i, SD_j, Sim_{IO}(SD_i, SD_j) \rangle \quad (3)$$

where $Sim_{IO}(SD_i, SD_j) \geq \delta$ is the functional similarity degree and $\delta \in [0, 1]$ is a threshold experimentally set. The functional similarity degree is computed to quantify how much SD_j provides at least the operations and I/Os required in SD_i ; no matter if SD_j provides additional operations and I/Os. The expression is reported in Table 1. The building block of this expression is the Dice coefficient used in Information Retrieval [11] and the computation of the concept affinity $CAff()$ between pairs of, respectively, (i) operations, (ii) I/Os parameters.

Functional coupling links between events EV_i raised by a semantic descriptor SD_i and operations OP_j executed by a semantic descriptor SD_j is formally defined as

$$\langle SD_i, SD_j, Coupl_{IO}(SD_i, SD_j) \rangle \quad (4)$$

where $Coupl_{IO}(SD_i, SD_j) \geq \theta$ is the coupling degree and $\theta \in [0, 1]$ is a threshold experimentally set. Each functional coupling link is associated to a set M_{ij}^c of candidate IO correspondences between outputs of EV_i and inputs of OP_j . $Coupl_{IO}()$ is obtained by computing values of event-operation coupling coefficients, $CouplEvOp(ev_i, op_j)$, evaluated as the total $CAff()$ between the outputs

Table 1. Coupling and functional similarity coefficients

FUNCTIONAL SIMILARITY DEGREE
$Sim_{IO}(SD_i, SD_j) = \frac{\sum_{s,t} CAff(in_s, in_t)}{ IN(SD_i) } + \frac{\sum_{h,k} CAff(out_h, out_k)}{ OUT(SD_i) } + \frac{\sum_{l,m} CAff(op_l, op_m)}{ OP(SD_i) } \in [0, 3]$
COUPLING DEGREE
$Coupl_{EvOp}(ev_i, op_j) = \frac{\sum_{h,k} CAff(out_h, in_k)}{ OUT_{ev}(ev_i) } \in [0, 1]$
$Coupl_{IO}(SD_i, SD_j) = \frac{\sum_{i,j} Coupl_{EvOp}(ev_i, op_j)}{ EV(SD_i) } \in [0, 1]$

of $ev_i \in EV_i$ and the inputs of $op_j \in OP_j$ (see Table II). Functional similarity and coupling have been detailed in [2].

Let be SD the set of semantic descriptors, $OP(SD)$ (resp., $EV(SD)$) the overall set of operations (resp., events) for all the semantic descriptors in SD , M^c the set of candidate correspondences between annotated event outputs and operation inputs detected during the functional coupling evaluation. The Web API registry is defined as a 4-uple $\langle SD, SL, CL, M^c \rangle$, where $SL \subseteq SD \times SD \times [0, 1]$ is the set of similarity links between descriptors and $CL \subseteq EV(SD) \times OP(SD) \times [0, 1]$ is the set of coupling links between event/operation pairs.

5 Web API Selection Patterns

In this section, we propose the selection patterns we identified to support the designer during the Web application development by exploiting semantic links in the Web API registry. We identified two possible patterns:

- Completion Pattern, to suggest a ranked list of descriptors that are functionally coupled to a selected descriptor SD_c and belongs to a given category cat_j ;
- Substitution Pattern, to suggest a ranked list of descriptors that are functionally similar to a selected descriptor SD_s and are functionally coupled with descriptors that are connected to SD_s through activation relationships in the Web application.

A *Selection Pattern* σ is defined as a 4-uple $\sigma_\tau = \langle SD_\tau, m_\tau, cond_\tau, \prec_\tau \rangle$, where τ is the goal of the selection pattern (completion or substitution). The metric m_τ is used to evaluate candidate descriptors to suggest. The condition $cond_\tau$ is used to filter out not relevant descriptors. Finally, \prec_τ is a ranking function to present the suggested descriptors.

According to this general definition, the *Completion Pattern*, denoted with σ_c , is defined as follows:

$$\sigma_c = \langle SD_c, Coupl_{IO}(SD_c, SD_i), Cond_c, \prec_c \rangle \quad (5)$$

where SD_c is the selected descriptor, $Cond_c$ is defined as $cat_j \in CAT(SD_i) \wedge Coupl_{IO}(\cdot) \geq \theta$, cat_j the category which suggested descriptors must belong to. Note that this implies the presence of a coupling link from SD_c to SD_i in the Web API registry. The function $\prec_c: SD \times SD$ defines a ranking over the set SD of descriptors in the registry such that $SD_i \prec_c SD_j$ if $Coupl_{IO}(SD_c, SD_i) \leq Coupl_{IO}(SD_c, SD_j)$. For a coupled descriptor that has been selected to be added to the Web application, semantic correspondences among I/Os are suggested to the designer. With reference to the motivating scenario, the completion pattern is used to suggest to Dante Web API descriptors that can be coupled with the Web API descriptor that, given the book/DVD title, returns the addresses of company shops where the book/DVD is available, for instance, Web APIs that visualize addresses on a map.

Similarly, the *Substitution Pattern*, denoted with σ_s , is formally defined as follows:

$$\sigma_s = \langle SD_s, \Lambda(SD_s, SD_i), Cond_s, \prec_s \rangle \quad (6)$$

where SD_s is the selected descriptor, $Cond_s$ is $Sim_{IO}(\cdot) \geq \delta \wedge CAT(SD_s) \cap CAT(SD_i) \neq \emptyset$. The function $\Lambda(SD_s, SD_i)$ is made up of three parts:

- the functional similarity between SD_s and SD_i , that is $Sim_{IO}(SD_s, SD_i)$;
- the sum of the functional coupling degree between SD_i and each SD_k in the Web application under development such that there exists an activation relationship from SD_s and SD_k ;
- the sum of the functional coupling degree between SD_h and SD_i , where SD_h are descriptors in the Web application under development such that there exists an activation relationship from SD_h and SD_s .

That is,

$$\Lambda(SD_s, SD_i) = Sim_{IO}(SD_s, SD_i) + \sum_k Coupl_{IO}(SD_i, SD_k) + \sum_h Coupl_{IO}(SD_h, SD_i) \quad (7)$$

The function $\prec_s: SD \times SD$ defines a ranking over the set SD of descriptors in the registry such that $SD_i \prec_s SD_j$ if $\Lambda(SD_s, SD_i) \leq \Lambda(SD_s, SD_j)$.

The condition $Sim_{IO}(SD_s, SD_i) \geq \delta$ gives priority to the presence of a similarity link from SD_s to SD_i in the registry. This is because the goal of this selection pattern is to find an alternative to SD_s (for example, different map viewer Web APIs). Low efforts to connect the candidate alternatives to the other descriptors is evaluated only for ranking purposes.

6 Preliminary Implementation and Evaluation

A prototype implementation of the framework to support easy Web application development, based on the discussed model, is presented in this section. The Web-based Graphical User Interface of the tool is shown in Figure 2. The interface has been implemented using the ZK open source framework, providing a library of AJAX components to implement different selection patterns.

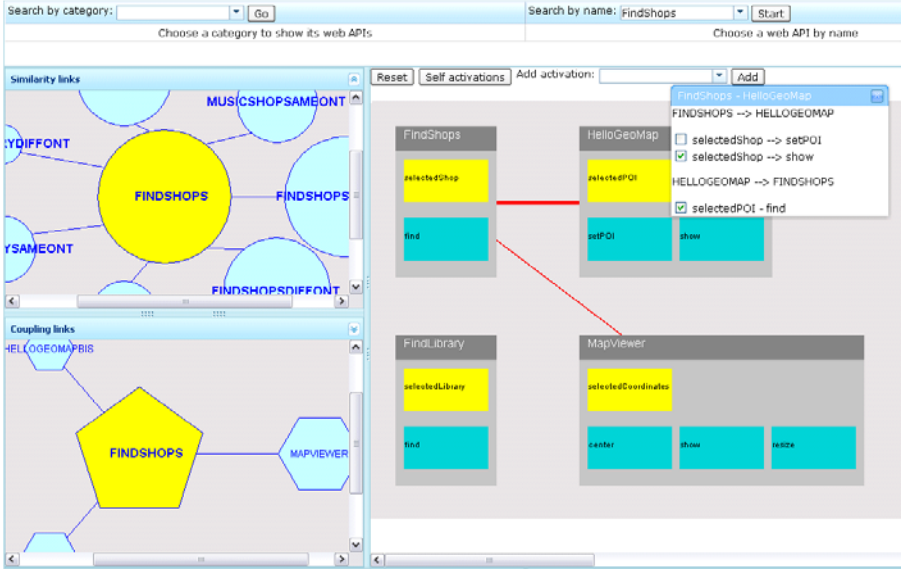


Fig. 2. Graphical User Interface of the design framework

In the upper part, the Web interface enables to search for available Web APIs by specifying their category or their name (**Search By category** and **Search By Name**). On the left, panels to browse the semantics-enabled Web API registry are shown. The selected descriptor SD_i is highlighted as a circle in the center of the “Similarity links” panel (e.g., the **FindShops** descriptor in Figure 2); all the descriptors SD_j related to SD_i through a similarity link are displayed as circles around SD_i ; dimension of each circle is proportional to the $Sim_{IO}(SD_i, SD_j)$ value. The selected descriptor SD_i is also highlighted as a pentagon in the center of the “Coupling links” panel; other descriptors SD_j coupled with SD_i are shown as hexagons around the pentagon; dimension of each hexagon is proportional to the $Coupl_{IO}(SD_i, SD_j)$ value (see, for example, the **MapViewer** descriptor). In the canvas on the right, the Web interface enables the designer to drag Web API descriptors and wire them to design the Web application. Each descriptor is represented as a rectangle containing the descriptor events (e.g., the **selectedShop** event for the **FindShops** descriptor) and operations (e.g., the **find** operation for the **FindShops** descriptor). By pushing the “Self connections” button, the system suggests activation relationships among descriptors which present IO correspondences. By clicking on the connection in the canvas, the designer can visualize the set of possible IO correspondences, which he/she can set or reset. The designer can also introduce his/her own activation relationships (“Add activations” facility).

Preliminary experiments have been performed to validate the effectiveness of the selection patterns. We used a 32 bit Windows machine with a 2.10 GHz AMD Athlon X2 Dual-Core CPU, 1 MB L2 Cache and 4GB RAM memory

using a set of annotated descriptors inspired by the motivating scenario (map visualization APIs, search APIs, etc.) and other descriptors randomly generated for testing the registry performances, obtaining a total of 150 descriptors. We started from a dataset of ten manually defined Web applications, each of them made up of a Web API descriptor SD_i coupled with other descriptors $\{SD_j\}$ from the registry. We searched for coupled descriptors for SD_i in the registry (Completion Pattern) and for descriptors which can substitute SD_i (Substitution Pattern). We considered a hit for the completion pattern evaluation for each Web API descriptor SD_j that is effectively suggested in the “Coupling link” panel. We evaluated the average precision and recall over the Web applications as the number of hits with respect to the total number of APIs returned by the registry (*precision*) and the total number of relevant APIs manually identified in the dataset (*recall*). Experiments have been performed by varying the coupling threshold θ in the range $[0, 1]$. The results are shown in Figure 3. Substitution pattern evaluation has been performed in a similar way.

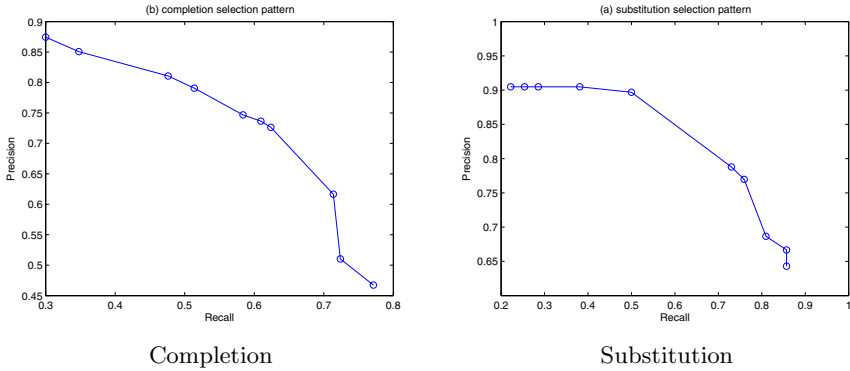


Fig. 3. Precision-Recall Curve for the two selection patterns

7 Conclusions and Future Work

In this paper we presented a support framework for easy Web application development, which includes organization of Web APIs based on automated matching techniques apt to establish semantic links between them according to properly defined similarity and coupling criteria and selection patterns to support interactive and proactive Web application development according to an exploratory perspective. A preliminary implementation and validation of the framework have also been presented. The framework is intended to be used in an integrated way with tools and frameworks for easy creation of Web applications starting from available Web APIs, to automate the generation of programming code to actually glue the selected Web APIs into the final application. An extensive experimentation in a real case scenario, including usability issues, is being performed.

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., Antonellis, V.D., Melchiori, M.: A Semantic Framework for collaborative Enterprise Knowledge Mashup. In: D’Atri, A., Ferrara, M., George, J.F., Spagnoletti, P. (eds.) Information Technology and Innovation Trends in Organizations, pp. 117–124. Physica Verlag, Heidelberg (2011)
3. Bianchini, D., Antonellis, V.D., Melchiori, M.: Flexible Semantic-based Service Matchmaking and Discovery. *World Wide Web Journal* 11(2), 227–251 (2008)
4. Daniel, F., Casati, F., Benatallah, B., Shan, M.: Hosted universal composition: Models, languages and infrastructure in mashArt. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009)
5. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A.P., Verma, K.: A Faceted Classification Based Approach to Search and Rank Web APIs. In: ICWS, pp. 177–184 (2008)
6. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. of the 35th Int. Conference on Very Large DataBases (VLDB 2009), Lyon, France, pp. 538–549 (2009)
7. Hoyer, V., Fischer, M.: Market overview of enterprise mashup tools. In: Bouguet-taya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 708–721. Springer, Heidelberg (2008)
8. Kopecky, J., Vitvar, T., Fensel, D.: hRESTS & MicroWSMO. Tech. rep., SOA4ALL Project, Deliverable D3.4.3 (2009)
9. 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)
10. Ngu, A.H.H., Carlson, M.P., Sheng, Q.Z., Young Paik, H.: Semantic-based mashup of composite applications. *IEEE T. Services Computing* 3(1), 2–15 (2010)
11. van Rijsbergen, C.J.: Information Retrieval. Butterworth (1979)

Preface to MORE-BI 2011

Business intelligence (BI) systems gather, store, and process data to turn it into information that is meaningful and relevant for decision-making in businesses and organizations. Successful engineering, use, and evolution of BI systems require a deep understanding of the requirements of decision-making processes in organizations, of the kinds of information used and produced in these processes, of the ways in which information can be obtained through acquisition and reasoning on data, of the transformations and analyses of that information, of how the necessary data can be acquired, stored, cleaned, how its quality can be improved, and of how heterogeneous data can be used together.

The first International Workshop on Modeling and Reasoning for Business Intelligence (MORE-BI 2011) was organized and collocated with the 30th International Conference on Conceptual Modeling (ER 2011), held in Brussels, Belgium, to stimulate discussions and contribute to the research on the concepts and relations relevant for the various steps in the engineering of BI systems, the conceptual modeling of requirements for BI systems, of the data used and produced by them, of the transformations and analyses of data, and associated topics to which researchers and practitioners of conceptual modeling can contribute, in the aim of constructing theoretically sound and practically relevant models and reasoning facilities to support the engineering of BI systems.

The call for papers attracted 23 abstracts, of which 22 resulted in full submissions. The submissions came from 64 authors from 14 countries and 4 continents. The program committee consisting of 31 researchers conducted three reviews of each submission and selected 7 submissions for presentation and discussion at the workshop, an acceptance rate of 32%.

We wish to thank all authors who have submitted their research to MORE-BI 2011. We are grateful to our colleagues in the steering committee for helping us define the topics and scope of the workshop, our colleagues in the program committee for the time invested in carefully reviewing the submissions under a very tight schedule, the participants who have helped make this an interesting event, and the local organizers and workshop chairs of ER 2011.

We hope that you find the workshop program and presentations of interest to research and practice of business intelligence, and that the workshop has allowed you to meet colleagues and practitioners focusing on modeling and reasoning for business intelligence. We look forward to receive your submissions and meet you at the next edition of the International Workshop on Modeling and Reasoning for Business Intelligence.

July 2011

Ivan J. Jureta
Stéphane Faulkner
Esteban Zimányi

Formal Concept Analysis for Qualitative Data Analysis over Triple Stores

Frithjof Dau and Barış Sertkaya

SAP Research Center Dresden, Germany
(frithjof.dau,baris.sertkaya)sap.com

Abstract. Business Intelligence solutions provide different means like OLAP, data mining or case based reasoning to explore data. Standard BI means are usually based on mathematical statistics and provide a quantitative analysis of the data. In this paper, a qualitative approach based on a mathematical theory called "Formal Concept Analysis" (FCA) is used instead. FCA allows clustering a given set of objects along attributes acting on the objects, hierarchically ordering those clusters, and finally visualizing the cluster hierarchy in so-called Hasse-diagrams. The approach in this paper is exemplified on a dataset of documents crawled from the SAP community network, which are persisted in a semantic triple store and evaluated with an existing FCA tool called "ToscanaJ" which has been modified in order to retrieve its data from a triple store.

1 Introduction

Business Intelligence (BI) solutions provide different means like OLAP, data mining or case based reasoning to explore data. Standard BI means are usually designed to work with numerical data, thus they provide a quantitative analysis of the data (aka "number crunching") based on mathematical statistics. In fact, classical BI examples show "accounting, finance, or some other calculation-heavy subject" [10]. To some extent, though arguably oversimplified, one can understand BI as acting on lists or tables filled with numbers.

Compared to number crunching, Formal Concept Analysis (FCA) [3] provides a complementing approach. The starting point of FCA are crosstables (called "formal contexts"), where the rows stand for some objects, the columns for some attributes, and the cells (intersections of rows and columns) carry the binary information whether an attribute applies to an object (usually indicated by a cross) or not. Based on this crosstable, the objects are clustered to meaningful sets. These clusters form a hierarchy, which can be visually displayed, e.g. by a so-called Hasse-diagram. A short introduction into FCA, as needed for this paper, is provided in the next section.

A general overview over the benefits of FCA in information science is provided by Priss in [8]. Relevant for this paper are the relationships between FCA and both Business Intelligence (BI) and Semantic Technologies (ST).

With respect to BI, FCA can be for example considered as a data mining technology, particularly for mining association rules [7]. More relevant to this paper

is the approach to explore data in relational databases with FCA. As described in the next section, a method called "conceptual scaling" allows transforming columns in a database, filled with arbitrary values, into formal contexts. Such scales can be compared to dimensions in BI applications. The exploration of data in databases with FCA is for example described in [4,6,13,12]. A number of tools for FCA have been developed. Most important for this paper is Toscana [14,9], developed in C, and its Java-based successor ToscanaJ [11]. Moreover, it should be mentioned that FCA has been used for exploring data warehouses as well [5].

FCA targets a formalization of the human understanding of concepts with their extensions and intensions, thus FCA indeed is a semantic technology. Though it does not belong to the core of Semantic Web technologies, FCA provides decent means to define and analyze concept hierarchies, so it comes as no surprise that FCA has been used in the realm of querying, browsing, completing and visualizing ontologies (e.g. OWLFCViewTab and OntoComp [11] plugins for the Protege ontology editor, and OntoViz), ontology alignment (e.g. FCA-Merge and OntEx), ontology engineering (e.g. relational exploration or role exploration) and ontology learning (e.g., Text2Onto). In this paper, we exemplify the benefits of FCA for (semantically enabled) BI by analyzing data in a triple store with FCA methods. In order to do so, the existing ToscanaJ tool has been modified such that it can retrieve data from triple stores instead of relational databases. A short introduction into ToscanaJ and its modifications are provided in Sec. 3. An often named benefit of ST compared to relational databases are the ST capabilities to better deal with unstructured data like text-documents. FCA has already been employed to create concept hierarchies out of the content of text documents. In this paper, we apply FCA on a dataset of documents crawled from the SAP community network [3] (SCN), but do not target to investigate the contents of the documents, but utilize meta-data of the documents (which have been created in the crawling process) for FCA-purposes. This use case is described in Sec. 4.

2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of applied mathematics that is based on a lattice-theoretic formalization of the notions of concepts and conceptual hierarchies. FCA provides efficient algorithms for analyzing data and discovering hidden dependencies in the data. It also allows the user to visualize the data in an easily understandable way. In FCA, data is represented in the form of a *formal context*, which in its simplest form is a way of specifying which attributes are satisfied by which objects.

Example 1. Consider the formal context in Fig. 1. It shows information about the gender of four customers and the product groups they are interested in. For

¹ <http://toscanaj.sourceforge.net>

² <http://ontocomp.googlecode.com>

³ <http://www.sdn.sap.com/irj/scn/index>

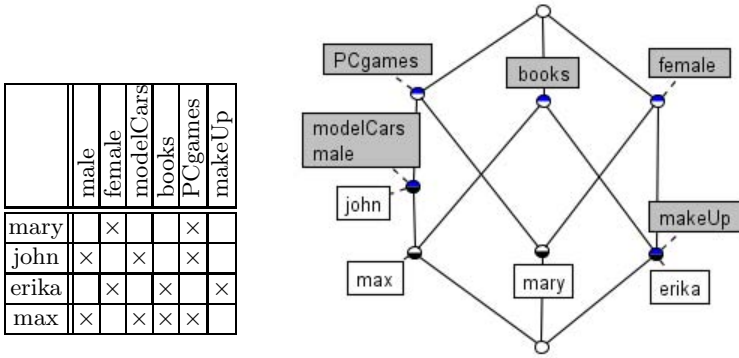


Fig. 1. Formal context of customers (left) and its concept lattice (right)

instance the last row states that Max is a male, he is interested in model cars, books and PC games but he is not interested in make-up items.

Given such a formal context, the first step for analyzing this context is usually computing the *formal concepts* of this context, which are “natural clusterings” of the data in the context. A formal concept is a pair consisting of an object set A and an attribute set B such that the objects in A share the attributes in B , and B consists of exactly those attributes that the objects in A have in common. The object set A is called the *extent*, and the attribute set B is called the *intent* of the formal concept (A, B) .

Example 2. Consider the formal context given in Ex. 1. It has nine formal concepts. One of them is $(\{max\}, \{male, modelCars, PCgames, books\})$ with the extent $\{max\}$ and the intent $\{male, modelCars, PCgames, books\}$. Note that max is male and has exactly the interests in modelCars, PCgames, books. On the other hand, max is the only male person with these interests. Another (less trivial) formal concept is $(\{john, max\}, \{male, modelCars, PCgames\})$. Both john and max are male and have modelCars and PCgames as (common) interests, and they are indeed the only male persons with (at least) these interests.

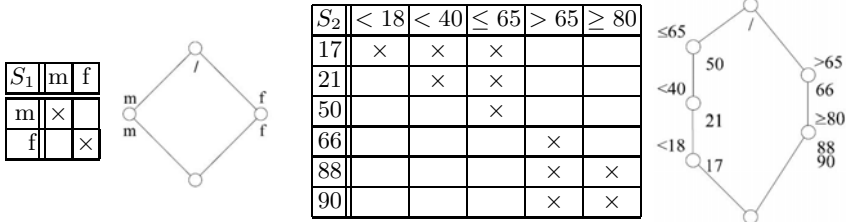
Once all formal concepts of a context are obtained, one orders them w.r.t. the inclusion of their extents (equivalently, inverse inclusion of their intents). For example, the two formal concepts of the above given example are ordered that way. This ordering gives a complete lattice (e.g. a hierarchy where any two elements have -like in trees- a least upper bound and -unlike in trees- a greatest lower bound), called the *concept lattice* of the context. A concept lattice contains all information represented in a formal context, i.e., we can easily read off the attributes, objects and the incidence relation of the underlying context. Moreover, concept lattice can be visualized, which makes it easier to see formal concepts of a context and interrelations among them. Thus it helps to understand the structure of the data in the formal context, and to query the knowledge represented in the formal context.

The nodes of a concept lattice represent the formal concepts of the underlying context. In order to improve readability of the lattice, we avoid writing down the extent and intent of every single node. Instead, we label the nodes with attribute and object names in such a way that every name appears only once in the lattice. In this labelling, the intent of the formal concept corresponding to a node can be determined by the attribute names that can be reached by the ascending lines, and its extent can be determined by the object names that can be reached by the descending lines. For instance consider the concept lattice in Figure 1 that results from the formal context in Example 1. The attribute names are written in boxes with gray background and object names are written in boxes with white background. The intent of the formal concept marked with the attribute name *books* is $\{books\}$ since there is no other attribute name that can be reached by an ascending line, and its extent is $\{max, erika\}$ since these are the only two object names that can be reached by a descending line from it. Similarly, the concept marked with the attribute names *modelCars* and *male*, and the object name *john* has the intent $\{modelCars, male, PCgames\}$ and the extent $\{john, max\}$.

FCA, as it has been described so far, can only deal with *binary* attributes. For real data, the situation is usually different: Attributes assign specific values (which might be strings, numbers, etc) to data. For example, RDF-triples (s, p, o) are exactly of this form: The attribute p - from now on we will use the RDF-term "property" instead- assigns the value o to the entity s . In FCA, a process called "conceptual scaling" is used to deal with this issue.

Let a specific property be given with a set of possible values. A *conceptual scale* is a specific context with the values of the property as formal objects. The choice of the formal attributes of the scale is a question of the design of the scale: The formal attributes are meaningful attributes to describe the values; they might be different entities or they might even be the values of the property again. To exemplify conceptual scaling, we reuse a toy example from [15], which is the following table provided on the right with two many-valued properties "sex" and "age". Note that empty cells are possible as well.

	sex	age
Adam	m	21
Betty	f	50
Chris		66
Dora	f	88
Eva	f	17
Fred	m	
George	m	90
Harry	m	50



Next, two conceptual scales for the properties "sex" and "age" and their line diagrams are provided.

With conceptual scales, the initial many-valued context can be transformed into a standard context, so that the corresponding concept lattice can be displayed.

For conceptual scales, the following two points should be noted:

1. There is no standard or even necessary interpretation of an attribute: It has to be decided by the field expert which scale is appropriate. As discussed in [2], this is indeed not a drawback, but an advantage of FCA.
2. Conceptual scales do not depend on the real data, but only on the properties (and their values, of course) used in the data set. As one can see in the example, a realized context is derived from the scales and the real data in a later step after the scales have been created.

Both points are important for ToscanaJ, which is discussed in the next section.

3 ToscanaJ

There is a variety of software for FCA available. Most of them support the creation of contexts from scratch and the subsequent computation and display of the corresponding concept lattices. Contrasting this approach, Elba and ToscanaJ are a suite of mature FCA-tools which allow to query and navigate through data *in databases*. They are intended to be a *Conceptual Information System (CIS)*. CISs are "systems that store, process, and present information using concept-oriented representations supporting tasks like data analysis, information retrieval, or theory building in a human centered way." Here, a CIS is an FCA-based system used to analyze data stored in one table of an RDBMS.

Similar to other BI-systems, in CIS we have to distinguish between a design phase and a run-time-phase (aka usage phase), with appropriate roles attached to the phases. In the design phase, a CIS engineer (being an expert for the CIS) together with a domain expert who has limited knowledge of a CIS) develops the CIS schema, i.e. those structures which will be later on used to access the system. This schema consists of manually created conceptual scales. Developing the scales is done with a CIS editor (Elba) and usually a highly iterative process. In the run-time phase, a CIS browser (ToscanaJ) allows a user to explore and analyze the real data in the database with the CIS schema.

The original Elba/ToscanaJ-suite has been developed to analyze data in a *relational table*, i.e. a table in a RDBMS or an excel-file. We have extended the suite in order to be able to access data in a *triple store*. This extended version of the suite uses the Sesame framework⁴ for accessing a triple store and querying the RDF data therein. It provides two ways of connecting to a triple store over Sesame. One of them is over HTTP via Apache Tomcat⁵, the other one is over the SAIL API⁶. Tomcat is an open source software implementation of the Java

⁴ See <http://www.openrdf.org/doc/sesame2/system>

⁵ See <http://tomcat.apache.org/>

⁶ See <http://www.openrdf.org/doc/sesame2/system/ch05.html>

Servlet and JavaServer Pages technologies by the Apache Software Foundation. The SAIL API (Storage And Inference Layer) is a low level system API for RDF stores and inferencers. It is used for abstracting from the storage details, allowing various types of storage and inference to be used.

In a triple store we do not directly have the notions of tables and columns like in databases. As table information we use the type information in the triple store: we treat the objects of triples with the predicate `rdf:type` as tables. As column information, we use the predicates relating the subjects of the selected type to any object. More precisely, in order to detect the columns we get those subjects of the selected type and retrieve all distinct predicates that relate these subjects to an object.

The Elba/ToscanaJ-suite provides different kinds of conceptual scales. We have extended three of them –namely nominal scales, attribute scales and context tables– in order to act on triple stores.

Nominal scales are the simplest type of scales one can automatically create in Elba. They are used for properties with mutually exclusive values. For a given property, the formal attributes of the nominal scale are selected values of that property. As each object is assigned at most one of these values, the attribute concepts form an anti-chain, and by definition, the scale cannot reveal any insight into attribute dependencies. In the example provided in the next section, we consider a predicate `threadStatus` for messages, which has the values `Answered` and `Unanswered`. This predicate is modelled as conceptual scale.

Attributes scales offer an attribute centered view which is very close to "classical" formal contexts and which allows to create complex scales in an intuitive manner. In an attribute list scale, each attribute is a property-value pair, which is manually selected from the triple store. Moreover, the CIS engineer can choose between a) ?use only combinations existing in the database? and b) ?use all possible combination?. If option a) is selected, then the diagram will only consist of concepts that could be derived from the data in the triple store, thus the diagram will reveal insights into dependencies between property-value pairs. If b) is chosen, a diagram of a Boolean lattice of all listed property-value pairs will be created independent of whether there exists objects in the triple store for each property-value combination or not.

Context table scales offer the most freedom and power to the CIS engineer. In context tables, arbitrary labels act as formal attributes. As now, in contrast to the last two types of scales, no property-value pairs are chosen as attributes, it has now explicitly to be specified which objects of the data set fulfill the formal attributes. This is done by entering SPARQL expressions, which act as formal objects, and by entering the incidence relation as well, i.e. the relation which here relates the formal objects (the SPARQL expressions) to the attributes (the labels).

4 Use Case

In order to evaluate our approach, we have used a dataset crawled from the SAP Community Network (SCN). SCN contains a number of forums for SAP

users and experts to share knowledge, or get help on SAP topics and products. The dataset we have used is taken from the forum *Service-Oriented Architecture (SOA)*, which contains 2600 threads and 10076 messages. The dataset is annotated by the crawler using ontologies from the NEPOMUK project. The used ontologies and their meanings are provided below along with short descriptions taken from the project website⁷.

- NEPOMUK Information Element Ontology (NIE): The NIE Framework is an attempt to provide unified vocabulary for describing native resources available on the desktop.
- NEPOMUK file ontology (NFO): The NFO intends to provide vocabulary to express information extracted from various sources. They include files, pieces of software and remote hosts.
- NEPOMUK Message Ontology (NMO): The NMO extends the NIE framework into the domain of messages. Kinds of messages covered by NMO include Emails and instant messages.
- NEPOMUK Contact Ontology (NCO): The NCO describes contact information, common in many places on the desktop.

From these ontologies, our dataset uses the following classes as types:

- `nie#DataObject`: A unit of data that is created, annotated and processed on the user desktop. It represents a native structure the user works with. This may be a file, a set of files or a part of a file.
- `nfo#RemoteDataObject`: A file data object stored at a remote location.
- `nie#InformationElement`: A unit of content the user works with. This is a superclass for all interpretations of a `DataObject`.
- `nco#Contact`: A Contact. A piece of data that can provide means to identify or communicate with an entity.
- `nmo#Message`: A message. Could be an email, instant messaging message, SMS message etc.

For analyzing experience levels of the users of the SOA forum, we used the *Contact* type above and created a scale based on the number of posts, number of questions, number of resolved questions information provided in the data. We have named users that have less than 50 posts as *newbie*, users that have more than 300 posts as *frequent*, users that have more than 1000 posts as *profi*, users that have asked more than 310 questions as *curious* and people that have resolved more than 230 questions as *problem solver*. Note that this scale uses different measures (number of posts, number of questions, numbers of answers).

The concept lattice in Figure 2 shows number of users with the mentioned experience levels. The diagram clearly displays the sub/super-concept-relationships between the experience levels, which is one of the main distinguishing features of visualizing data using concept lattices. E.g. we can read from the lattice that

⁷ <http://www.semanticdesktop.org/ontologies>

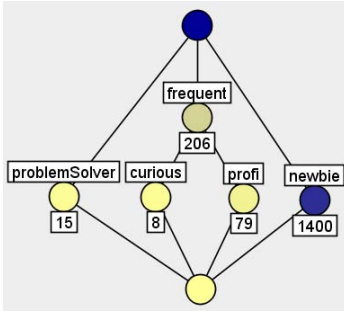


Fig. 2. Diagram of the scale based on number of posts, questions, resolved questions

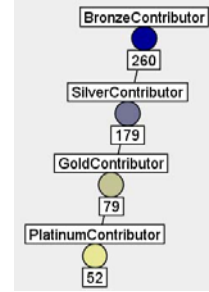


Fig. 3. Diagram of the scale based on number of points

curious and professional users are both also frequent users, whereas problem solvers and newbies are not.

Next, for analyzing experience levels based on the *number of points* information in our dataset we created another scale. This time, as labels we took contributor types that are officially defined by SCN as *bronze*, *silver*, *gold* and *platinum* contributors, which have more than 250, 500, 1500 and 2500 points respectively. The concept lattice of this scale is shown in Figure 3. This scale is a so-called *ordinal* scale, which means that the formal concepts are ordered as a chain. This is also easily seen in the concept lattice of this scale. Obviously, a user that has more than 2500 points also has more than 1500 points, and so on.

The above displayed concept lattices are separately informative about the properties of forum users, i.e., the first one about experience levels based on number of posts, questions and resolved questions, and the second one about number of points. One of the most powerful techniques of FCA is to “combine” such lattices to give a combined view of several lattices together, which is called a *nested line diagram*. In its simplest form, a nested line diagram is a concept lattice whose concepts are themselves also concept lattices. Nested line diagrams allow the user to select a concept and zoom into it to see the lattice nested in that concept. Figure 4 shows the nested line diagram of the diagrams in the Figures 2 and 3. Note that the outer diagram is actually the one in Figure 3. The four bigger circles correspond to the four types of contributors in that figure. The inner diagrams are the diagram in Figure 2. Figure 5 shows an excerpt of the nested diagram that corresponds to the node *golden contributor*, and Figure 6 shows the inner diagram of this node. Note that the number of users corresponding to different levels of experience in this diagram differs from that of diagram in Figure 2. The reason is that, now we zoomed into the node gold contributor so the information in the inner diagram is restricted to the gold contributors only. For instance, as seen in this diagram there are no newbies that are gold contributors, which is quite natural. On the other hand 79 of the gold contributors are profi users. In ToscanaJ, and thus in our extension of it

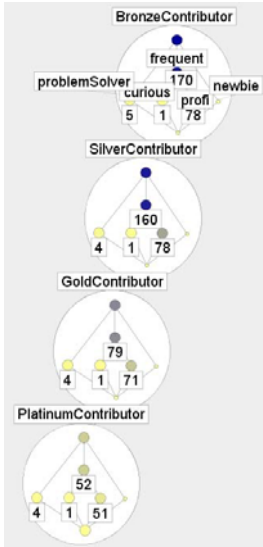


Fig. 4. Nesting the above two scales

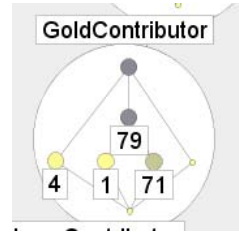


Fig. 5. Detail of Figure 4

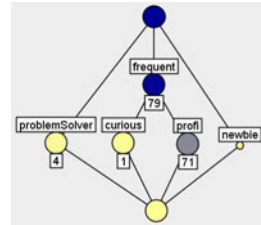


Fig. 6. Inner diagram of Fig. 5

to triple stores, one can nest an arbitrary number of diagrams and can browse nested diagrams easily by zooming in and out.

5 Conclusion and Further Research

We have discussed how FCA can be used as a methodology for analyzing data in triple stores by extending the Toscana.j suite. As scales in the Toscana.J workflow are manually crafted in the design phase of a CIS, this workflow is feasible for stable schemata. For ST, this is usually not the case: here the paradigm of agile schema development is prevalent. As future work we plan to implement automatic or at least semi-automatic generation of scales based both on the schema information and the actual data in the triple store. existing BI approaches. Another future research direction is the development of hybrid solutions, combining "classical" BI with FCA. This covers combinations of scales and their diagrams with BI diagrams for numerical data, like pie charts or sun-burst diagrams, and compared to nesting of scales, different approaches for using simultaneously several scales. In our work we have considered RDF models only as simple object-attribute-value models, ignoring the subclass relationships. As future work we are also going to work on integrating such knowledge into FCA as background knowledge for concept lattice computation.

Disclaimer: Parts of this work have been carried out in the CUBIST project, which is funded by the European Commission under the 7th Framework Programme of ICT, topic 4.3: Intelligent Information Management.

References

1. Becker, P., Hereth, J., Stumme, G.: ToscanaJ: An open source tool for qualitative data analysis. In: Duquenne, V., Ganter, B., Liquiere, M., Nguifo, E.M., Stumme, G. (eds.) *Advances in Formal Concept Analysis for Knowledge Discovery in Databases, FCAKDD 2002* (2002)
2. Dau, F., Klinger, J.: From Formal Concept Analysis to Contextual Logic. In: Ganter, B., Stumme, G., Wille, R. (eds.) *Formal Concept Analysis. LNCS (LNAI)*, vol. 3626, pp. 81–100. Springer, Heidelberg (2005)
3. Ganter, B., Wille, R.: *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin (1999)
4. Hereth, J.: *Formale begriffsanalyse und data warehousing*. Masters thesis, TU Darmstadt, Germany (2000)
5. Hereth, J.: Relational Scaling and Databases. In: Priss, U., Corbett, D., Angelova, G. (eds.) *ICCS 2002. LNCS (LNAI)*, vol. 2393, pp. 62–76. Springer, Heidelberg (2002)
6. Hereth, J., Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery - a human-centered approach. *Journal of Applied Artificial Intelligence (AAI)* 17(3), 281–301 (2003)
7. Lakhali, L., Stumme, G.: Efficient Mining of Association Rules Based on Formal Concept Analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) *Formal Concept Analysis. LNCS (LNAI)*, vol. 3626, pp. 180–195. Springer, Heidelberg (2005)
8. Priss, U.: Formal concept analysis in information science. *Annual Review of Information Science and Technology* 40 (2005)
9. Roth-Hintz, M., Mieth, M., Wetter, T., Strahinger, S., Groh, B., Wille, R.: Investigating snomed by formal concept analysis. In: *Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (2000)
10. Scheps, S.: *Business Intelligence for Dummies*. John Wiley and Sons Ltd., Chichester (2008)
11. Sertkaya, B.: ONTOCOMP: A PROTÉGÉ Plugin for Completing OWL Ontologies. In: Aroyo, L., Traverso, P. (eds.) *ESWC 2009. LNCS*, vol. 5554, pp. 898–902. Springer, Heidelberg (2009)
12. Stumme, G.: Conceptual on-line analytical processing. In: Tanaka, K., Ghandeharizadeh, S., Kambayashi, Y. (eds.) *Information Organization and Databases*. ch. 14. Kluwer, Boston (2000)
13. Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery in databases using formal concept analysis methods. In: Zytzkow, J.M., Quafofou, M. (eds.) *PKDD 1998. LNCS (LNAI)*, vol. 1510, pp. 450–458. Springer, Heidelberg (1998)
14. Vogt, F., Wille, R.: Toscana — a graphical tool for analyzing and exploring data. In: Tamassia, R., Tollis, I.G. (eds.) *GD 1995. LNCS*, vol. 1027, pp. 226–233. Springer, Heidelberg (1996)
15. Wolff, K.E.: A first course in formal concept analysis. In: Faulbaum, F. (ed.) *Proceedings of Advances in Statistical Software*, vol. 4, pp. 429–438 (1993)

Semantic Cockpit: An Ontology-Driven, Interactive Business Intelligence Tool for Comparative Data Analysis*

Bernd Neumayr¹, Michael Schreffl¹, and Konrad Linner²

¹ Department of Business Informatics - Data & Knowledge Engineering
Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria
² *solvistas* GmbH, Römerstraße 18/1, 4020 Linz

Abstract. Business analysts frequently use Cockpits or Dashboards as front ends to data warehouses for inspecting and comparing multi-dimensional data at various levels of detail. These tools, however, perform badly in supporting a business analyst in his or her business intelligence task of understanding and evaluating a business within its environmental context through comparative data analysis. With important business knowledge either unrepresented or represented in a form not processable by automatic reasoning, the analyst is limited in the analyses that can be formulated and she or he heavily suffers from information overload with the need to re-judge similar situations again and again, and to re-discriminate between already explained and novel relationships between data. In an ongoing research project we try to overcome these limitations by applying and extending semantic technologies, such as ontologies and business rules, for comparative data analysis. The resulting *Semantic Cockpit* assists and guides the business analyst due to reasoning about various kinds of knowledge, explicitly represented by machine-processable ontologies, such as organisation-internal knowledge, organisation external domain knowledge, the semantics of measures and scores, knowledge about insights gained from previous analysis, and knowledge about how to act upon unusually low or high comparison scores. This paper outlines the architecture of the Semantic Cockpit and introduces its core ideas by a sample use case.

1 Introduction

Comparative analysis of data gathered about past business activities is one of the critical initial steps by a business analyst in her or his business intelligence task of understanding and evaluating a business within its environmental context. Data warehouse (DWH) technology for collecting and processing relevant data and accompanying On-Line-Analytical-Processing (OLAP)-Tools support the business analyst to analyze and compare data aggregated over various dimensions (such as *time, location, product group*). Dashboards and cockpits have been developed

* *SemCockpit* is a collaborative research project funded by the Austrian Ministry of Transport, Innovation, and Technology in program 'FIT-IT Semantic Systems and Services'. The project started in March 2011 and will end in August 2013.

recently as front end tools for data warehouses that allow the business analyst to predefine and interactively select analysis reports and to build alert mechanisms into user interfaces. These tools assist him through different kinds of graphs to interactively inspect aggregated measures (e.g., average number of prescriptions per patient) related to group of entities of interest (e.g., general practitioners) and to compare these against corresponding measures of one or multiple peer (or comparison) groups, potentially also through a normalized score, visualized by a gauge indicating the result of this comparison. Thereby, he identifies opportunities and problems, explores possible causes, or discovers interesting phenomena (relationships between data). His comparative analysis may lead immediately to insights that can be used for strategy formulation or implementation, or may trigger further analysis by dedicated data mining tools, whereby the insights gained so far help him to guide the formulation of the right analysis questions.

“Unintelligent” Cockpits offer only limited support to the business analyst for his comparative analysis. With important meta knowledge unrepresented or represented in a form not processable by automatic reasoning, he is limited in the analysis that can be formulated and his success entirely depends on his domain knowledge as business expert. Some simple examples: The analysis “compare the prescription habits of general practitioners in tourist regions to other practitioners in relation to the percentage of prescribing generic drugs vs. innovator drug X” cannot be formulated if domain knowledge about pharmacology is not represented in machine-processable form. New concepts, even as simple as “general practitioners in tourist regions”, are not easy to formulate, if expressible at all. Judging scores comparisons of flu medications across months requires the background knowledge that sales of flu medications are typically 30 % higher in winter months. Business analysts have to deal with a lot of data and commonly have insufficient personal experience to guide themselves during the process of information selection and interpretation [9]. It is up to the business analyst to discriminate between usual phenomena and interesting situations that may give rise to further action or need further analysis; moreover, similar results re-appear all the time, overloading analysis results with already explained usual phenomena such that he may unnecessarily repeat further analysis steps.

The Semantic Cockpit assists and guides the business analyst in defining analysis tasks, discriminating between usual phenomena and novel interesting situations to be followed up to avoid him being drown in information, and recommending actions or indicating further analysis steps that should be applied to follow up interesting situations. The Semantic Cockpit is an intelligent partner of the business analyst due to reasoning about various kinds of knowledge, explicitly represented by machine-processable ontologies such as: organization-internal knowledge (e.g., definitions of concepts corresponding to business terms such as weekend and evening visits), organization external domain knowledge (e.g., about medications or illnesses), the semantics of measures and measure values (i.e., about upon what possible groups of interests a measure may be defined and which particular group of interest a measure value describes), the semantics of scores (i.e., what measure about a group of interest is scored against which group of comparison along what dimension), knowledge about insights gained from previous analysis (e.g., typical percentage of higher sales of flu medications in winter months), and knowledge about how to act upon a striking low or high

comparison score (e.g., initiate further analysis or recommend actions such as an additional training for pharmacists).

The paper is structured as follows: in Sec. 2 we introduce the overall architecture of the Semantic Cockpit framework and introduce a show case from the health insurance industry. In Sec. 3 we describe the Semantic Cockpit process. In Sec. 4 we describe cockpit design and analysis by a use case. In Sec. 5 we give a concise overview of related work. Sec. 6 concludes the paper with an overview of the current state of the project.

2 Architecture and Sample Analysis Task

The left hand side of Fig. 1 illustrates the work of a business analyst with a conventional OLAP-frontend to a data warehouse, while the right hand side shows the setting and components of the *Semantic Cockpit*. In the former, the analyst poses various OLAP queries against the data warehouse to inspect and compare data from various perspectives at different detail with the aim to discover interesting deviations or abnormalities that give rise to new insights about how the business functions, potential drivers of best-practice examples or deficiencies in the organization, such as potential fraud. The cockpit itself is un-intelligent. The business analyst brings in all required knowledge and is unassisted in combining it with the data in the data warehouse: He must manually link domain knowledge (e.g. about therapeutical drugs) to data in the data warehouse (and cannot link both in one query), must re-formulate similar queries all the time as the cockpit does not assist him what kinds of measures and scores previously defined may be re-used (as such or modified) in a new context, must remember previous insights to orientate himself in the bulk of information provided, must remember previously encountered situations to discriminate whether a just discovered deviation in data is already covered by similar previous insights, and he must remember whether this deviation can be justified by given, unchangeable environment conditions or may give rise to strategic action. Apart from the possibility to inspect data, he is unassisted by the OLAP-frontend in preparing an analysis report for management.

The right hand side of Fig. 1 illustrates the work of a business analyst with the *Semantic Cockpit* and the cockpits main components. The *Semantic Cockpit* is an intelligent partner of the business analyst. It has access to various machine-readable sources of knowledge formerly known only by the business analyst and now assists the cockpit in intelligently interacting with him: a *domain ontology* comprises individuals corresponding to entities in the data warehouse and linking to concepts of (imported) ontologies (e.g., an *external drug ontology* such as the *Australian Medicines Terminology*¹), a *multi-dimensional measure and score ontology* describing the semantics of measures and scores based on concepts defined upon the domain ontology, a *judgment rule ontology* that captures knowledge about previous insights gained and related actions.

In Sec. 4 we will explain the functioning of the cockpits major components (*Multi-Dimensional-Ontology (MDO) Engine*, *Semantic Cockpit Frontend*, and

¹ <http://www.ehealthinfo.gov.au/vendors/clinical-terminologies/australian-medicines-terminology/>

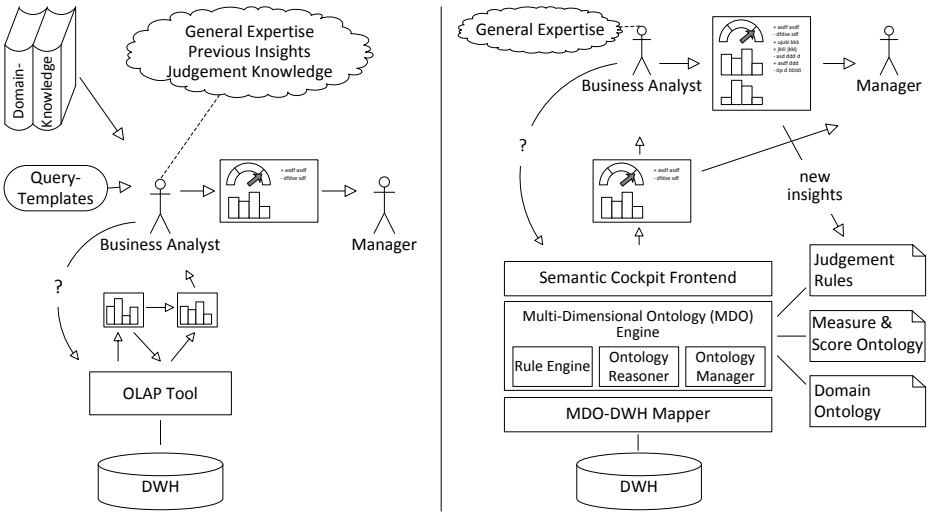


Fig. 1. Conventional Comparative Data Analysis (left) vs. Semantic Cockpit: Setting & Components (right)

MDO-DataWarehouse Mapper) together with a step-by-step description of the interaction of the business analyst with the Semantic Cockpit using the following simplified show case:

Upper Austrian Public Health Insurance wishes to increase the percentage of prescriptions of low-priced Psychotropic Drugs with indication “anti depression” during 2009 by General Practitioners in medium-to-large cities in Upper Austria and asks the business analyst to prepare a comparative analysis report and, if possible, recommend strategic or operational actions. Notice that such open-ended analysis questions tied to specific kinds of drugs, groups of doctors, and locations are common in a health insurance setting and, at such early state of investigation, are usually not solved by specific data mining tools (But these may be well used in later stages of an investigation when particular mining tasks can be formulated based on results from the preceding open-ended analysis). Such an open-ended analysis will, for example, comprise comparisons of a group of interest (prescriptions of low-priced Psychotropic Drugs with indication “anti depression” of 2009 in Upper Austria in medium-to-large cities) against several groups of comparison, by generalizing or varying the dimensions (e.g., from Upper Austria to Austria, or from 2009 to 2008) to gain insights based on studying in detail from various perspectives the data tied to striking differences encountered.

3 Semantic Cockpit Process

Fig. 2 shows a selection of the ontologies of our sample Semantic Cockpit and indicates the process back-stage of the use case described in Sec. 4.

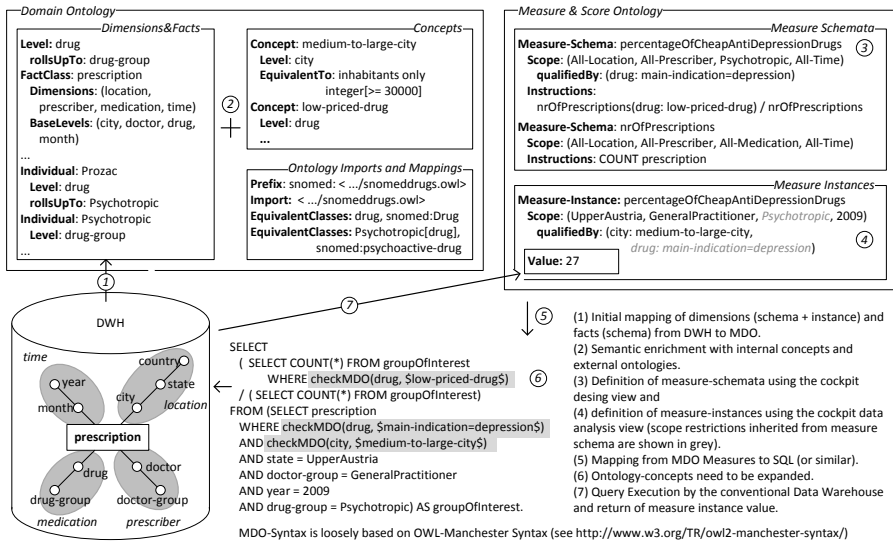


Fig. 2. Semantic Cockpit Process (simplified, without Scores and Judgement Rules)

The *Domain Ontology* is initialized by mapping the schema of the data warehouse into an ontological representation, abstracting from specific data warehouse details. While dimension instances (entities) and the roll-up hierarchy of dimension instances are also included in the ontology, facts reside only in the data warehouse. This initial Domain Ontology is enriched by additional *Concepts* (such as *medium-to-large-city*) that constitute business-relevant subsets of entities of one dimension level and by links to external ontologies. In our example, the OWL class *psychoactive-drug* in the external drug ontology is equivalent to the descendants of drug group *Psychotropic* at level *drug* (notated by *Psychotropic[drug]*). The measure ontology shows a multi-dimensional ontology representation of the measure schema defined by the analyst.

A *measure schema* consists of a *scope* that is given by a *qualified multi-dimensional point*, i.e., a cross product of individuals (dimension instances in the DWH) optionally with one or more *qualifiers* that are concepts or ontological expressions over concepts at some dimension level beneath the indicated multi-dimensional point, and of measurement *instructions*. The measure is defined for every multi-dimensional point subsumed by the scope.

The measurement defined by the business analyst is represented as *measure instance* in the md-measure ontology. To calculate the measure value, the *Ontology-DWH Mapper* takes this measure instance together with the measure schema (and its ontological representation of the associated measurement instructions) as input and translates it into an OLAP query with evaluation calls for embedded ontological predicates. This OLAP query is evaluated against the Data Warehouse after ontological predicates have been evaluated in interaction with the *MDO engine* (see Fig. 2).

4 Sample Use Case of Cockpit Design and Analysis

We will now describe step-by-step how the business analyst interacts with the *Semantic Cockpit* and how the *Semantic Cockpit* exploits its knowledge to intelligently support the analyst using our sample analysis task.

First, the business analyst opens the *Cockpit Design View* (see Fig. 3), which is partitioned into five parts: the *Measure Definition Board*, the *Score Definition Board*, the *Judgment Rule Board*, the *Domain Ontology Context View*, and the *Measure and Score Ontology Context View*.

Fig. 3. Cockpit Design View (simplified)

In our case, the business analyst wishes to define a general measure `percentageOfCheapAntiDepressionDrugs` for psychotropic drugs that have its main indication “anti depression” for all locations, all prescribers, and all times. To define this scope, he moves down in the *Measure Definition Board* in the Drug Dimension from the top-most multi-dimensional point of all dimensions to (All-Location, All-Prescriber, Psychotropic, All-Time) and adds the qualification `main-indication = 'anti depression'` upon selecting the role `main-indication` from the properties presented for drug group `Psychotropic` (which corresponds to `psychoactive-drug` in the external drug ontology) in the *Ontology Context View*. For future use, he may define a new concept `AntiDepressionPsychotropic` for this qualification that will be added and correctly placed into the domain ontology by the *Ontology Manager* (If that concept had been defined previously, the business analyst would have seen it in the *Ontology Context View* for possible selection).

After having defined the scope of interest for the measure, the business analyst specifies the *measurement instructions*. He checks the *Measure and Score Ontology Context View* for measures already defined for the scope of interest. In our simple case this view contains the measure `nrOfprescriptions`. He drags this measure into the *Measurement Instructions* section of the *Measure Definition Board* and further specializes the measure by adding qualification `LowPricedDrug` from the *Domain Ontology*. To calculate the percentage, he drags the measure `nrOfPrescriptions` once more into the *Measure Definition Board* and indicates that the value of the main measure is the ratio of both submeasures.

Next, the business analyst defines in the *Score Definition Board* a *score* to be used for comparing (based on the defined measure) a specific group of interest (in the scope of the measure) against some specific group of comparison. In our simple example, the business analyst selects the predefined scoring function `percentageDifference` and scales the score value to `[0,100]`.

The *Judgement Rule Board* will be explained later.

The business analyst moves to the *Cockpit Data Analysis View* (see Fig. 4), that is partitioned into four parts: the *Control Board*, the *Score Board*, the *Measure Board*, and the *Domain Ontology Context View*, to apply measure `percentageOfCheapAntiDepressionDrugs` to a particular group of interest within the scope of the measure. In the *Control Board* the business analyst can narrow down the group of interest as required for his specific analysis. For our analysis task the business analyst moves in the location dimension from `All-Locations` to `Upper-Austria` and in the doctor dimension from `All-Doctors` to `GenPractitioner` to multi-dimensional point (`UpperAustria`, `GenPractitioner`, `Psychotropic`, `All-Time`) and selects additional qualification `medium-to-large-city` from the domain ontology about locations using the *Ontology Context View* (not shown expanded in Fig. 4 but as in Fig. 3).

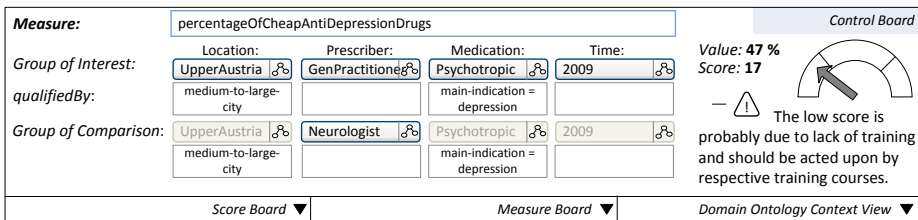


Fig. 4. Cockpit Data Analysis View (simplified; Score Board, Measure Board and Domain Ontology Context View unexpanded)

Scoring, i.e., determining a comparison score of a group of interest against a group of comparison, is performed in basically the same way as outlined above for measuring. The business analyst selects in the *Control Board* a group of comparison that is to be compared against the previously selected group of interest by indicating the variation in the scope of interest through moving up or to siblings in one or several dimensions. In our example, the business analyst compares general practitioners against neurologists. A *score instance* (consisting of measure, group of interest and group of comparison) is generated by the *MDO*

engine (whereby the *Ontology Reasoner* determines the scope of comparison from the scope of interest and the indicated variation), and translated by the *MDO-DWH mapper* into an OLAP query to calculate the score (not shown in Fig. 4).

Typically, the business analyst will perform several single- or multi-variant comparisons, e.g., across location, prescriber, or time, and use the *Score Board* to drill down in scores (e.g., showing how each state scores in a nation-wide comparison) or the *Measure Board* to drill down in measures as known from conventional dashboards. Different to conventional dashboards, however, the *Semantic Cockpit Frontend* may be configured through generic *guidance rules* to automatically drill down in scores (by interacting with the *MDO-Engine*) based on a resulting low or high comparison score, e.g. to state level, if a nationwide comparison of Upper Austria resulted in a striking score.

Our scoring example already gives a possible explanation for the low score of general practitioners against neurologists (lack of training) and recommends an action (offering training courses). This will not be the case when the business analyst encounters such a situation for the first time. Then the business analyst may reason that general practitioners may be less trained than neurologists and might, therefore, differentiate less between descriptions of various kinds of anti depression drugs. For future analysis (in other years or at other locations) the business analyst decides to add a generic judgment rule `recommendTraining` to the knowledge base expressing that low scores of general practitioners in cross-doctor group comparisons to neurologists of prescribing low-priced anti depression drugs may be due to lack of training and should be acted upon by respective training courses . He uses the *Judgment Rule Board* (see Fig. 3) to define that rule, by identifying the *scope of the situation* through a *scope of interest* (in the same way as explained above for measures) and a *scope of application* that indicates the comparison group(s) for which the rule is applicable. In our case, the scope of the situation is given by the multi-dimensional point of the measure schema `percentageOfCheapAntiDepressionDrugs`, (All-Locations, GenPractitioner, Psychotropic, All-Times). The situation should be relevant only for comparisons in the prescriber dimension against neurologists (but not for comparisons in the location or time dimension). The business analyst states this in that he defines the scope of application to be comparisons in the dimension `prescriber` against `Neurologist`. He then defines the condition of the rule to be a score value below 20 and the action. If in later analysis, in another year for the same or a different state, apart from these differences, the same situation occurs as just analyzed, the judgment rules fires and the score will be annotated by the previous finding. For this, the *Cockpit Engine* will forward the situation and comparison at hand to the *Rule Engine*, which will use the *Ontology Reasoner* for classifying the situation and comparison at hand to determine whether the rule is applicable. If it is applicable and the condition holds as well, it will report the action to the *Cockpit Engine*. In a more advanced setting, such a judgment rule may be configured to monitor the DWH continuously or at predefined intervals or predefined events and automatically prepares a report to management, consisting of score and measure diagrams together with recommended actions.

Typical actions of judgment rules are: to explain a striking high or low comparison score; to recommend strategic or operational action upon a low score (such

as initiating a fraud inspection); or to recommend a further problem analysis by experts to find possible drivers for high or low performance against comparison groups. Only the situation that a low or high score is still unexplained, requires further study by the business analyst; thus, he is guided in his analysis by the insight gained so far, shielded from information overload and can concentrate his effort on following up situations unexplained so far.

5 Related Work

The Semantic Cockpit is based on multidimensional modeling at the conceptual level (an issue extensively discussed in the literature, with the *Dimensional Fact Model* [3] being our main reference) enriched by the explicit representation of and reasoning with additional knowledge, such as domain knowledge represented in ontologies, semantics of derived measures and scores, and previous insights represented as judgement rules.

Combining ontologies and multidimensional modeling has been discussed from different perspectives and with different applications in mind: Nebot et al. [5] propose *Multidimensional Integrated Ontologies* as a basis for the multidimensional analysis of Semantic Web data. Romero and Abelló [8] introduce an approach to derive a multidimensional conceptual model starting from domain knowledge represented in ontologies. Khouri and Bellatreche [4] introduce a methodology for designing Data Warehouses from ontology-based operational databases. Niemi and Niinimäki [6] discuss how ontologies can be used for reasoning about summarizability in OLAP. Sell et al. [9] focus on using ontologies as a means to represent data in business terms in order to simplify data analysis and customizing BI applications. Baader and Sattler [1] and Calvanese et al. [2] extend Description-Logics-based ontologies with aggregation functions.

Conceptual modeling and reasoning with (derived) measures and scores has not received much attention so far, however it is related to the work of Pardillo et al. [7], which extends OCL for defining OLAP queries as part of conceptual multidimensional models.

The interplay of rules and ontologies was subject of the REVERSE project (<http://reverse.net>) and is subject of the ongoing ONTORULE project (<http://ontorule-project.eu/>). The representation and reasoning over multi-dimensional situation-condition-action rules representing interesting relationships between compared data, as it is required to represent judgment knowledge in the Semantic Cockpit, has not been researched so far.

6 Conclusion

In this paper we outlined the components and usage of the Semantic Cockpit, which is developed in an ongoing cooperative research project (see footnote on page 1).

We implemented a first proof-of-concept prototype that we use –together with our partners from health insurance industry– to study detailed requirements and performance issues and to further revise and extend the architecture and basic approach presented in this paper.

In the remaining phases of the project till 2013 we will investigate and implement language constructs and reasoning techniques for measure & score ontologies and judgement rules, together with their mapping to the database level. We will investigate the complexity and performance of the different reasoning tasks, the costs of integrating external ontologies, problems with the quality of external ontologies and the possibility to further automate design and analysis tasks.

We expect that the Semantic Cockpit leads to higher-quality analysis results and significant cost savings. We will validate the expected quantitative and qualitative improvements by field studies carried out with end-users from health insurance industry.

References

1. Baader, F., Sattler, U.: Description logics with aggregates and concrete domains. *Inf. Syst.* 28(8), 979–1004 (2003)
2. Calvanese, D., Kharlamov, E., Nutt, W., Thorne, C.: Aggregate queries over ontologies. In: *ONISW 2008: Proceeding of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web*, pp. 97–104. ACM, New York (2008)
3. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: A conceptual model for data warehouses. *Int. J. Cooperative Inf. Syst.* 7(2-3), 215–247 (1998)
4. Khouri, S., Bellatreche, L.: A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In: Song II, Y., Ordóñez, C. (eds.) *DOLAP*, pp. 19–24. ACM, New York (2010)
5. Nebot, V., Llavori, R.B., Pérez-Martínez, J.M., Aramburu, M.J., Pedersen, T.B.: Multidimensional integrated ontologies: A framework for designing semantic data warehouses. *J. Data Semantics* 13, 1–36 (2009)
6. Niemi, T., Niinimäki, M.: Ontologies and summarizability in olap. In: *SAC*, pp. 1349–1353 (2010)
7. Pardillo, J., Mazón, J.-N., Trujillo, J.: Extending ocl for olap querying on conceptual multidimensional models of data warehouses. *Inf. Sci.* 180(5), 584–601 (2010)
8. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. *Data Knowl. Eng.* 69(11), 1138–1157 (2010)
9. Sell, D., da Silva, D.C., Beppler, F.D., Napoli, M., Ghisi, F.B., dos Santos Pacheco, R.C., Todesco, J.L.: Sbi: a semantic framework to support business intelligence. In: Duke, A., Hepp, M., Bontcheva, K., Vilain, M.B. (eds.) *OBI. ACM International Conference Proceeding Series*, vol. 308, p. 11. ACM, New York (2008)

A Model-Driven Approach for Enforcing Summarizability in Multidimensional Modeling

Jose-Norberto Mazón¹, Jens Lechtenbörger², and Juan Trujillo¹

¹ Lucentia, Dept. of Software and Computing Systems, University of Alicante, Spain
{jnmazon,jtrujillo}@dlsi.ua.es

² Dept. of Information Systems, University of Münster, Germany
lechten@wi.uni-muenster.de

Abstract. The development of a data warehouse system is based on a conceptual multidimensional model, which provides a high level of abstraction in the accurate and expressive description of real-world situations. Once this model has been designed, the corresponding logical representation must be obtained as the basis of the implementation of the data warehouse according to one specific technology. However, there is a semantic gap between the dimension hierarchies modeled in a conceptual multidimensional model and its implementation. This gap particularly complicates a suitable treatment of summarizability issues, which may in turn lead to erroneous results from business intelligence tools. Therefore, it is crucial not only to capture adequate dimension hierarchies in the conceptual multidimensional model of the data warehouse, but also to correctly transform these multidimensional structures in a summarizability-compliant representation. A model-driven normalization process is therefore defined in this paper to address this summarizability-aware transformation of the dimension hierarchies in rich conceptual models.

1 Introduction

Business intelligence tools, such as OLAP (On-Line Analytical Processing) tools depend on the multidimensional (MD) structures of a data warehouse that allow analysts to explore, navigate, and aggregate information at different levels of detail to support the decision making process. Current approaches for data warehouse design advocate to start the development by defining a conceptual model in order to describe real-world situations by using MD structures [13]. These structures contain two main elements. On one hand, dimensions which specify different ways the data can be viewed, aggregated, and sorted (e.g., according to time, store, customer, product, etc.). On the other hand, events of interest for an analyst (e.g., sales of products, treatments of patients, duration of processes, etc.) are represented as facts which are described in terms of a set of measures. Every fact is based on a set of dimensions that determine the granularity adopted for representing the fact's measures. Dimensions, in turn, are organized as hierarchies of levels that allow analysts to aggregate data at

different levels of detail. Importantly, dimension hierarchies must ensure summarizability, which refers to the possibility of accurately computing aggregate values with a coarser level of detail from values with a finer level of detail. If summarizability is violated, then incorrect results can be derived in business intelligence tools, and therefore erroneous analysis decisions [3,4]. In addition, summarizability is a necessary precondition for performance optimizations based on pre-aggregation [10].

Usually, summarizability are not solved at the conceptual level, but at late stages of the development by using information from the data contained in the implemented data warehouse [11,9]. This way of proceeding poses problems to data warehouse designers, since great efforts are required to ensure summarizability due to the huge amount of data stored. However, if one tries to ensure summarizability at the conceptual level, one typically obtains MD models that are more difficult to understand as they contain an excessive amount of detail that is not required in the initial design steps. As understandability must be one inherent property of conceptual MD models, designers should be able to specify rich conceptual models without being concerned with summarizability problems. Then, a normalization process should be applied to transform the designed MD model into a constrained conceptual model, which is restricted to those MD structures that do not violate summarizability. This normalized model should also provide a high level of expressiveness in describing real-world situations.

Bearing these considerations in mind, in this paper, we take advantage of model-driven development to propose a comprehensive normalization process by using widely adopted formalisms. In essence, we describe how to (i) design different kinds of dimension hierarchies in a conceptual model in order to easily and understandably represent real-world situations regardless of summarizability problems, and (ii) automatically derive equivalent normalized conceptual models, which are constrained to those kind of dimension hierarchies that do not violate summarizability. From this normalized MD model, an implementation that satisfies summarizability can be easily deployed in any database platform and can be accurately queried by any business intelligence tool.

The rest of this paper is structured as follows. Related work is described in Section 2. Our normalization process for ensuring summarizability in dimension hierarchies is presented in Section 3. Finally, conclusions and potential for future work are provided in Section 4.

2 Related Work

In [9] we present a survey on summarizability issues in MD modeling. Briefly, MD modeling aims to represent measurable facts for real-world events under different perspectives of detail, which are specified via dimensions. For example, using the UML profile for MD modeling proposed in [5], Fig. 1 represents *Sales* via a *Fact* (⊞) class and *Date*, *Product*, and *Customer* via *Dimension* (⌞) classes. Facts are composed of measures or fact attributes, which are represented as attributes with the *FactAttribute* stereotype (FA) such as *Price* and *Quantity* in Fig. 1. Moreover,

dimension levels, which allow to analyze measures at a specific level of detail, are specified by *Base* classes (\overline{B}) such as *Day* and *Week*. Associations (represented by the stereotype *Rolls-UpTo*, \odot) between pairs of *Base* classes form dimension hierarchies. Importantly, *Rolls-UpTo* associations between *Base* classes as well as associations between *Fact* and *Dimension* classes are annotated with UML multiplicities. E.g., the multiplicities for the association between *Country* and *Region* in Fig. 1 indicate that every region belongs to exactly one country (“1” in role *r* at the *Country* end) whereas there are countries (such as “Andorra”, “Monaco”, etc.) without associated regions (“0..*” in role *d* at the *Region* end). Finally, UML generalization relationships between *Base* classes can be used to represent optional dimension levels within a hierarchy.

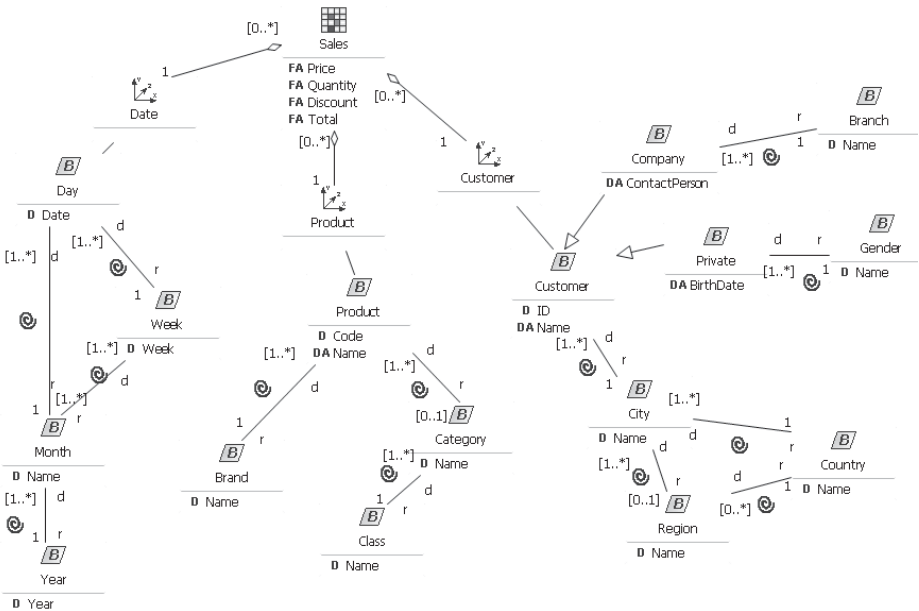


Fig. 1. Sample scenario

In the context of MD data, summarizability presents a guarantee concerning the correctness of aggregated measures. The notion of summarizability was originally introduced in [12] and later characterized in [4]. As observed in [9], minimum and maximum multiplicities of associations characterize sub-structures where summarizability is violated. In earlier work [8], we have shown how to normalize MD schemata with non-summarizable fact-dimension associations into summarizable ones. With respect to non-summarizable *Rolls-UpTo* associations, however, a similar approach still needs to be designed.

Indeed, in a fundamental work Pedersen et al. [10,11] propose instance level algorithms to automatically transform dimension hierarchies to achieve summarizability. As this proposal works at the instance level, it is necessary to transform

the data that will populate the DW, which may involve considerable efforts of preprocessing. In particular, ETL processes become more complex, as summarizability checks must be incorporated and executed for every update. In addition, as the data transformations produce artificial data values, data analysis becomes more complex.

In [6,7] the authors present a classification of different kinds of complex dimension *hierarchies*, and they define the MultiDimER model for the conceptual design of complex MD models based on an extension of the well-known Entity-Relationship (ER) model. The idea is that this classification guides developers to properly capture at a conceptual level the precise semantics of different kinds of hierarchies without being limited by current data analysis tools. Furthermore, the authors discuss how to map these conceptual hierarchies to the relational model (enabling implementation in commercial tools). Unfortunately, the mapping between the conceptual and the logical level is described informally. In addition, the commented mapping is tool-dependent and it may vary depending on the scenario.

3 Model-Driven Development of Dimension Hierarchies

While a conceptual MD model should represent the real world in a way that is easy to understand and that supports discussions with end users, a logical MD model must support the implementation in a chosen target technology in such a way that data can be accurately analyzed. To bridge this semantic gap, we propose a model-driven normalization process to derive an intermediate model, which should still be expressed at the conceptual level (to avoid limitations imposed by particular target models) but where MD structures that violate summarizability conditions are replaced with summarizable alternatives (which are typically more precise as we will see below). In particular, we address the three types of non-summarizable *Rolls-UpTo* associations enumerated in [9]: Non-strict, roll-up incomplete, drill-down incomplete

Non-strict associations are those where the maximum multiplicity at role r is $*$ (instead of 1). For example, the association between *Week* and *Month* in Fig. 1 is non-strict, and requires special care to avoid the well-known double counting problem. Next, an association is *drill-down incomplete* if the minimum multiplicity at role d is 0; otherwise, it is drill-down complete. For example, the association between *Country* and *Region* in Fig. 1 is drill-down incomplete as there are countries (such as “Andorra”, “Monaco”, etc.) without associated regions. As explained in [9] in this case one has to be careful when drilling down from aggregate values at the level *Country* towards the level *Region* as values for countries without regions may not be accounted for, leading to inconsistent grand totals. Finally, an association is *roll-up incomplete* if the minimum multiplicity at role r is 0. For example, the association between *Product* and *Category* in Fig. 1 is roll-up incomplete. As explained in [9] in this case one has to be careful when rolling up from aggregate values at the level *Product* towards the level *Category* as values for products that are not assigned to any category will not be accounted for, leading again to inconsistent grand totals.

In the following subsections, we show how to carry out a normalization process to obtain a conceptual MD model that ensures summarizability while accurately capturing the expressiveness of the demanded real-world situation. This process is composed of a set of transformations to obtain a normalized MD model constrained to hierarchies that contain only those elements and relationships that do not violate summarizability. For the sake of understanding, these transformations are first defined informally and only two of them are formally described in QVT (Query/View/Transformation) due to space constraints.

3.1 Eliminating Non-strictness

Non-strictness is eliminated by one transformation which is based on one QVT relation (named as *nonStrictBases*) that replaces all occurrences of non-strict *Rolls-UpTo* associations in the source model with constructs that do not violate summarizability. The rationale behind this QVT relation is as follows. A non-strict association, i.e., a many-to-many relationship among dimension instances, represents the least restrictive possible case. In fact, even if no association is declared among *Base* classes then adding a non-strict (and incomplete) association does not impose any additional restriction on the instances. E.g., in Fig. 1 every *Product* has a *Brand* and may have a *Category*. As the relationship among *Brand* and *Category* is not declared explicitly, we may reasonably assume to find many *Brands* associated with the *Products* for each *Category* and vice versa. Hence, we may safely add a non-strict association from *Category* to *Brand* without changing the meaning of the model.

More generally, consider a non-strict association between *Base* classes b_1 and b_2 . On the one hand, this association is redundant and can be removed safely if there are alternative strict *Rolls-UpTo* associations to b_2 (which is the *Base* class that causes the non-strictness). On the other, the association cannot be removed if the *Base* class b_2 does not play role r in some strict association. In this latter case, removing the association would isolate b_1 from b_2 . Moreover, in this case the instances of b_2 are in a many-to-many relationship not only with b_1 but also with all *Base* classes that roll-up from (possibly transitively) b_1 . This many-to-many relation is naturally expressed by moving b_2 into a newly created dimension, which again leads to the removal of the non-strict association.

As an example for a many-to-many relation that should be represented via a newly created dimension, assume that *Products* in the scenario of Fig. 1 are books for which the authors are of interest. Then, we may add the *Base* class *Author* along with a non-strict association from *Book* to *Author* (see Fig. 2(a)).

Although such an association allows to represent the relationship between books and their authors, this relationship is unrelated to the sales of individual books. Besides, computing sales volumes per author based on that relationship may easily lead to summarizability problems due to double counting. Hence, our normalization process removes *Author* from the *Fact* class *Sales* and transforms the non-strict association into the natural many-to-many relationship of a newly created fact, in this example into a two-dimensional *Fact* class *NewFact* with *Dimension* classes *Book* and *Author*. As *Dimension* class *Book* is common to

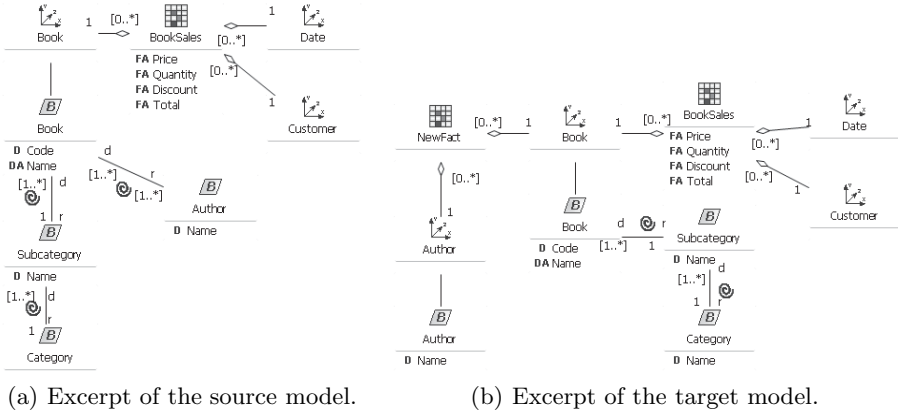


Fig. 2. Newly created dimension for eliminating non-strictness: the Book sales example

both facts, queries involving *Sales* for *Authors* are still possible via drill-across operations. Moreover, to support summarizable queries, the designer may want to manually add additional measures, e.g., each author's share per book, to the new fact schema. The target model is shown in Fig. 2(b).

The *nonStrictBases* relation is shown in Fig. 3(a). It checks that there is a non-strict *Rolls-UpTo* association between two *Base* classes (b_1 and b_2) in the source model (multiplicity $*$ in the role r) in order to create the appropriate elements in the target model to obtain a representation without non-strict associations. The two cases concerning the treatment of non-strict associations explained above are captured by a condition that must hold to execute this relation (see *when* clause of Fig. 3(a)). This condition checks whether *Base* class b_2 plays role r in some strict association in the source model. On the one hand, if this condition does not hold then the QVT relation is not launched, and *no* new *Rolls-UpTo* association is created in the target model, since the two *Base* classes are related via a many-to-many relationship by default. On the other, if the condition is satisfied then b_2 does not play role r in some strict association. In this case, a new *Fact* class f is created which is associated with two new *Dimension* classes: d_1 and d_2 . Specifically, d_1 corresponds to the *Dimension* class related to the *Base* classes b_1 and b_2 of the source model, while d_2 is a new *Dimension* class whose defining *Base* class is b_2p (which is a copy of the *Base* class that causes the non-strictness in the source model b_2).

After the execution of the *nonStrictBases* relation, an excerpt of the target model related to the *Date* dimension is shown in Fig. 4(a). Here, the non-strict *Rolls-UpTo* association between *Month* and *Week* has disappeared, since in this way both *Bases* classes are already many-to-many related, and there is no need for the creation of a new *Fact* class. The remaining parts of the source model that do not present non-strictness problems are copied directly to the target model. It is worth noting that all the possible strict *Rolls-UpTo* association between *Base* classes are assumed to be explicitly modeled in the source model and they will

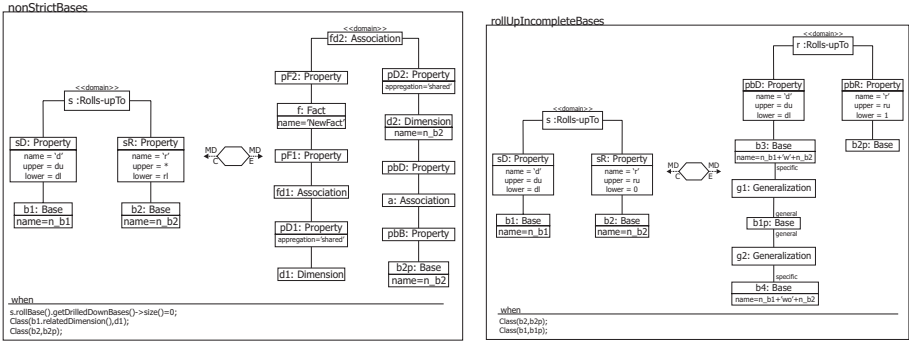


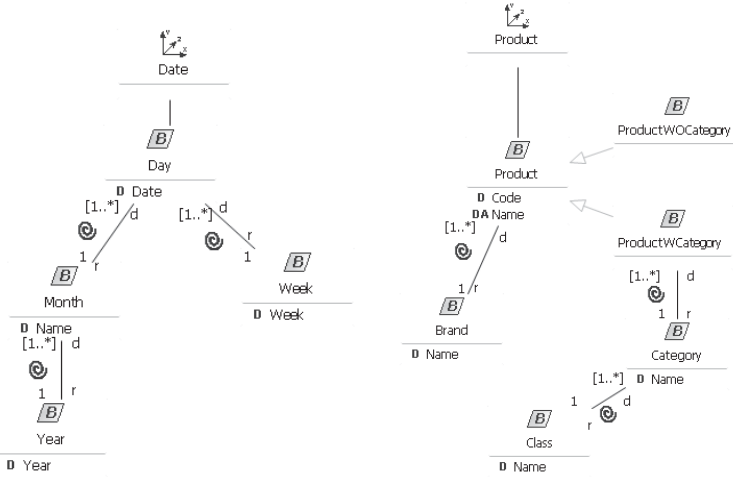
Fig. 3. QVT relations for normalizing hierarchies

also appear in the target model. In this way, analysis capabilities of the source model are not negatively affected. Anyway, if users would still like to navigate between aggregate values by means of a non-strict relationship, then a new type of association could be added to the conceptual model; such “navigational” arcs, however, would need special support from implementing OLAP tools and are beyond the scope of this paper.

With respect to the soundness of the *nonStrictBases* relation shown in Fig. 3(a) we focus on the non-strict association of the source model, say association *s* between *b*₁ and *b*₂, which is represented differently in the target model. We consider two cases: First, the QVT relation just removes the non-strict association. In this case, the *when* clause of the relation assures that *b*₂ is alternatively reachable via strict *Rolls-UpTo* associations. Hence, in the target model both *Base* classes still belong to a common dimension hierarchy, where all pairs of *Base* classes are related via many-to-many relationships by default as we have explained above, and there is no loss of information when removing the non-strict association. Second, the QVT relation creates a new *Fact* class to represent the many-to-many relationship. In this case, the associations between instances of *b*₁ and *b*₂ of the source model are simply stored as instances of the new *Fact* class in the target model. Hence, the information that can be represented under the source model can also be represented under the target model.

3.2 Eliminating Roll-Up Incompleteness

In line with the general analysis concerning optional properties for conceptual design in [1] we argue that multiplicities of 0 should be avoided in *understandable* conceptual models whenever possible. Instead, we advocate to apply generalization for conceptual MD modeling as well. To eliminate roll-up incompleteness, the transformation is based on one QVT relation that replaces all occurrences of roll-up incomplete *Rolls-UpTo* associations in the source model with generalization constructs. In this case, the optional *Base* class (which has multiplicity 0 at the role *r*) should be associated with a suitable sub-class in a generalization



(a) Normalized *Date* dimension. (b) Normalized *Product* dimension.

Fig. 4. Examples of normalized hierarchies

between *Base* classes: one to reflect instances with the optional property and the other one to reflect instances without that property.

The corresponding QVT relation (*rollUpIncompleteBases*) is shown in Fig. 3(b). It checks roll-up incompleteness in the *Rolls-UpTo* association between *Base* classes in the source model. Specifically, if a 0 multiplicity is detected in the role r of a *Rolls-UpTo* association s between two *Base* classes b_1 (e.g., *Product* in Fig. 1) and b_2 (e.g., *Category* in Fig. 1), then the relation enforces the creation of new elements in the target model as follows: Two new *Base* classes b_{1p} and b_{2p} that correspond to the source *Base* classes b_1 and b_2 , respectively. In addition, two new *Base* sub-classes of b_{1p} , namely b_3 and b_4 , are created via new generalization relationships g_1 and g_2 . Here, b_3 reflects the instances of its super-class b_{1p} that are associated with some instance of the optional *Base* class b_{2p} , and b_4 reflects the remaining instances of b_{1p} . Furthermore, the roll-up incomplete association s between b_1 and b_2 is replaced with a roll-up complete association r between b_3 and b_{2p} .

After the execution of the *rollUpIncompleteBases* relation, an excerpt of the target model related to the *Product* dimension is shown in Fig. 4(b). Here, two new *Base* classes, *ProductWCategory* and *ProductWOCategory*, are created to reflect those products that belong to a category or not, respectively. Again, those parts of the source model that do not present roll-up incompleteness problems are copied directly to the target model.

With respect to the soundness of the *rollUpIncompleteBases* relation, we focus on the roll-up incomplete association of the source model, say association s between b_1 in the role d and b_2 in the role r , which is represented differently in the target model. First, b_1 and b_2 are still present in the target model (as b_{1p} and b_{2p}). Moreover, if an instance of b_1 is not related to any instance of b_2

(which exemplifies the incompleteness of the association) then this instance is simply stored as an instance of the same class under the target model (i.e. b_1p) and as an instance of its subclass b_4 . If, however, an instance i_1 of b_1 is related to some instance i_2 of b_2 in the source model, then in the target model i_1 will be an instance of b_1p and simultaneously an instance of b_1p 's sub-class b_3 . Now, this sub-class b_3 has a roll-up complete association with b_2p , which allows to retrieve the associated instance i_2 in the target model. Hence, the information that can be represented under the source model can also be represented under the target model.

3.3 Eliminating Drill-Down Incompleteness

The transformation that eliminates drill-down incompleteness is based on one QVT relation that replaces all occurrences of drill-down incomplete *Rolls-UpTo* associations with normalized elements. Due to space constraints this transformation is not shown, but the rationale of this QVT relation is removing summarizability problems by using generalization constructs. In this way, the optional *Base* class (which has multiplicity 0 at the role d , e.g., *Region* in Fig. 1) should be associated with a sub-class in a generalization between *Base* classes.

4 Conclusions and Future Work

Data warehouse designers have to make a great effort in defining dimension hierarchies that accurately reflect real-world situations in a MD model, whilst summarizability problems are avoided. In this paper, we have described a normalization approach for ensuring that the implemented MD model will be queried without the summarizability problems derived from the dimension hierarchies. Following our approach, designers can define dimension hierarchies that violate summarizability conditions in a conceptual model by using our UML profile. This conceptual model reflects real-world situations in an understandable way. Later, several transformations can be applied to automatically obtain a normalized MD model whose dimension hierarchies do not allow situations that attempt against summarizability, thus avoiding erroneous analysis of data.

Finally, we remark that the multidimensional normal forms defined in [2], which formalize quality aspects of MD models, deal with schemata with an expressiveness that is similar to the one of our normalized models. However, so far there is no work that considers the definition of such normal forms for semantically rich conceptual models. As an avenue for future work it appears attractive to define normal forms at the level of our normalized model: The generalization constructs included in this level enable a simplified treatment of optional levels (in [2], generalization is “simulated” by a careful application of context dependencies).

Our planned future work consists of evaluating and giving mechanisms to deal with the notion of “type compatibility” for summarizability of [4], which checks that the statistical function associated with the measure is summarizable according to the type of the measure and the type of the related dimensions.

Acknowledgments. This work has been partially supported by the following projects: SERENIDAD (PEII-11-0327-7035) from Junta de Comunidades de Castilla-La Mancha (Spain) and by the MESOLAP (TIN2010-14860) project from the Spanish Ministry of Education and Science.

References

1. Bodart, F., Patel, A., Sim, M., Weber, R.: Should optional properties be used in conceptual modelling? a theory and three empirical tests. *Info. Sys. Research* 12(4), 384–405 (2001)
2. Lechtenbörger, J., Vossen, G.: Multidimensional normal forms for data warehouse design. *Inf. Syst.* 28(5), 415–434 (2003)
3. Lehner, W., Albrecht, J., Wedekind, H.: Normal forms for multidimensional databases. In: Rafanelli, M., Jarke, M. (eds.) *SSDBM*, pp. 63–72. IEEE Computer Society, Los Alamitos (1998)
4. Lenz, H.J., Shoshani, A.: Summarizability in OLAP and statistical data bases. In: Ioannidis, Y.E., Hansen, D.M. (eds.) *SSDBM*, pp. 132–143. IEEE Computer Society, Los Alamitos (1997)
5. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. *Data Knowl. Eng.* 59(3), 725–769 (2006)
6. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data Knowl. Eng.* 59(2), 348–377 (2006)
7. Malinowski, E., Zimányi, E.: *Advanced data warehouse design: From conventional to spatial and temporal applications*. Springer, Heidelberg (2008)
8. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: Solving summarizability problems in fact-dimension relationships for multidimensional models. In: Song, I.Y., Abelló, A. (eds.) *DOLAP*, pp. 57–64. ACM, New York (2008)
9. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modeling. *Data Knowl. Eng.* 68(12), 1452–1469 (2009)
10. Pedersen, T.B., Jensen, C.S., Dyreson, C.E.: Extending practical pre-aggregation in on-line analytical processing. In: *VLDB*, pp. 663–674 (1999)
11. Pedersen, T.B., Jensen, C.S., Dyreson, C.E.: A foundation for capturing and querying complex multidimensional data. *Inf. Syst.* 26(5), 383–423 (2001)
12. Rafanelli, M., Shoshani, A.: *STORM: A statistical object representation model*. In: Michalewicz, Z. (ed.) *SSDBM 1990*. LNCS, vol. 420, pp. 14–29. Springer, Heidelberg (1990)
13. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: Song, I.Y., Vassiliadis, P. (eds.) *DOLAP*, pp. 3–10. ACM, New York (2006)

Repairing Dimension Hierarchies under Inconsistent Reclassification

Mónica Caniupán¹ and Alejandro Vaisman^{2,3}

¹ Universidad del Bío-Bío, Chile
mcaniupa@ubiobio.cl

² Universidad de la República, Uruguay
avaisman@fing.edu.uy

³ Université Libre de Bruxelles

Abstract. On-Line Analytical Processing (OLAP) dimensions are usually modelled as a hierarchical set of categories (the *dimension schema*), and *dimension instances*. The latter consist in a set of elements for each category, and relations between these elements (denoted *rollup*). To guarantee summarizability, a dimension is required to be *strict*, that is, every element of the dimension instance must have a unique ancestor in each of its ancestor categories. In practice, elements in a dimension instance are often reclassified, meaning that their rollups are changed (e.g., if the current available information is proved to be wrong). After this operation the dimension may become non-strict. To fix this problem, we propose to compute a set of *minimal r-repairs* for the new non-strict dimension. Each minimal *r-repair* is a *strict* dimension that keeps the result of the reclassification, and is obtained by performing a minimum number of insertions and deletions to the instance graph. We show that, although in the general case finding an *r-repair* is NP-complete, for real-world dimension schemas, computing such repairs can be done in polynomial time. We present algorithms for this, and discuss their computational complexity.

1 Introduction

Data Warehouses (DWs) integrate data from different sources, also keeping their history for analysis and decision support [1]. DWs represent data according to *dimensions* and *facts*. The former are modeled as hierarchies of sets of elements (called *dimension instances*), where each element belongs to a category from a hierarchy, or lattice of categories (called a *dimension schema*). Figure 1(a) shows the dimension schema of a *Phone Traffic DW* designed for an online Chilean phone call company, with dimensions Time and Phone (complete example in [2]). Figure 1(b) shows a dimension *instance* for the Phone schema. Here, TCH (Talcahuano), TEM (Temuco) and CCP (Concepción) are elements of the category City, and IX and VIII are elements of Region. The facts stored in the Phone Traffic DW correspond to the number of incoming and outgoing calls of a phone number at a given date. The fact table is shown in Figure 1(c).

To guarantee summarizability [3,4], a dimension must satisfy some constraints. First, it must be *strict*, that is, every element of a category should reach (i.e., roll-up to) no more than one element in each ancestor category (for example, the dimension instance in Figure 1(b) is strict). Second, it must be *covering*, meaning that every member of a dimension level rolls-up to some element in another dimension level. Strict and

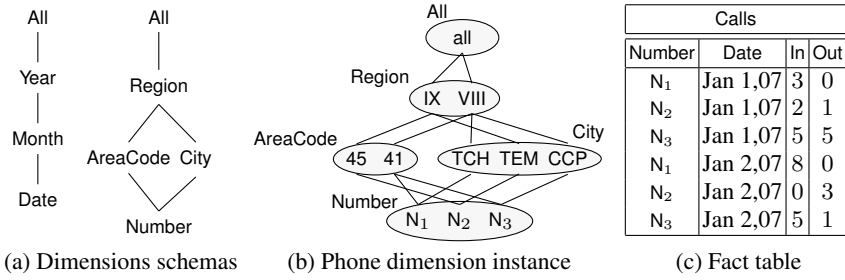


Fig. 1. The Phone Traffic DW (c.f. [2])

covering dimensions allow to pre-compute aggregations, which can be used to compute aggregations at higher category levels, increasing query answering efficiency [5].

Dimensions can be updated to adapt to changes in data sources or modifications to the business rules [6,7], or even due to errors in the original data. Since strictness is not enforced in current commercial DW systems, after a dimension update is performed, a dimension instance may become non-strict. Considering that organizational DWs can contain terabytes of data [1], ensuring strictness of dimensions is crucial for efficient query answering. As an example, suppose that in Figure 1, the phone number N_2 was incorrectly classified as belonging to Temuco (TEM) while in fact it belonged to Concepcion (CCP). The DW administration must re-assign N_2 to the correct city, an operation we denote *Reclassification*. In the presence of strictness and covering constraints, assigning N_2 to Concepcion makes the dimension to become non-strict, since N_2 still has 45 as its area code. Thus, while summarizing fact data, we can reach region IX if we choose the path number→AreaCode→Region, or region VIII if the path is number→city→Region. To fix this inconsistency, a solution could be to move N_2 to the area code 41. There are, however, situations where the solution is not obvious, as we show in this paper.

At least two approaches to the reclassification problem could be followed: (a) To prevent reclassification if we know that it can lead to a non-strict dimension [7]. We denote *this consistent reclassification*. (b) To allow any reclassification and *repair* the updated dimension if it becomes non-strict. Note that, in the general case, the repair may undo the reclassification, in particular if we are looking for a minimal repair (i.e., the repair ‘closest’ to the original dimension, cf. Section 3). In this paper we study approach (b), which captures the case when the user defines a reclassification about which he/she is absolutely sure. Therefore, the new rollup must be taken for certain, and a dimension instance, consistent with a set of constraints, must be produced by means of a (possibly minimal) repair that keeps the reclassification. We refer to these repairs as *r-repairs*. The *r-repaired* dimension instance is obtained from \mathcal{D} by performing insertions and deletions of edges between elements. We show that in the general case, finding a *minimal r-repair* is NP-hard (Section 3). We also present an algorithm (Section 4) that obtains an *r-repair* in polynomial time for the class of dimensions that contain at most one conflicting level [6] (intuitively, a dimension that can lead to non-strict paths), and study complexity issues.

2 Background and Data Model

A dimension schema \mathcal{S} consists of a pair (\mathcal{C}, \nearrow) , where \mathcal{C} is a set of *categories*, and \nearrow is a child/parent relation between categories. The dimension schema can be also represented with a directed acyclic graph where the vertices correspond to the categories and the edges to the child/parent relation. The transitive and reflexive closure of \nearrow is denoted by \nearrow^* . *There are no shortcuts in the schemas*, that is, if $c_i \nearrow c_j$ there is no category c_k such that $c_i \nearrow^* c_k$ and $c_k \nearrow^* c_j$. Every dimension schema contains a distinguished top category called All which is reachable from all other categories, i.e. for every $c \in \mathcal{C}$, $c \nearrow^* \text{All}$. The leaf categories are called bottom categories. \square

A dimension instance \mathcal{D} over a dimension schema $\mathcal{S} = (\mathcal{C}, \nearrow)$ is a tuple $(\mathcal{M}, <)$, such that: (i) \mathcal{M} is a finite collection of ground atoms of the form $c(a)$ where $c \in \mathcal{C}$ and a is a constant. If $c(a) \in \mathcal{M}$, a is said to be an element of c . The constant all is the only element in category All . Categories are assumed to be disjoint, i.e. if $c_i(a), c_j(a) \in \mathcal{M}$ then $i = j$. There is a function Cat that maps elements to categories so that $\text{Cat}(a) = c_i$ if and only if $c_i(a) \in \mathcal{M}$. (ii) The relation $<$ contains the child/parent relationships between elements of different categories, and is compatible with \nearrow : If $a < b$, then $\text{Cat}(a) \nearrow \text{Cat}(b)$. We denote $<^*$ the reflexive and transitive closure of $<$.

In what follows we use the term dimension to refer to dimension instances. For each pair of categories c_i and c_j such that $c_i \nearrow c_j$ there is a *rollup* relation denoted $\mathcal{R}_{\mathcal{D}}(c_i, c_j)$ that consists of the following set of pairs $\{(a, b) \mid \text{Cat}(a) = c_i, \text{Cat}(b) = c_j \text{ and } a <^* b\}$. We next define two kinds of constraints a dimension should satisfy.

Definition 1. [Strictness and Covering Constraints [8]] A *strictness constraint* over a hierarchy schema $\mathcal{S} = (\mathcal{C}, \nearrow)$ is an expression of the form $c_i \rightarrow c_j$ where $c_i, c_j \in \mathcal{C}$ and $c_i \nearrow^* c_j$. The dimension \mathcal{D} satisfies the strictness constraint $c_i \rightarrow c_j$ if and only if the rollup relation $\mathcal{R}_{\mathcal{D}}(c_i, c_j)$ is strict. A *covering constraint* over \mathcal{S} is an expression of the form $c_i \Rightarrow c_j$ where $c_i, c_j \in \mathcal{C}$ and $c_i \nearrow^* c_j$. The dimension \mathcal{D} satisfies the covering constraint $c_i \Rightarrow c_j$ if and only if the rollup relation $\mathcal{R}_{\mathcal{D}}(c_i, c_j)$ is covering. $\Sigma_s(\mathcal{S})$ and $\Sigma_c(\mathcal{S})$ denote, respectively, the set of all possible strictness and covering constraints over hierarchy schema \mathcal{S} . \square

A dimension \mathcal{D} over an schema \mathcal{S} is *inconsistent* if it fails to satisfy $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$. A dimension \mathcal{D} is said to be *strict (covering)* if all of its rollup relations are strict (covering) [3]. Otherwise, the dimension is said to be *non-strict (non-covering)*. The dimension in Figure 1(b) is both covering and strict. Thus, a query asking for the number of calls grouped by Region can be computed by using pre-computed aggregations at categories AreaCode or City. This is not the case of dimension \mathcal{D} in Figure 2, which is covering, but not strict (N_3 rolls-up to both IX and VIII in category Region).

Most research and industrial applications of DWs assume strict and covering dimensions [19,10]. For these kinds of dimensions, summarization operations between two categories that are connected in the schema are always correct [3]. In practice, dimensions may be non-strict and non-covering [11,12], which can lead to incorrect results when summarizing data. We can prevent these errors specifying integrity constraints for rollup relations [11].

¹ To simplify the presentation and without loss of generality, we assume that categories do not have attributes, and schemas have a unique bottom category.

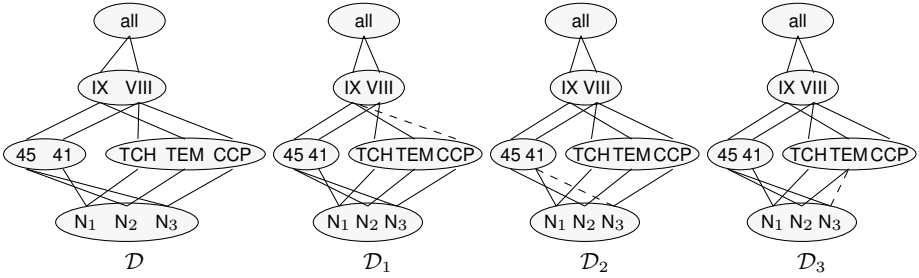


Fig. 2. (Non-strict dimension instance (category names are omitted)(\mathcal{D}); Repairs of the dimension (dashed edges were inserted to restore strictness) (\mathcal{D}_1 - \mathcal{D}_3)

3 Inconsistent Reclassification and r-Repairs

The reclassify operator presented in [7] is not defined for every possible dimension. It allows to reclassify edges of dimensions only if the resulting dimension satisfies $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$. When the dimension contains a *conflicting level* (Definition 2), a reclassification may leave the dimension inconsistent with respect to strictness, and therefore the operation is forbidden.

Definition 2. [Conflicting Levels (cf. [13])] Given a dimension instance $\mathcal{D} = (\mathcal{M}, <)$ over a schema $\mathcal{S} = (\mathcal{C}, \nearrow)$, a pair of categories c_i and c_j such that $c_i \nearrow c_j$, a pair of elements a, b with $\text{Cat}(a) = c_i$ and $\text{Cat}(b) = c_j$. A category c_k such that $c_j \nearrow^* c_k$ is *conflicting* with respect to reclassification, if there exists a category c_m such that $c_m \nearrow^* c_i$, there is an alternative path between c_m and c_k not including the edge (c_i, c_j) , and a is reached by at least one element in c_m . \square

As an illustration, category Region in Figure 1(a) is conflicting with respect to reclassification from AreaCode to Region or a reclassification from City to Region. Consider a reclassification from AreaCode to Region, in this case, $c_i = \text{AreaCode}$, $c_j = c_k = \text{Region}$, and $c_m = \text{Number}$. The edges (Number, City) and (City, Region) form an alternative path from Number to Region not including edge (AreaCode, Region).

In practice, preventing reclassification is not an admissible solution (eg., if the operation is applied to correct erroneous data). Therefore, non-strict dimensions resulting from the operation must be allowed. Definition 3 captures this semantics.

Definition 3. [Reclassify] Given a dimension instance $\mathcal{D} = (\mathcal{M}, <)$ over a dimension schema $\mathcal{S} = (\mathcal{C}, \nearrow)$, a pair of categories c_i and c_j such that $c_i \nearrow c_j$, a pair of elements a, b with $\text{Cat}(a) = c_i$ and $\text{Cat}(b) = c_j$, operator $\text{Reclassify}(\mathcal{D}, c_i, c_j, a, b)$ returns a new dimension instance $\mathcal{D}_u = (\mathcal{M}_u, <_u)$ with $\mathcal{M}_u = \mathcal{M}$, and $<_u = (< \setminus \{(a, c) \mid (a, c) \in < \text{ and } \text{Cat}(c) = c_j\}) \cup \{(a, b)\}$. \square

For example, the result of applying $\text{Reclassify}(\mathcal{D}, \text{Number}, \text{AreaCode}, N_3, 45)$ on dimension \mathcal{D} in Figure 1(b) is the non-strict dimension \mathcal{D} in Figure 2. Since we know that the area code for N_3 is 45, we must perform the reclassification and, in a subsequent step, repair the updated dimension in order to comply with the strictness constraint.

Thus, if a reclassification according to Definition 3 yields a non-strict dimension, we must *repair* the dimension [8], i.e., perform the necessary changes in order to obtain

dimension satisfying the constraints. In particular, from all possible repairs we are interested in finding the *minimal* one. Intuitively, a minimal repair of a dimension is a new dimension that satisfies a given set of constraints, and is obtained by applying a minimum number of insertions and deletions to the original rollup relations. Although techniques to compute repairs with respect to a set of constraints are well-known in the field of relational databases [14], they cannot be applied in a DW setting, since it is not trivial to represent strictness or covering constraints using relational constraints like functional dependencies [8].

Defining a minimal repair requires a notion of *distance* between dimensions: Let $\mathcal{D} = (\mathcal{M}, <_{\mathcal{D}})$ and $\mathcal{D}' = (\mathcal{M}, <_{\mathcal{D}'})$ be two dimensions over the same schema $\mathcal{S} = (\mathcal{C}, \nearrow)$. The distance between \mathcal{D} and \mathcal{D}' is defined as $dist(\mathcal{D}, \mathcal{D}') = |(<_{\mathcal{D}'} \setminus <_{\mathcal{D}}) \cup (<_{\mathcal{D}} \setminus <_{\mathcal{D}'})|$, i.e. the cardinality of the symmetric difference between the two roll-up relations [8]. Based on this, the definition of repair is as follows [8]: Given a dimension $\mathcal{D} = (\mathcal{M}, <)$ over a schema $\mathcal{S} = (\mathcal{C}, \nearrow)$, and a set of constraints $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$, a repair of \mathcal{D} with respect to Σ is a dimension $\mathcal{D}' = (\mathcal{M}', <')$ over \mathcal{S} , such that \mathcal{D}' satisfies Σ , and $\mathcal{M}' = \mathcal{M}$. A *minimal repair* of \mathcal{D} is a repair \mathcal{D}' such that $dist(\mathcal{D}, \mathcal{D}')$ is minimal among all the repairs of \mathcal{D} .

Minimal repairs according to [8] may result in dimensions where the reclassification is undone. If we assume that a reclassification always represents information that is certain, this semantics is not appropriate. For example, the minimal repairs (as defined in [8]) for dimension \mathcal{D} in Figure 2 are dimensions \mathcal{D}_1 , \mathcal{D}_2 and \mathcal{D}_3 . All of them were obtained from \mathcal{D} by performing insertions and/or deletions of edges. For example, \mathcal{D}_1 is generated by deleting edge (CCP,VIII) and inserting (CCP,IX). The distances between the repairs and \mathcal{D} are: (a) $dist(\mathcal{D}, \mathcal{D}_1) = |(CCP,IX), (CCP,VIII)| = 2$; (b) $dist(\mathcal{D}, \mathcal{D}_2) = |(N_3,41), (N_3,45)| = 2$; (c) $dist(\mathcal{D}, \mathcal{D}_3) = |(N_3,TEM), (N_3,CCP)| = 2$. Note that dimension \mathcal{D}_2 has undone the reclassification, and should not be considered an appropriate repair.

In light of the above, we next study the problem of finding repairs that do not undo the reclassification. We denote these repairs *r-repairs*.

3.1 r-Repairs

Given a dimension \mathcal{D} , a set \mathcal{R} of reclassify operations is called *valid* if the following holds: for every $Reclassify(\mathcal{D}, c_i, c_j, a, b) \in \mathcal{R}$ there is no $Reclassify(\mathcal{D}, c_i, c_j, a, c) \in \mathcal{R}$ with $c \neq b$. In what follows we consider only valid reclassification.

Definition 4. [r-repairs] Given a dimension $\mathcal{D} = (\mathcal{M}, <)$ defined over a schema $\mathcal{S} = (\mathcal{C}, \nearrow)$ that is consistent with respect to $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$, and a valid set of reclassifications \mathcal{R} , an *r-repair* for \mathcal{D} under \mathcal{R} is a dimension $\mathcal{D}' = (\mathcal{M}', <')$ defined over \mathcal{S} such that: (i) \mathcal{D}' satisfies Σ ; (ii) $\mathcal{M}' = \mathcal{M}$; and (iii) for every $Reclassify(\mathcal{D}, c_i, c_j, a, b) \in \mathcal{R}$, the pair $(a, b) \in <'$. Let $r-repair(\mathcal{D}, \mathcal{R})$ be the set of *r-repairs* for \mathcal{D} under \mathcal{R} . A *minimal r-repair* \mathcal{D}' for a dimension \mathcal{D} and a set of reclassifications \mathcal{R} is an *r-repair* such that $dist(\mathcal{D}, \mathcal{D}')$ is minimal among all the *r-repairs* in $r-repair(\mathcal{D}, \mathcal{R})$. \square

Note that the distance between a dimension \mathcal{D} and an *r-repair* \mathcal{D}' is bounded by the size of \mathcal{D} . For the dimension in Figure 1(b) and the set $\mathcal{R} = \{Reclassify(\mathcal{D}, Number, AreaCode, N_3, 45)\}$, $r-repair(\mathcal{D}, \mathcal{R})$ contains dimensions \mathcal{D}_1 and \mathcal{D}_3 in Figure 2

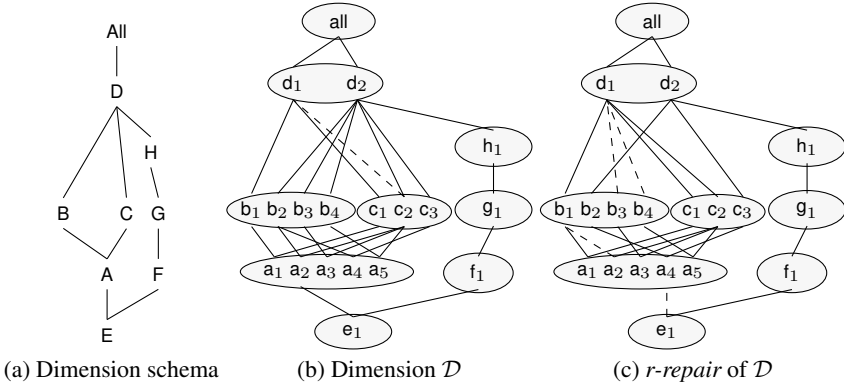


Fig. 3. Heuristics

The existence of *r-repairs* cannot be guaranteed if the reclassification set contains more than one reclassify operation. However, a positive result can be obtained for a set \mathcal{R} containing only one reclassification.

Theorem 1. Let \mathcal{D} be a dimension instance over a schema \mathcal{S} with constraints $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$, and a reclassification set \mathcal{R} applied over \mathcal{D} , producing a new dimension \mathcal{D}_u ; the problem of deciding if there exists an *r-repair* \mathcal{D}' of \mathcal{D}_u with respect to Σ , such that $dist(\mathcal{D}_u, \mathcal{D}') \leq k$ is NP-complete. \square

Theorem 2. Given a dimension \mathcal{D} over a schema \mathcal{S} with constraints $\Sigma = \Sigma_s(\mathcal{S}) \cup \Sigma_c(\mathcal{S})$, and a set of reclassification \mathcal{R} , the problem of deciding if \mathcal{D}' is a minimal *r-repair* of \mathcal{D} with respect to Σ is co-NP-complete. \square

The proof of Theorem 1 proceeds by reduction from the set covering problem. Theorem 2 follows from considering the complement of the problem, that is, deciding whether or not the *r-repair* is minimal with respect to Σ and \mathcal{R} which, from Theorem 1 is NP-complete.

4 Algorithms for *r-Repairs*

We study *r-repairs* for dimensions containing *at most one conflicting level* (Definition 2) that becomes inconsistent with respect to strictness after a reclassify operation. Solving this problem efficiently we cover a wide range of real-life situations [9,10]. We present two heuristics and algorithms that find an *r-repair* for a dimension under reclassification in polynomial time. These heuristics are such that the distance between the *r-repair* obtained and the minimal *r-repair* is bound. In what follows we assume that after a reclassification a dimension becomes inconsistent with respect to strictness. Thus, we can identify the paths that make the dimension non-strict. We denote these paths *non-strict*.

The first heuristic we propose ensures that when choosing a repair operation we do not generate new non-strict paths. Let us consider the dimension schema and instance in Figure 3 (a)-(b). Element c_2 in category C is reclassified from d_2 to d_1 in category D as indicated in Figure 3 (b) (dashed edge). The elements incident to c_2 are a_2, a_3 , and

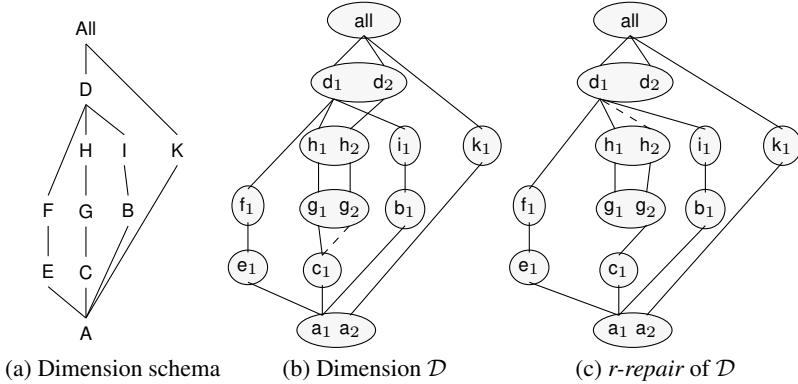


Fig. 4. The dimension schema, instance and r -repair for the Algorithm

a_5 . The non-strict paths are: (i) $a_2 \rightarrow b_2 \rightarrow d_2$, $a_2 \rightarrow c_2 \rightarrow d_1$. (ii) $a_3 \rightarrow b_3 \rightarrow d_2$, $a_3 \rightarrow c_2 \rightarrow d_1$. (iii) $a_5 \rightarrow b_4 \rightarrow d_2$, $a_5 \rightarrow c_2 \rightarrow d_1$. Elements b_3 and b_4 in category B have no incident edges other than a_3 and a_5 , respectively. Thus, a possible repair operation would delete edges (b_3, d_2) and (b_4, d_2) , and insert edges (b_3, d_1) and (b_4, d_1) (dashed edges in Figure 3(c)). This repair operation *does not add new non-strict paths*. Conversely, b_2 has two incident edges (from a_2 and a_4), and a_4 rolls-up to d_2 via c_3 ; therefore, deleting (b_2, d_2) and inserting (b_2, d_1) , would produce a violation of strictness, and the following new non-strict paths: $a_4 \rightarrow b_2 \rightarrow d_1$, $a_4 \rightarrow c_3 \rightarrow d_2$. A solution would be a repair that changes the parent of a_2 either to b_1 (as depicted in Figure 3(c)), to b_3 , or to b_4 . In addition, all edges incident to a_2 must also be reclassified if they reach d_2 through a path not including the edge (A,B) in the dimension hierarchy. This is the case of e_1 that can be repaired by deleting edge (h_1, d_2) and inserting (h_1, d_1) or by moving e_1 to any element a_i reaching d_2 , like a_4 , as we show in Figure 3(c).

The second heuristic is aimed at guaranteeing that at each step the algorithm chooses a repair operation that, accomplishing the “no new conflicts” heuristics, requires the least number of changes.

4.1 Computing the r -Repairs

We illustrate the algorithms with the dimension schema and instance in Figure 4(a)-(b). The first algorithm *search_path()* reads the dimension schema from a table storing the child/parent relation between categories, obtains the conflicting level (CL) and stored it in variable *cat_cl*, and verifies that the schema has a unique such CL. It also computes the category from where the paths reaching the CL start, and stores it in the variable *cat_bottom*. For instance, for the child/parent relation of the categories in Figure 4(a), variable *cat_cl*=D since D is the CL, and the *cat_bottom* = A. Then, the algorithm computes the paths of categories that reach the CL, and stores them in the structure *cat_list_paths*. Any update operation involving categories in this structure may leave the dimension schema non-strict. For the schema in Figure 4, *cat_list_paths* contains: [1]: $A \rightarrow E \rightarrow F \rightarrow D$. [2]: $A \rightarrow C \rightarrow G \rightarrow H \rightarrow D$. [3]: $A \rightarrow B \rightarrow I \rightarrow D$. Then, Algorithm *search_repair()* in Figure 5 applies repair operations over categories in this structure.

Algorithm *search_repair()* (Figure 5) first verifies if an update operation leaves the dimension instance non-strict. For this, the list *list_inconsistent_paths* containing the non-strict paths is produced (line 2). If the dimension is non-strict, the algorithm restores consistency of every non-strict path on the list. Let *Reclassify*($\mathcal{D}, c_1, c_2, e_u, p_u$) be the update operation that affects categories in *cat_list_paths* and leaves the dimension non-strict. The algorithm obtains, for every *cat_bottom* element in the inconsistent list, the number of paths reaching *new_CL* (new parent of e_u in CL) and *old_CL* (old parent in CL) (Lines 5 to 10). If these numbers are equal, it means that there are only two alternative paths for the corresponding *cat_bottom* element, and the algorithm tries to keep *new_CL* as the ancestor to these paths in the CL (lines 12 to 21). If not, it means that there are more paths reaching *old_CL*, and the algorithm tries to update the edge reaching *new_CL*, since this produces less changes (lines 25 to 26). If not, the algorithm assigns *new_CL* to all the paths reaching *old_CL* (lines 28 to 31).

As an illustration, consider the reclassification *Reclassify*($\mathcal{D}, c, g, c_1, g_2$) applied over the dimension in Figure 4(b). The reclassification affects to the bottom element a_1 and therefore *list_inconsistent_paths* contains the paths: [1]: $a_1 \rightarrow e_1 \rightarrow f_1 \rightarrow d_1$. [2]: $a_1 \rightarrow c_1 \rightarrow g_2 \rightarrow h_2 \rightarrow d_2$. [3]: $a_1 \rightarrow b_1 \rightarrow i_1 \rightarrow d_1$. The old and new parent in CL for a_1 are: *old_CL* = d_1 , *new_CL* = d_2 , and the number of paths reaching d_1 and d_2 are, respectively, 2 and 1. Since there are more paths reaching the old parent in CL, the algorithm tries to keep d_1 as the ancestor in \mathcal{D} for all the conflicting paths. This operation is possible given that element h_2 does not have other child different from g_2 , and also the update is not performed over h_2 (validations performed by function *check_change_to_new_CL*); thus, the algorithm deletes edge (h_2, d_2) and inserts (h_2, d_1) (function *change_new_CL*), producing the repair shown in Figure 4(c).

Proposition 1. Given a dimension \mathcal{D} over a schema \mathcal{S} , and a set of reclassify operations \mathcal{R} of size 1. (a) Algorithm *search_repair()* terminates in a finite number of steps. (b) Algorithm *search_repair()* finds an *r-repair* for dimension \mathcal{D} .

4.2 Complexity Analysis

Algorithm *search_path()*, that captures the hierarchy schema of a dimension, runs in $O(k)$ with k being the number of paths reaching the conflicting level and starting at the bottom category. For Algorithm *search_repair()*, the most expensive function is *check_Consistency()*, that finds out if the dimension instance becomes inconsistent after an update, and, in that case, generates the list of inconsistent paths. It needs to verify that every element at the bottom category reaches a unique element in the CL after an update. Let n be the number of elements at the bottom category, m the longest path from the bottom category to the CL, and k the number of alternative paths from the bottom category to the CL. Note that m , k and n are all independent values. Then, the function has to search k paths for n elements at the bottom category. Thus, the algorithm runs in $O(n * m * k)$ in a worst case scenario, which implies that all elements in the bottom category are in conflict after an update, which is quite unlikely. Consider also that, in general, the number of categories between the bottom and the CL category is small, as well as the number of alternative paths to the CL category. More than often the hierarchy schema is a tree (i.e., there is no CL), and in this case the algorithm runs in $O(n * \log m * k)$ (the longest path can be computed in $\log m$). The rest of the functions in the algorithm run in lineal time using the list of inconsistent paths, the list of categories in the hierarchy schema, and the rollup functions.

search_repair ()

```

Structure paths {String element, String category, *next, *below};
paths list_inconsistent_paths = NULL;
String new_CL, old_CL, e_u, p_u, parent_child_CL, child_CL1, child_CL2;
Int cost=0, cont_same_elements;
1: if check_Consistency() = 0 then
2:   list_inconsistent_paths= non_strict_paths();
3:   while (list_inconsistent_paths.below ≠ NULL) do
4:     i=0;
5:     cont_same_elements = find_number_paths(list_inconsistent_paths(i));
6:     new_CL = find_new_parent_CL(list_inconsistent_paths(i),e_u);
7:     old_CL = find_old_parent_CL(list_inconsistent_paths(i),new_CL);
8:     {the parents in the CL before and after the update};
9:     cont_1 =number_paths_reaching_element(list_inconsistent_paths(i),new_CL);
10:    cont_2 =number_paths_reaching_element(list_inconsistent_paths(i),old_CL);
11:    if (cont_1 = cont_2) then
12:      {Same # of paths reaching the old and new parent in CL, → try to keep the new parent};
13:      child_CL1 = find_child_CL(list_inconsistent_paths(i),old_CL);
14:      child_CL2 = find_child_CL(list_inconsistent_paths(i),new_CL);
15:      {it captures the element in the category that reach the old(new) parent in CL};
16:      if (check_change_to_new_CL(child_CL1)=1) then
17:        {It is possible to change to the new parent in CL};
18:        cost = cost + change_new_CL(list_inconsistent_paths(i),child_CL1, new_CL);
19:      else
20:        cost = cost + change_old_CL(list_inconsistent_paths(i),child_CL2, old_CL);
21:      end if
22:    else
23:      {# of paths reaching the old parent in CL is greater than the # of paths reaching the new parent in CL, → try
24:      to keep the old parent (second heuristics)};
25:      child_CL2 = find_child_CL(list_inconsistent_paths(i),new_CL);
26:      if (check_change_to_old_CL(child_CL2)=1) then
27:        cost = cost + change_old_CL(list_inconsistent_paths(i),child_CL2, old_CL);
28:      else
29:        for j = 1 TO cont_2 do
30:          child_CL1 = find_child_CL(list_inconsistent_paths(i),old_CL);
31:          cost = cost + change_new_CL(list_inconsistent_paths(i),child_CL1, new_CL);
32:        end for
33:      end if
34:    end if
35:    i = i + cont_same_elements;
36:    move(list_inconsistent_paths,i);
37:  end while
end if

```

Fig. 5. Generating an *r-repair*

5 Discussion and Conclusion

Most efforts addressing inconsistency issues in DWs focus in solving inconsistencies between operational and warehouse data [15,16,17]. No much work has been devoted to study the inconsistencies that may arise when dimension updates are applied. This is due probably to the fact that dimensions were assumed to be static. Hurtado et al. showed that dimensions need to be updated when, for instance, changes in the business rules that lead to the warehouse design occur, or data in the operational sources are updated [7,6]. In these works, dimension updates guarantee that the dimensions remain consistent after the updating operations are applied (if there is a risk of inconsistency, updates are prevented). A similar approach is adopted in [18]. Other approaches to dimension updates accept that the changes may leave the updated dimension inconsistent. Therefore, repairing techniques must be applied in order to guarantee summarizability

[4], which is crucial for OLAP operations. Pedersen et al. [12] presented a first approach to this problem, transforming non-strict into strict dimensions by means of insertion of artificial elements. Caniupán et al. present a logic programming approach to repair dimensions that are inconsistent with respect to a set of constraints [8,19]. Although important to gain insight into the problem of repairing inconsistent dimensions, and containing some interesting theoretical results, from a practical point of view, the approach presented in [8,19] would be computationally expensive in real-world cases. Besides, DW administrators and developers are not acquainted with logic programs. Moreover, for the specific case of reclassification, the work in [8,19] only deal with repairs that may undo the update. On the contrary, the *r-repairs* we present in this paper do not undo the reclassification. Finally, the minimal repairs obtained in [8,19] could lead to rollup functions that do not make sense in the real world (e.g., relating dimension members that are not actually related in any way). Following a different approach, we propose efficient algorithms that lead to consistent dimensions (although not necessarily minimal with respect to the distance function), and where undesired solutions could be prevented.

We have shown that, in general, finding *r-repairs* for dimension instances is NP-complete. However, we also showed that in practice, computing *r-repairs* can be done in polynomial time when the set of updates contains only one reclassification, and the dimension schema has at most one conflicting level. We have explored algorithms to compute *r-repairs* for this class of dimension schemas, and discussed their computational complexity, being in a worst case scenario of order $O(n * m * k)$, where the key term is n , the number of elements in the bottom level affected by the inconsistencies. We would like to remark the fact that in the algorithms presented in this paper, for the sake of generality, we did not include the possibility of preventing rollups that could make no sense in practice. However, it is straightforward to enhance the *search-repair* algorithm to consider only repairs that are acceptable by the user. At least two approaches can be followed here: to prioritize the rollup functions (as, for example, is proposed in [20]), or even to define some rollups to be fixed (and therefore, not allowed to be changed). Of course, in the latter case, it may be the case where a minimal *r-repair* does not exist. We leave this discussion as future work, as well as the experimentation of the algorithms in real-world data warehouses.

Acknowledgements. This project was partially funded by FONDECYT, Chile grant number 11070186. Part of this research was done during visit of Alejandro Vaisman to University del Bío-Bío in 2010. Currently, Mónica Caniupán is funded by DIUBB 110115 2/R. A. Vaisman has been partially funded by LACCIR project LACR-FJR-R1210LAC004.

References

1. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26, 65–74 (1997)
2. Bertossi, L., Bravo, L., Caniupán, M.: Consistent query answering in data warehouses. In: AMW (2009)
3. Hurtado, C., Gutierrez, C., Mendelzon, A.: Capturing Summarizability with Integrity Constraints in OLAP. ACM Transactions on Database Systems 30, 854–886 (2005)
4. Lenz, H., Shoshani, A.: Summarizability in OLAP and Statistical Data Bases. In: SSDBM, pp. 132–143 (1997)

5. Rafanelli, M., Shoshani, A.: STORM: a Statistical Object Representation Model. In: Michalewicz, Z. (ed.) SSDBM 1990. LNCS, vol. 420, pp. 14–29. Springer, Heidelberg (1990)
6. Hurtado, C., Mendelzon, A., Vaisman, A.: Maintaining Data Cubes under Dimension Updates. In: ICDE, pp. 346–355 (1999)
7. Hurtado, C., Mendelzon, A., Vaisman, A.: Updating OLAP Dimensions. In: DOLAP, pp. 60–66 (1999)
8. Caniupán, M., Bravo, L., Hurtado, C.: A logic programming approach for repairing inconsistent dimensions in data warehouses. Submitted to Data and Knowledge Engineering (2010)
9. Dodge, G., Gorman, T.: Essential Oracle8i Data Warehousing: Designing, Building, and Managing Oracle Data Warehouses (with Website). John Wiley & Sons, Inc., Chichester (2000)
10. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. John Wiley & Sons, Inc., Chichester (2002)
11. Hurtado, C., Mendelzon, A.: Reasoning about summarizability in heterogeneous multidimensional schemas. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 375. Springer, Heidelberg (2000)
12. Pedersen, T., Jensen, C., Dyreson, C.: Extending Practical Pre-Aggregation in On-Line Analytical Processing. In: VLDB, pp. 663–674 (1999)
13. Vaisman, A.: Updates, View Maintenance and Materialized Views in Multidimensional Databases. PhD thesis, Universidad de Buenos Aires (2001)
14. Bertossi, L.: Consistent query answering in databases. ACM Sigmod Record 35, 68–76 (2006)
15. Zhuge, Y., Garcia-Molina, H., Wiener, J.L.: Multiple View Consistency for Data Warehousing. In: ICDE, pp. 289–300 (1997)
16. Gupta, H., Mumick, I.S.: Selection of views to materialize under a maintenance cost constraint. In: Beerl, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 453–470. Springer, Heidelberg (1998)
17. Schlesinger, L., Lehner, W.: Extending Data Warehouses by Semiconsistent Views. In: DMDW, pp. 43–51 (2002)
18. Letz, C., Henn, E.T., Vossen, G.: Consistency in Data Warehouse Dimensions. In: IDEAS, pp. 224–232 (2002)
19. Bravo, L., Caniupán, M., Hurtado, C.: Logic programs for repairing inconsistent dimensions in data warehouses. In: AMW (2010)
20. Espil, M.M., Vaisman, A., Terribile, L.: Revising data cubes with exceptions: a rule-based perspective. In: DMDW, pp. 72–81 (2002)

GrHyMM: A Graph-Oriented Hybrid Multidimensional Model

Francesco Di Tria, Ezio Lefons, and Filippo Tangorra

Dipartimento di Informatica
Università degli Studi di Bari “Aldo Moro”
Via Orabona 4, 70125, Bari, Italy
{francescoditria, lefons, tangorra}@di.uniba.it

Abstract. The main methodologies for the data warehouse design are based on two approaches which are opposite and alternative each other. The one, based on the data-driven approach, aims to produce a conceptual schema mainly through a reengineering process of the data sources, while minimizing the involvement of end users. The other is based on the requirement-driven approach and aims to produce a conceptual schema only on the basis of requirements expressed by end users. As each of these approaches has valuable advantages, it is emerged the necessity to adopt a hybrid methodology which combines the best features of the two approaches. We introduce a conceptual model that is based on a graph-oriented representation of the data sources. The core of the proposed hybrid methodology is constituted by an automatic process of reengineering of data sources that produces the conceptual schema using a set of requirement-derived constraints.

Keywords: *i** framework, graph-based modelling, schema validation.

1 Introduction

The actual lack of a standard process for data warehouse design has led to the definition of several methodologies. This is especially true in the conceptual design, where opposite approaches can be adopted based on the *requirement-driven* and *data-driven* methodologies [1]. The requirement-driven approach, also known as *demand-driven* or *goal-oriented* methodology, aims to define multidimensional schemas using business goals resulting from the decision makers' needs. The data sources are considered later, when the Extraction, Transformation, and Loading (ETL) phase is designed [2]. In this feeding plan, the multidimensional concepts (such as facts, dimensions, and measures) have to be mapped on the data sources in order to define the procedures to populate the data warehouse by cleaned data. At this point, it may happen that the designer discovers that the needed data are not currently available at the sources. On the contrary, the data-driven approach, also known as *supply-driven* methodology, aims to define multidimensional schemas on the basis of a remodelling of the data sources. This process is individually executed by the designer who minimizes the involvement of end users and, consequently, goes towards a possible failure of their expectations. To overcome these limits, in the last years the necessity to define hybrid methodologies arose [3].

Here, we present a Graph-oriented Hybrid Multidimensional Model (*GrHyMM* for short) for data warehouse conceptual design. This model is based on a graph-oriented representation of (part of) data sources [4, 5]. In this way, the traditional operations performed in the reengineering process (such as adding and removing attributes, or modifying functional dependencies) correspond to basic operations on graphs (such as adding and removing nodes). Nevertheless, this remodelling activity relies on a set of constraints that have to be derived from the requirement analysis, avoiding the oversight of business needs. Using these constraints, it is possible to perform the remodelling activity in a supervised and automatic way. At the end of the conceptual design, the validation of the obtained schema is performed [6], in order to verify whether it supports all queries included into a preliminary workload, which contains the typical queries that end users will execute in the analytical processing step.

The paper is organized as follows. Section 2 introduces the conceptual design framework. Section 3 presents the language we defined to represent business goals. Section 4 discusses the source analysis. Section 5 illustrates the multidimensional model that is at the basis of the hybrid methodology. Finally, Section 6 contains our concluding remarks and notes on work in progress.

2 Methodological Framework

Our framework for data warehouse design is depicted in Figure 1 and describes the activities that must be performed by designers, along with the produced artifacts.

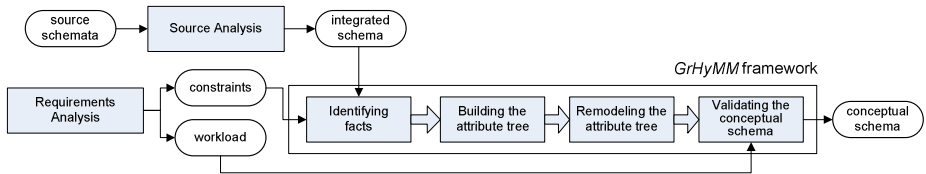


Fig. 1. Framework for data warehouse design

The activities are performed sequentially as follows.

1. *Requirement analysis.* In this step, the needs of end users (namely the decision makers) are investigated. To this aim, we use the i^* framework and its application to data warehousing [7], which allows to explicitly represent business goals in reference to the several actors of the system. In general, there are two main categories of actors: the decision makers and the data warehouse itself, each of them performs specific tasks in order to achieve their own goals. The designer has to: (1a) define the tasks and the goals of the different actors using the i^* framework; (1b) translate the tasks of the decision makers into a preliminary workload (*see*, Section 3.1); and (1c) translate the goals and the tasks of the data warehouse into a set of constraints (*see*, Section 3.2). Then, both artifacts (*ie*, the workload and set of constraints) must be given in input to the conceptual design process in order to start the modelling phase in an automatic way.

2. *Source analysis*. In this step, the different schemas of data sources must be analyzed and then reconciled, in order to obtain a global and integrated schema. Also this artifact must be given in input to the conceptual design process. The constraints derived from the requirement analysis are used to perform a reengineering of the source schema.
3. *Conceptual design*. This step is based on *GrHyMM*, the graph-oriented multidimensional model. The sub-steps to be performed in this phase are: (3a) identifying facts present in the source schema on the basis of the constraints; (3b) building an attribute tree for each fact; (3c) remodelling the tree on the basis of the constraints; and (3d) verifying whether all the remodelled trees agree with the workload (this is the so-called validation process).
4. *Logical design*. In this step, the conceptual schema is transformed into a logical one with reference to the data model of the target database system.
5. *Physical design*. The logical schema implementation ends the design process with defining the physical database properties based on the specific features provided by the database system, such as indexing, partitioning, and so on.

Here, we address only the first steps 1 to 3, as steps 4 and 5 strongly depend on target systems. As an example, ROLAP and MOLAP systems can be utilized in step 4.

3 Requirement Analysis

In the *i** framework, user requirements, alias business goals, are exploded into a more detailed hierarchy of nested goals: (a) strategic goals, or high-level objectives to be reached by the organization; (b) decision goals, to answer how strategic goals can be satisfied; and (c) information goals, to define which information is needed for decision making. To do this, the designer must produce a model describing the relationships among the main actors of the organization, along their own interests. This model is the so-called *strategic dependency* model and aims to outline how the data warehouse helps decision makers to achieve business goals. As in this context the *i** framework applies to data warehousing, the data warehouse itself has to be an actor of the system. Each actor in a strategic dependency model is further detailed in a *strategic rationale* model that shows the specific tasks the actor has to perform in order to achieve a given goal.

In Figure 2, there is reported an example of strategic rationale model showing the strategic goals and the tasks of the decision maker and data warehouse actors.

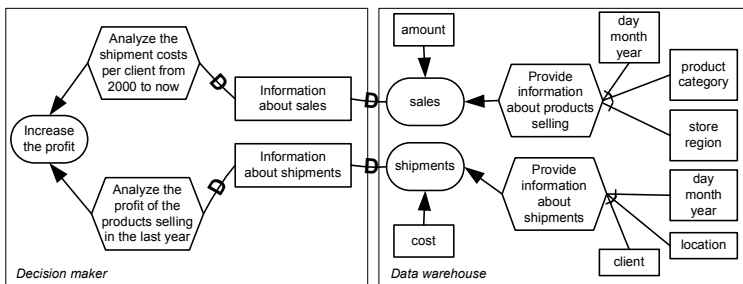


Fig. 2. Strategic rationale model

In our methodology, the designer translates the tasks of the decision makers into a workload, while the goals and the tasks of the data warehouse are transformed into a set of constraints that must be taken into account in the next data remodelling phase.

3.1 Workload Representation

The workload contains a set of queries derived from user requirements and helps the designer to identify the information the users are interested in. In a few words, it includes the typical queries that will be executed by decision makers in the analytical processing step (OLAP).

The grammar for a high-level representation of the queries of the workload follows.

```

<query>          ::- <function>(<fact_pattern>);
<fact_pattern>  ::- <fact>[<aggreg_pattern>;<sel_clause>].<measure>
<aggreg_pattern> ::- <level> | <aggreg_pattern>,<level>
<sel_clause>    ::- <attribute> <comp_op> <constant> |
                  <sel_clause> <logical_op> <sel_clause>
<function>      ::- avg | sum | count
<logical_op>    ::- and | or
<comp_op>       ::- ≥ | ≤ | < | > | =
<fact>          ::- <identifier>
<measure>       ::- <identifier>
<level>         ::- <identifier>
<attribute>     ::- <identifier>.

```

Notice that *constant* is any number or string, and *identifier* a user-defined name corresponding to a valid variable (the name of a table or column, for example).

Example 1. With reference to the decision makers' tasks shown in Figure 2, we have to manually translate them in high-level queries. First, the task “Analyze the profit per products sold in 2010” corresponds to the SQL-like statement “*select product, sum(amount) from sale join product where year = 2010 group by product_id;*” and grammar-based statement “*sum(sale[product; year = 2010].amount);*”. On the other hand, the second task corresponds to the query “*select client, sum(cost) from shipment join client where year ≥ 2000 and year ≤ 2011 group by client_id;*” and statement “*sum(shipment[client; year ≥ 2000 and year ≤ 2011].cost);*”. \diamond

The workload will be used in the final step of the conceptual design in order to perform the validation process. If all the queries of the workload can be effectively executed over the schema, then such a schema is assumed to be validated and the designer can safely translate it into the corresponding logical schema. Otherwise, the conceptual design process must be revised.

3.2 Constraints Representation

For each resource needed from decision makers, the data warehouse must provide adequate information by achieving its own goals. Moreover, a goal must have measures that are resources to be used in order to provide the information required for decision making. Therefore, a fact is generated in order to allow the data warehouse

to achieve its own goal. Finally, for each measure, a context of analysis must be provided by a task of the data warehouse. So, starting from measures and dimensions emerged from business goals, they can be identified some constraints the designer must necessarily consider. The constraints will be used in the main step of the conceptual design in order to perform the remodeling process in a supervised way.

The grammar for a high-level representation of constraints follows.

```

<constraint>    ::- <fact>[<dimensions>]. [<measures>];
<dimensions>   ::- <dimension> | <dimensions>; <dimension>
<dimension>    ::- <level> | <dimension>, <level>
<measures>     ::- <measure> | <measures>, <measure>
<level>        ::- <identifier>
<measure>      ::- <identifier>
<fact>         ::- <identifier>.

```

Example 2. The constraint “*sales[product, category; day, month, year; store, region]. [amount];*” states that we must have a fact, namely *sales*, provided with the *amount* measure. This fact has three dimensions. The first dimension has two levels: *product* (the first dimensional level of the hierarchy) and *category* (the second one). The second dimension has levels *day*, *month*, and *year*, while the third dimension levels *store* and *region*. ◇

4 Sources Analysis

The aim of this activity is to produce a global and reconciled schema coming from the integration of the heterogeneous data sources. In this phase, it is very important to align the different schemas [8] using a unique model. The methodology to accomplish this is faced in [9], where a supermodel is proposed provided with a set of meta-constructs that can be mapped to constructs of specific models. The transformation between schemas is performed via Datalog rules.

A class of problems derives from inconsistency among schemas, as cardinality constraints. As an example, in a university database a professor can hold one or many courses but in another one a professor can hold zero or two courses. Whereas there is an agreement on the meaning of the *professor* entity, there is a discordance about how many courses a professor may hold.

Inconsistencies can be treated using an ontological approach [10]. So, starting from a logical schema, an abstraction process has to be performed to derive a conceptual schema, such as an Entity Relationship (ER) schema. Then, all the inconsistencies in the conceptual schemas must be solved by defining a common and shared ontology. Indeed, an ER schema is used to represent *locally true* concepts, that is, concepts that are true in a given context. On the other hand, an ontology is used to represent *necessarily true* concepts, that is, concepts that are always true [11].

Relative to the previous example, if a professor may hold one or many courses in one database and a professor may hold zero or two courses in another database, then this inconsistency can be reconciled in an ontology that states that a professor, in general, may hold zero or many courses.

In what follows, we assume that there are no inconsistencies among schemas. In particular, we assume that also user requirements agree with the concepts and the terminology used in data sources.

5 Conceptual Design

This section describes how to build and to remodel an attribute tree to represent a fact identified from user requirements. In detail, the design consists of four steps to be performed in automatic way: the *identification of facts* is done by marking possible fact tables in data source and by matching them with constraints derived from user requirements; the *building of an attribute tree* is defined by a set of formal assumptions; the *remodeling of the tree* is defined by a novel algorithm; and, at last, the *validation* is based on a set of non-ambiguous rules represented using the *first order logic* [6].

5.1 Identifying Facts

The main difficulty in this step is to correctly map the multidimensional concepts to the integrated schema. For example, in a relational schema, the designer must face the problem to identify which table can be considered as a fact. To solve this problem, several methodologies have been defined to automatically identify multidimensional concepts in order to correctly map them onto the data sources.

For example, in [12] the user requirements can be entirely derived by defining a preliminary workload. On the basis of this assumption, the authors propose an algorithm able to automatically create a graph (whose nodes are the tables of the data sources and edges are the joins between tables) that aims to identify whether each table can be considered as a fact table or a dimension table. They state that a correct labelling of all nodes generates a valid multidimensional schema. The labels are assigned by examining the role played by tables and attributes in the SQL queries included in the preliminary workload.

In our model, we mainly deal with the correct identification of facts, as these are the starting point to build the initial attribute tree. According to some heuristics reported in [13], we suggest that a table in the data source is a candidate fact provided that: (a) it is a very-frequently-updated relation; (b) it has numeric attributes; (c) it does not have its own primary key; (d) its primary key is composed of two or more foreign keys; and (e) it represents a many-to-many relationship among relations.

We consider the facts involved in the constraints coming from the requirement analysis. Given a fact F_1 in a constraint, we choose a candidate fact F_2 in the integrated schema such that F_2 corresponds to F_1 . We assume no inconsistencies exist (nor syntactic neither semantic) among user requirements and data sources, though in many cases user requirements do not agree with concepts in data source.

Given a fact, we now show how to build the initial attribute tree. Of course, there can be as many trees as many facts.

5.2 Building the Attribute Tree

Here, we present *GrHyMM*, a suitable model to represent relational databases.

Let $G = (N, E)$ be a tree, where:

- $N = \{A_1, A_2, \dots, A_n\}$ is a set of n nodes,
- $E = \{(A_i, A_j) \mid A_i, A_j \in N, i \neq j\} \subset N \times N$ is a set of edges, and
- A_1 is the root of G .

Assumption 1. Let $R(X_1, \dots, X_n)$ be a relation, and let $G = (N, E)$ be an attribute tree. We assume $X_i \in N, \forall i = 1, \dots, n$. \diamond

We also assume that $G = (N, E)$ is the tree obtained from R , by invoking the algorithm *tree*(R). In particular, $G = (N, E)$ is the tree constructed by considering the relation R as a starting point.

Assumption 2. Let $R(X_1, \dots, X_n)$ be a relation, and let $G = (N, E)$ be a tree. We assume $(X_i, X_j) \in E$, if X_i is the primary key of R and $i \neq j$. \diamond

On the basis of Assumption 2, the edge (X_i, X_j) states that the non-trivial (for $i \neq j$) functional dependency $X_i \rightarrow X_j$ holds on R (in this case, established by a primary key constraint).

It is worth noting that the primary key of a relation can be composed of more than one attribute. In this case, the node representing the primary key is a composite node.

Example 3. Let *Student*(code, university, name, address) be a relation, whose primary key is composed of *code* and *university*. Then, naming the primary key by the relation name, we have that:

- $(code, university) \rightarrow code \Rightarrow (Student, code) \in E$
- $(code, university) \rightarrow university \Rightarrow (Student, university) \in E$
- $(code, university) \rightarrow name \Rightarrow (Student, name) \in E$
- $(code, university) \rightarrow address \Rightarrow (Student, address) \in E$. \diamond

Assumption 3. Let $R(X_1, \dots, X_n)$ and $S(Y_1, \dots, Y_m)$ be relations, and let $G = (N, E)$ be a tree. We assume $(X_i, Y_j) \in E$, if

- Y_j is the primary key of the relation S ,
- X_i is a foreign key referencing Y_j . \diamond

On the basis of Assumption 3, an edge can also indicate the functional dependency established by a foreign key constraint. So, the next Assumption 4 describes how to build a tree starting from a relation having r foreign keys. Accordingly, we assume that the algorithm *tree* is able to navigate among relations via foreign key constraints. To this end, Assumption 5 describes how to build a tree when a *many-to-many* relationship is encountered in the navigation process.

Assumption 4. Let $R_1(X_{11}, \dots, X_{1h}), R_2(X_{21}, \dots, X_{2k}), \dots, R_r(X_{r1}, \dots, X_{rq})$ be r relations, and let $T(Z_1, \dots, Z_p)$ be a relation where $r \leq p$. If

- X_{ij_i} is the primary key of the relation $R_i, i = 1, 2, \dots, r$, and
- $\exists Z_{t_1}, \dots, Z_{t_r} \in T$, such that

- $\forall i = 1, \dots, r, Z_{t_i}$ is a foreign key referencing X_{ij} , and
- the set $Z = \{Z_{t_1}, \dots, Z_{t_r}\}$ forms the primary key of the relation T ,

then, the tree $G = (N, E)$ is defined so:

- $N = \{T\} \cup \{X_{ij} \mid X_{ij} \in R_i; i = 1, \dots, r\}$,
- T is the root node of the tree,
- $(X_{ij}, X_{il}) \in E, \forall j \neq l$,
- $(T, X_{ij}) \in E, \forall i = 1, \dots, r$, and
- $(T, Z_{t_v}) \in E, Z_{t_v} \in T$ and $Z_{t_v} \notin Z$. ◇

Assumption 5. Let $R(X_1, \dots, X_h)$, $S(Y_1, \dots, Y_k)$, and $T(Z_1, \dots, Z_p)$ be three relations, and let $G = (N, E)$ a tree. If

- X_i is the primary key of the relation R ,
- Y_j is the primary key of the relation S ,
- $\exists Z_{t_i}, Z_{t_j} \in T$ such that
 - Z_{t_i} is a foreign key referencing X_i ,
 - Z_{t_j} is a foreign key referencing Y_j , and
 - (Z_{t_i}, Z_{t_j}) is the primary key of the relation T , and
- T is not the root node of the G ,

then

- $(X_i, T) \in E$,
- $(T, Y_j) \in E$, and
- $(T, Z_{t_v}) \in E, \forall v = 1, \dots, p$ and $v \neq t_i, t_j$. ◇

Example 4. Let us consider the schema of Figure 3(a). The algorithm *tree* starts from *Sale* and reaches *Cataloging*. The primary key of the *Cataloging* relation is composed of foreign keys. In fact, this relation is an intermediate relation, that establishes a *many-to-many* relationship between *Product* and *Catalog*. So, the edge between *Product* and *Cataloging* is represented by a *dot-headed* arrow, as depicted in Figure 3(b), to indicate that, given a *prodlid* instance, many occurrences in the *Cataloging* relation will correspond to. ◇

In conclusion, we note that, sometimes, a node can have multiple parents. This happens when an attribute is pointed out by more than one table. In this case, the tree is a graph or a semi-tree, since there is always a root node representing the fact.

5.3 Remodeling the Attribute Tree

In what follows, given an attribute A , we denote with A the node representing the corresponding attribute. Moreover, we indicate with $A \rightarrow B$ the edge existing from the A node to the B node. At last, we denote with $(A, B) :- C$ the fact that the attribute C can be computed using A and B , that is, C is a derived measure. As an example, $(price, quantity) :- amount$ means that there exists an expression to compute *amount*

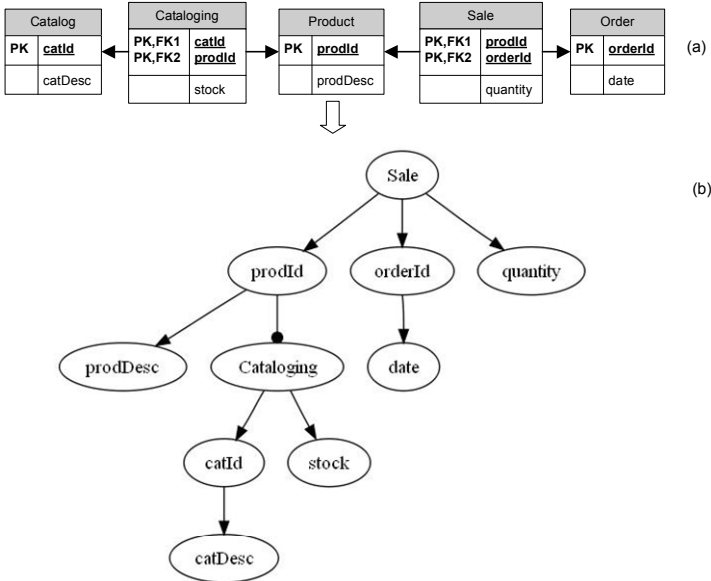


Fig. 3. (a) relational schema; (b) the result of $tree(Sale)$

using *price* and *quantity*. The basic operations on a tree are: (a) *add* $A \rightarrow B$, or adding an edge from A to B ; (b) *remove* $A \rightarrow B$, or removing the edge from A to B ; and (c) *prune* A , or removing the A node with all its children.

Let us consider a tree T and a constraint coming from the user requirements. In informal way, we create as many root children as many measures there are in the constraint. Moreover, we add a root child for each first dimensional level in the constraint. In a recursive way, the other dimensional levels are added as children nodes of their own predecessor levels. In general, when we add a new node B to a parent node A , the node B can be created *ex novo* or can be already present in the tree. In the latter case, the edge between B and the old parent of B must be deleted and a new edge between B and the new parent A must be created; this is the so-called *change parent* operation.

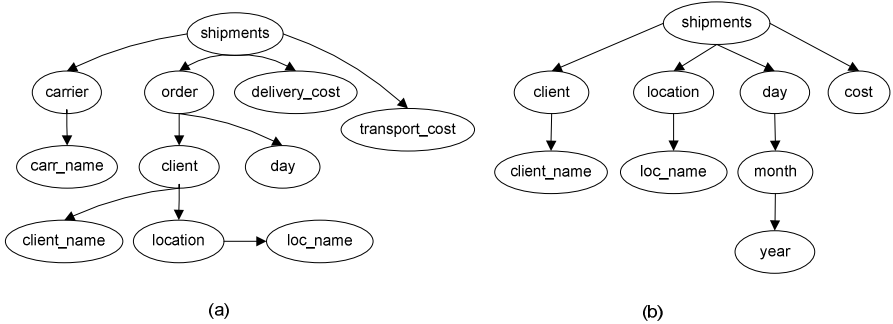
Algorithm 1 performs the attribute tree remodelling.

Example 5. On the tree of Figure 4(a), given the constraint $shipments[client; location; day, month, year].[cost]$; we perform the following operations. First, since *cost* can be computed by the sum of *transport_cost* and *delivery_cost*, then we delete both the *transport_cost* and *delivery_cost* nodes, and we add the *cost* node as root child (lines 6-8). Second, also the *client*, *location*, and *day* nodes becomes three distinct root children via a change parent operation (lines 15-18). Third, in the constraint, the only hierarchy to be built is formed by *day*, *month*, and *year*. So, a new node *month* becomes the child of *day* and a new node *year* becomes the child of *month* (line 24). All the other nodes, such as *carrier* and *order* are pruned (lines 30-31). The resulting tree is shown in Figure 4(b).

Algorithm 1. Attribute tree remodelling.

Input: T graph

1. **for each** measure M
2. **if** $M \in T$ **then**
3. remove $parent(M) \rightarrow M$
4. add $root \rightarrow M$
5. **else**
6. **if** $(X_1, X_2, \dots, X_n) :- M$ **and** $(X_1, X_2, \dots, X_n) \in T$
7. remove $parent(X_i) \rightarrow X_i$ for $i = 1, \dots, n$
8. add $root \rightarrow M$
9. **end if**
10. **end if**
11. **end for**
12. **for each** dimension D
13. **for each** level L_j **in** D, for $j = 1, \dots, m$
14. **if** $L_j = L_1$
15. **if** $L_j \in T$ **then**
16. remove $parent(L_j) \rightarrow L_j$
17. **end if**
18. add $root \rightarrow L_j$
19. **else**
20. **if** $parent(L_j) \neq L_{j-1}$
21. **if** $L_j \in T$ **then**
22. remove $parent(L_j) \rightarrow L_j$
23. **end if**
24. add $L_{j-1} \rightarrow L_j$
25. **end if**
26. **end if**
27. **end for**
28. **end for**
29. **for each** $N \in T$
30. **if** $\nexists D$ **such that** $N \in D$
31. prune N
32. **end if**
33. **end for**

**Fig. 4.** (a) *shipments* attribute tree; (b) remodelled *shipments* attribute tree

5.4 Validating the Conceptual Schema

In the validation process [14], we have several attribute trees (whose collection is a conceptual schema) and a workload, composed of high-level queries. A query is assumed to be validated if there exists at least an attribute tree such that the following conditions hold: (a) the fact is the root of the tree; (b) the measures are the children nodes of the root; (c) for each level in the aggregation pattern, there exists a path from the root to a node X , where X is a non-leaf node representing the level; and (d) for each attribute in the selection clause, there exists a path from the root to a node Y , where Y is a leaf node representing the attribute.

If all the queries are validated, then each attribute tree can be considered as a cube, where the root is the fact, non-leaf nodes are aggregation levels, and leaf nodes are descriptive attributes belonging to a level. So, the conceptual design ends and the designer can transform the conceptual schema into a logical one. On the other hand, if a query cannot be validated, then the designer has to opportunely modify the tree. For example, if an attribute of the selection clause is not in the tree, then the designer can decide to add a further node. A deeper discussion on the validation process can be found in [6], along with the methodology to execute this task in automatic way.

Example 6. Let us consider the following query: *sum(shipments[client; year \geq 2000 and year \leq 2011].cost)*. Such a query is validated on the tree of Figure 4(b) because *shipments* is the root of the tree, *cost* is the child node of the root, *client* is a non-leaf node reachable from the root (that is, *client* is a correct aggregation level belonging to the first dimension), and *year* is a leaf node reachable from the root. \diamond

The validation process ends the conceptual design. Two similar approaches can be found in [15, 16]. They differ from our approach in that they produce conceptual schemas by reconciling user requirements with data sources. In this way, possible lack of compliance to the user requirements needs can arise, for some requirements could have been disregarded in order to preserve data integrity. On the other hand, we start from data source and use user requirements as constraints to remodel a source schema in automatic way, preserving both business needs and data integrity.

6 Conclusions

In this paper, we have presented a multidimensional model for data warehouse conceptual design. This model can be surely adopted in a hybrid methodology for data warehouse design. In fact, the model uses a graph-oriented representation of an integrated relational schema. So, the data remodelling process is expressed according to traditional operations on a tree and is based on a set of constraints derived from the requirement analysis, along with a preliminary workload to be used to validate the conceptual schema. The main contribution of this model is that all the steps to be performed in the conceptual design can be automated thanks to the high degree of formalization given by i^* schemas and to the precise and non-ambiguous rules to be followed in the remodeling activity.

We are now working on the development of a logical program that considers such rules for the construction and the remodeling of the attribute trees, and the validation of the resulting conceptual schema.

References

1. Ballard, C., Herreman, D., Schau, D., Bell, R., Kim, E., Valencic, A: *Data Modeling Techniques for Data Warehousing*. IBM Redbooks (1998)
2. Kimball, R.: *The Data Warehouse Lifecycle Toolkit*, 2nd edn. *Practical Techniques for Building Data Warehouse and Business Intelligence Systems*. John Wiley & Sons, Chichester (2008)
3. Romero, O., Abelló, A.: A Survey of Multidimensional Modeling Methodologies. *International Journal of Data Warehousing and Mining* 5(2), 1–23 (2009)
4. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *International Journal of Cooperative Information Systems* 7, 215–247 (1998)
5. dell’Aquila, C., Di Tria, F., Lefons, E., Tangorra, F.: Dimensional Fact Model Extension via Predicate Calculus. In: *24th International Symposium on Computer and Information Sciences*, Cyprus, pp. 211–217 (2009)
6. dell’Aquila, C., Di Tria, F., Lefons, E., Tangorra, F.: Logic Programming for Data Warehouse Conceptual Schema Validation. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) *DAWAK 2010*. LNCS, vol. 6263, pp. 1–12. Springer, Heidelberg (2010)
7. Mazón, J.N., Trujillo, J., Serrano, M., Piattini, M.: Designing Data Warehouses: from Business Requirement Analysis to Multidimensional Modeling. In: Cox, K., Dubois, E., Pigneur, Y., Bleistein, S.J., Verner, J., Davis, A.M., Wieringa, R. (eds.) *Requirements Engineering for Business Need and IT Alignment*. Wales Press, University of New South (2005)
8. Rahm, E., Bernstein, P.: A Survey of Approaches to Automatic Schema Matching. *The International Journal on Very Large Data Bases* 10(4), 334–350 (2001)
9. Atzeni, P., Cappellari, P., Torlone, R., Bernstein, P., Gianforme, G.: Model-Independent Schema Translation. *The International Journal on Very Large Data Bases* 17(6), 1347–1370 (2008)
10. Romero, O., Abelló, A.: Automating Multidimensional Design from Ontologies. In: *Proceedings of the ACM 10th International Workshop on Data Warehousing and OLAP*, Lisbon, Portugal, pp. 1–8 (November 9, 2007)
11. Spyns, P., Meersman, R., Jarrar, M.: Data Modelling versus Ontology Engineering. *ACM SIGMOD Record* 31(4), 12–17 (2002)
12. Romero, O., Abelló, A.: Multidimensional Design by Examples. In: Tjoa, A.M., Trujillo, J. (eds.) *DaWaK 2006*. LNCS, vol. 4081, pp. 85–94. Springer, Heidelberg (2006)
13. Carmè, A., Mazón, J.N., Rizzi, S.: A Model-Driven Heuristic Approach for Detecting Multidimensional Facts in Relational Data Sources. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) *DAWAK 2010*. LNCS, vol. 6263, pp. 13–24. Springer, Heidelberg (2010)
14. Golfarelli, M., Rizzi, S.: A Methodological Framework for Data Warehouse Design. In: *1st ACM International Workshop on Data Warehousing and OLAP*, Washington D.C., USA, pp. 3–9 (1998)
15. Giorgini, P., Rizzi, S., Garzetti, M.: GRANd: A Goal-oriented Approach to Requirement Analysis. *Decision Support Systems* 45(1), 4–21 (2008)
16. Mazón, J.N., Trujillo, J., Lechtenborger, J.: Reconciling Requirement-driven Data Warehouses with Data Sources via Multidimensional Normal Forms. *Data Knowl. Eng.* 63(3), 725–751 (2007)

Ontologies and Functional Dependencies for Data Integration and Reconciliation

Abdelghani Bakhtouchi¹, Ladjel Bellatreche², and Yamine Ait-Ameur²

¹ National High School for Computer Science (ESI), Algiers, Algeria
a_bakhtouchi@esi.dz

² LISI/ENSMA – Poitiers University, Futuroscope, France
{bellatreche,yamine}@ensma.fr

Abstract. Integrating data sources is the key success of business intelligence systems. The exponential growth of autonomous data sources over the Internet and enterprise intranets makes the development of integration solutions more complex. This is due to two main factors: (i) the management of the source *heterogeneity* and (ii) the *reconciliation* of query results. To deal with the first factor, several research efforts proposed the use of ontologies to explicit semantic of each source. Two main trends are used to reconcile the query results: (i) the supposition that different entities of sources representing the same concept have the same key – a strong hypothesis that violates the autonomy of sources. (ii) The use of *statistical* methods which are not usually suitable for *sensitive-applications*. In this paper, we propose a methodology integrating sources referencing *shared domain ontology* enriched with *functional dependencies* (\mathcal{FD}) in a mediation architecture. The presence of \mathcal{FD} gives more autonomy of sources in choosing their primary keys and facilitates the result reconciliation. Our methodology is validated using dataset of Lehigh University Benchmark.

1 Introduction

In the past decades, **E**nterprise and **I**nformation **I**ntegration (EII) became an established business, where commercial and academic tools integrating data from various sources exist. They provide uniform and transparent access to data. The spectacular development of this business is largely due to companies requiring being able to access data located over the Internet and within their Intranets [8,12]. Integration problem inputs are a set of *distributed, heterogeneous, autonomous* sources where each one has its schemes and populations. It outputs a unified description of source schemes via an integrated schema and mapping rules allowing the access to data sources. The construction of a data integration system is a hard task due to the following main points: **(a)** the large number of data sources candidate for integration, **(b)** the lack of explicitation of the semantic of sources, **(c)** the heterogeneity of sources and **(d)** the autonomy of sources. **(a)** *The explosion of data sources*: the number of data sources involved in the integration process is increasing. The amount of information generated in

the world increases by 30% every year and this rate is bound to accelerate [8], especially in domains such as E-commerce, engineering, etc. Integrating these mountains of data requires *automatic solutions*. **(b) The lack of explicitation of the semantic of sources:** the semantics of data sources is usually not explicit. Most sources participating in the integration process were designed to satisfy *day-to-day* applications and not to be integrated in the future. Often, the small amount of semantic contained in their conceptual models is lost, since only their logical models are implemented and used by applications. The presence of a conceptual model may offer designers to express the application requirements and domain knowledge in an intelligible form for a user. Thus, its absence or any other semantic representation in final databases makes their interpretation and understanding complicated, even for designers who have good knowledge of the application domain. **(c) The heterogeneity of data sources impacts both the structure and the semantic.** Structural heterogeneity exists because data sources may have different structures and/or different formats to store their data. The autonomy of the sources increases heterogeneity significantly. Indeed, the data sources are designed independently by various designers with different application objectives. Semantic heterogeneity presents a major issue in developing integration systems [11]. It is due to different interpretations of real world objects, generating several categories of conflicts (naming conflicts, scaling conflicts, confounding conflicts and representation conflicts [10]). **(d) Autonomy of sources:** most sources involved in the data integration are fully autonomous in choosing their schemes.

To deal with semantic problem and ensure an automatic data integration, an important number of research studies propose the use of ontologies, where several integration systems were proposed around this hypothesis. We can cite COIN [10], Observer [14], OntoDaWa [3], etc. In [3], we claim that if the semantic of each source participating in the integration process is explicit (in *a priori* way), the integration process becomes automatic. This means the used ontology exists before the creation of the sources. This assumption is reasonable since several domain ontologies exist in various areas: medicine, engineering, travel, etc. The explicitation of different concepts used in a database leads to the concept of *ontology-base databases (OBDB)*. Several academic and industrial systems offer solutions to manage this type of databases (e.g., *Jena, Sesame, Oracle, IBM Sor*).

To deal with data reconciliation issue, we figure out that most existing integration systems (with a mediator architecture) suppose that there is a common single identifier for each common concept between the sources. The assumption facilitates the reconciliation of query results, but it violates the sources autonomy. Other research efforts use of entity reconciliation methods [4,17] performed either by *entity matching* and *data fusion*. These approaches are efficient for linguistic and information retrieval applications [8], but they may suffer in the context of sensitive applications such as banking, healthcare, travel, engineering, etc. The similarity between conceptual models and ontologies and the recent work focusing on the definition of functional dependencies \mathcal{FD} on ontologies [16]

motivate us to consider them as a part of the ontology definition. The presence of \mathcal{FD} allows designers generating all candidate keys for each class of an ontology. As consequence, a source designer may pick her/his primary keys from candidate keys. This gives more autonomy for sources and contributes in reconciling query results. This issue is studied in this paper, where a methodology of integrating ontology-based databases referencing a shared ontology enriched by functional dependencies is given in a mediator architecture.

The paper is structured as follows. Section 2 presents the concepts and the formal definitions of ontology and \mathcal{OBDB} . Section 3 describes an extension of ontology model by \mathcal{FD} and shows their impact on data reconciliation. Section 4 describes in details our integration system with its main components. Section 5 presents experimental studies. Section 6 concludes the paper.

2 Background

In this section, we present concepts and definitions related to the ontology and data ontology-based database to facilitate the understanding of our proposal.

Formal Model for Ontologies. In this work, we concentrate on canonical ontologies that describe concepts and not the words of a language and whose definitions do not contain any redundancy. They adopt an approach of structuring of information in term of classes and properties and associate to these classes and properties a single identifiers reusable in various languages. These ontologies can be considered as *shared conceptual models*. They contain the core classes and play the role of a global schema in database integration architecture. Formally, a such ontology is defined as the quadruplet $\mathcal{O} : \langle \mathcal{C}, \mathcal{P}, Sub, Applic \rangle$ [3], where: (-) \mathcal{C} is the set of the classes used to describe the concepts of a given domain; (-) \mathcal{P} is the set of properties used to describe the instances of the \mathcal{C} classes; (-) Sub is the subsumption function defined as $Sub: \mathcal{C} \rightarrow 2^{\mathcal{C}}$. For a class c of the ontology, it associates its direct subsumed classes. Sub defines a partial order over \mathcal{C} . (-) $Applic$ is a function defined as $Applic: \mathcal{C} \rightarrow 2^{\mathcal{P}}$. It associates to each class of the ontology, the properties that are applicable for each instance of this class and that may be used, in the database, for describing its instances. Note that for each $c \in \mathcal{C}$, only a subset of $Applic(c)$ may be used in any particular database, for describing c instances.

\mathcal{OBDB} is a database *usually composed of four part* [7], since it stores both *data* and *ontology* describing their sense. *Part 1* and *2* are traditional parts available in all DBMSs, namely the *data part* that contains instance data and *meta-base* part that contains the system catalog. The ontology part (3) allows the representation of ontologies in the database. The *meta-schema* (4) part records the ontology model into a reflexive *meta-model*. For the *ontology part*, the *meta-schema part* plays the same role as the one played by the meta-base in traditional databases. By means of naming convention, the meta-base part also represents the logical model of the content, and its link with the ontology, thus

representing implicitly the conceptual model of data in database relations. Formally, an *OBDB* is a quadruplet $\langle \mathcal{O}, \mathcal{I}, \mathcal{Sch}, \mathcal{Pop} \rangle$, where: (-) \mathcal{O} is an ontology $\mathcal{O} : \langle \mathcal{C}, \mathcal{P}, \mathcal{Sub}, \mathcal{Applic} \rangle$, (-) \mathcal{I} is the set of instances of the database. The semantics of these instances is described in \mathcal{O} by characterizing them by classes and properties values, (-) $\mathcal{Pop} : \mathcal{C} \rightarrow 2^{\mathcal{I}}$ associates to each class its own instances. $\mathcal{Pop}(c)$ constitutes the population of c , (-) $\mathcal{Sch} : \mathcal{C} \rightarrow 2^{\mathcal{P}}$ associates to each ontology class c of \mathcal{C} the properties, which are effectively used to describe the instances of the class c . For each class c , $\mathcal{Sch}(c)$ must satisfy: $\mathcal{Sch}(c) \subseteq \mathcal{Applic}(c)$.

3 Adding Functional Dependencies to Ontologies

Traditional ontology formalisms do not support \mathcal{FD} in their definition. Data dependencies have been introduced as a general formalism for a large class of database constraints that augments the expressivity of database models [1]. \mathcal{FD} compose a particularly interesting data dependency that elegantly models the relationships between attributes of a relation. \mathcal{FD} are used for defining primary keys and in the normalization theory. Other important application of \mathcal{FD} in database includes query rewriting and query evaluation [13]. If we transpose similar rules to ontology world, we discover that \mathcal{FD} could be very useful to enrich the expressivity of the knowledge representation and data cleaning [9].

Recently, couple of research studies \mathcal{FD} in the context of ontologies. Two main categories of \mathcal{FD} are distinguished [5,6,16,18]: (1) *intra-class dependencies* and (2) *inter-class dependencies*. In the first category, we find the work of [2] which is quite similar to those proposed in traditional databases. Each ontology class c may have a set of \mathcal{FD} defined on its simple properties $\{p_1, \dots, p_n\}$ ¹.

The second category involves dependency between classes. Two types arise: (1) \mathcal{FD} with a single class in the left part and (2) \mathcal{FD} with several classes in the left part. In the first type; we find the work of [16]. The authors define a \mathcal{FD} ($C_1 \rightarrow C_2$) between classes C_1 and C_2 when each instance of the class C_1 determines a single instance of the class C_2 . In the second type, [5] defines a \mathcal{FD} ($R : \{C_1, \dots, C_n\} \rightarrow C$) between the classes $\{C_1, \dots, C_n\}$ linked to a root class R by properties (or properties chains) and a class C linked to the root class by a property (or properties chain), if the instances of the n classes determines a single instance of the class C .

Since, \mathcal{FD} are a part of ontologies (ontological concept), we propose to extend the initial formal model of ontologies by considering \mathcal{FD} . To do so, we first formalise the \mathcal{FD} in a similar way as [5]. A \mathcal{FD} is composed of the following elements: the *left part LP*, the *right part RP* and a *root class R*. This definition can also be expressed as an implication, as in traditional \mathcal{FD} : $fd R : LP \rightarrow RP$ with R a class, $LP = \{p_{1,1}, \dots, p_{n,m_n}\}$ a set of paths (properties chains) and $RP = \{p_1, \dots, p_l\}$ a path (chain of properties). The left part LP is a set of n paths. A path is composed of a chain of properties, each one being p_i . The right part is defined by a single path, which is composed of l properties. The root class

¹ A simple property is a property with atomic range.

R is the starting point of all paths in the left part and the right part, so that a \mathcal{FD} expresses relationships among properties of a single instance of the class R .

After the \mathcal{FD} formalisation, we propose to extend the initial ontology model proposed in Section 2 as follows: $\mathcal{O} : \langle \mathcal{C}, \mathcal{P}, \text{Sub}, \text{Applic}, \mathcal{FD} \rangle$, where \mathcal{FD} is a binary relationship $\mathcal{FD}: \mathcal{C} \rightarrow (2^{\mathcal{P}} \times 2^{\mathcal{P}} \times \dots \times 2^{\mathcal{P}}, 2^{\mathcal{P}})$ which associates to each class c of \mathcal{C} , the set of the functional dependencies (LP, RP) , where the class c is the root ($fd_c : LP \rightarrow RP$).

Note that reconciliation of query results in a mediator architecture leads to four possible cases: (1) manual reconciliation based on the experience and deep knowledge of data sources of designers which is practically impossible in the real life, where a large number of sources is involved. (2) Only sources having common identifiers are taken into consideration to process queries. In this case, mediator may propagate the query on sources having the common identifiers. This solution compromises the quality of returned results. (3) Query results are merged, where some instances overlap which may cause error. (4) Overlapping instances may be discarded using probabilistic reconciliation.

The presence of \mathcal{FD} may help and facilitate the data reconciliation, especially when no common identifier is used by various sources. To illustrate this point, let us consider the following example.

Example 1. Let S_1 , S_2 and S_3 be three sources containing the same relation *Customer*. With different properties as follows: $S_1.Customer$ ($id(PK)$, $name$, $address$, $phoneNumber$), $S_2.Customer$ ($id(PK)$, $name$, $phoneNumber$) and $S_3.Customer$ ($phoneNumber(PK)$, $name$, $address$). On this table, the following \mathcal{FD} are defined: $fd_1 : Customer : id \rightarrow name$, $fd_2 : Customer : id \rightarrow address$, $fd_3 : Customer : id \rightarrow phoneNumber$, $fd_4 : Customer : phoneNumber \rightarrow name$, $fd_5 : Customer : phoneNumber \rightarrow address$.

This table has two candidate keys: id and $phoneNumber$. Suppose that the mediator schema contains a *Customer* relation with the following properties: $id(PK)$, $name$, $address$. Suppose the following query: "list names and addresses of all customers". The mediator decomposes this query on the three sources. Without \mathcal{FD} , we cannot reconcile all sources, since the source S_3 has different identifier. By using $fd_4 : phoneNumber \rightarrow name$ and $fd_5 : phoneNumber \rightarrow address$, we notice that the attribute $phoneNumber$ is a common candidate key between the three sources. Therefore, a reconciliation of the results coming from these three sources becomes possible.

4 Our Integration Methodology

Contrary to classical systems which require the existence of all sources before running the integration process, our system integrates the sources on their arrival. The mediator global schema is incrementally enriched when considering a new source. Different modules composing our integration system are described in Figure 1: (1) an OBDB repository, (2) a user interface, (3) a query engine and (4) a result reconciliator.

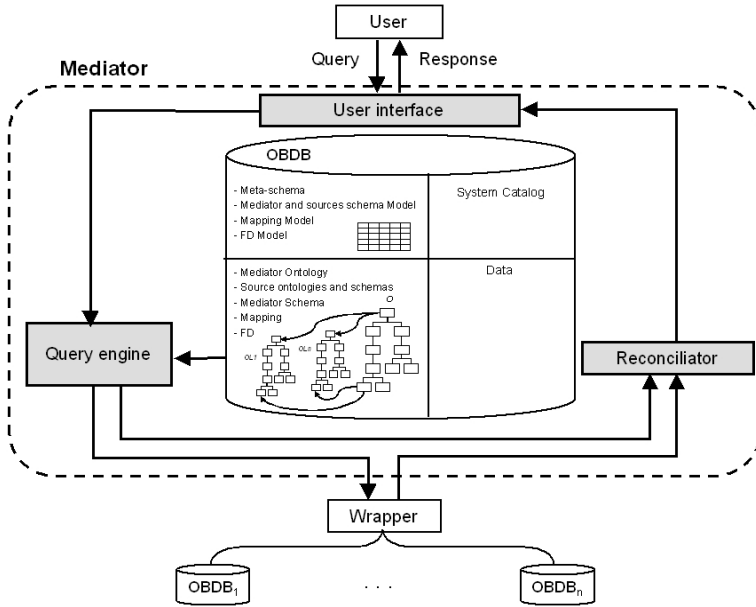


Fig. 1. Different Components of our Integration System

4.1 The OBDB Repository

Our mediator uses the same structure as the used sources participating in the integration process. It follows OntoDB model [7], where two parts are well identified (Section 2) (Figure 4). The meta-schema part of this repository contains (i) a mediator and source schema model, (ii) a model of mapping between the mediator ontology and source ontologies and (iii) a \mathcal{FD} model. In the ontology part, we store the mediator ontology, source ontologies and schemas, the mapping between the mediator ontology and source ontologies and the \mathcal{FD} between the classes and properties of the mediator ontology. The data part can be used as a caching for optimise queries (in this work this issue is not addressed).

Formally, the ontology part is defined as a triplet $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

1. $\mathcal{G} : \langle O, Sch \rangle$ is the global schema which is composed of the mediator ontology $O : \langle C, P, Applic, Sub, FD \rangle$ and the schema Sch of this ontology classes. $Sch : C \rightarrow 2^P$ associates to each mediator ontology class c of C the properties describing the instances of the class c , which are valuated in at least one integrated source.
2. \mathcal{S} is the set of source schemas, where each source schema is defined as a couple $S_i : \langle OL_i, SchL_i \rangle$. $O_i : \langle C_i, P_i, Applic_i, Sub_i, FD_i \rangle$ is the ontology of S_i and $SchL_i : CL_i \rightarrow 2^{PL_i}$ is the schema of the OL_i ontology classes.
3. \mathcal{M} is the mapping between the classes of mediator ontology \mathcal{O} and the classes of source ontologies OL . $\mathcal{M} : C \rightarrow 2^{\{CL_1 \cup \dots \cup CL_n\}}$ associates to each

mediator ontology class c of C the classes of source ontologies in correspondence with the class c .

Before starting the integration process, the integration system components are initialized as follows: The mediator ontology O is imported by selection of classes and properties from a shared ontology $O_s : \langle C_s, P_s, Applic_s, Sub_s, FD_s \rangle$. The process is done following these steps:

1. Starting from user's requirements, the administrator of the mediator selects classes and properties from the shared ontology.
2. The selected classes and properties are added to the mediator ontology.
3. If an imported class has a super class, its super class is imported also, and so on if its super class has a super class until attaining the root class in the hierarchy.
4. To keep the semantic of complex properties (object properties), the importation of a complex property involves the importation of its range class.
5. Likewise, to keep \mathcal{FD} , the importation of a property appearing in a right part of a \mathcal{FD} implies the importation of all the properties of the left part of this \mathcal{FD} .

The schema Sch , the source schemas \mathcal{S} and the mapping \mathcal{M} are initially empty ($Sch(c) = \phi \forall c \in C$, $\mathcal{S} = \phi$ and $\mathcal{M}(c) = \phi \forall c \in C$).

Integrating a new source. the mediator ontology is determined only once whereas the schema Sch , the source schemas \mathcal{S} and the mapping \mathcal{M} are updated after each integration of a new source. The integration of a new source $S_i : \langle O_i, I_i, Sch_i, Pop_i \rangle$ is run in the following steps:

1. We update the source schemas \mathcal{S} by adding the schema of the new source ($\mathcal{S} = \mathcal{S} \cup S_i : \langle OL_i, SchL_i \rangle$).
2. In the ontology $OL_i : \langle CL_i, PL_i, SubL_i, ApplicL_i \rangle$, we keep only classes and properties existing in the mediator ontology ($OL_i = O \cap O_i$).
3. We import the schemas of the OL_i classes from Sch_i ($\forall c \in CL_i \ SchL_i(c) = Sch_i(c)$).
4. We update the schema of the mediator ontology classes Sch by adding the properties valued in S_i to the schema of their classes ($Sch(c) = Sch(c) \cup \{p \in SchL_i(c)\}$).
5. We update the mapping of the mediator classes by adding the mapping between the classes of the mediator ontology O and the classes of the new source ontology OL_i ($\forall c \in CL_i \ \mathcal{M}(c) = M(c) \cup S_i.c$).

4.2 The User Interface

It allows the user to express her/his query and displays its results. After pressing the input query, the user interface send to the query engine a Select-Project-Join queries defined on a set of classes and properties of the mediator ontology.

4.3 The Query Engine

The query engine performs the following tasks for a given user query Q_i . Let $Proj$ be the set of projected properties used in Q_i . Among this set, two types of FD may be distinguished: (1) *direct FD* already exist in the mediator ontology, where a property from $Proj$ exists in the right part of FD and (2) *generated FD* obtained using a similar algorithm of [15]. The presence of both types of dependencies allows us to generate the reconciliation key. The query engines identifies then the relevant sources and rewrites a global query defined on mediator ontology in local queries defined in sources, where each one is sent to relevant sources. Finally, the reconciliator merges the results using the reconciliation key and sends the final result to the user interface.

5 Validation of Our Architecture

To validate the feasibility and efficiency of our system, we conduct experiments using dataset of Lehigh University Benchmark (LUBM) and its 14 queries². The used ontology of LUBM has 45 classes and 32 properties (including 25 object properties, and 7 data type properties). Based on this ontology a set of ontology-based databases is generated. All experiments have been carried out on an Intel Pentium IV machine, with 3,2 GHz processor clock frequency, equipped with 1 Gb of RAM, under the operating system Windows XP professional. Two main

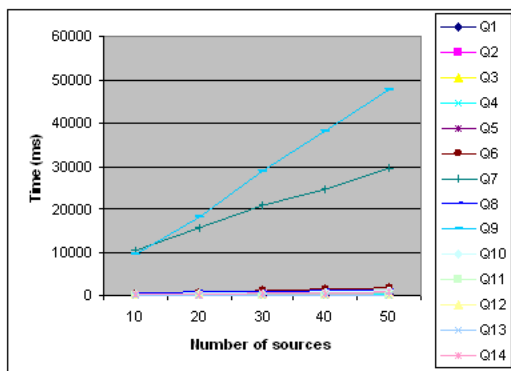


Fig. 2. Query Response Time vs. Number of Sources

experiments are conducted to evaluate the scalability of our system based on the number of sources and instances. Figure 2 shows the results of executing 14 queries in (millisecond) by varying the number of sources participating in the integration process from 10 to 50. Generated sources have the same schema (22 relations). The biggest relation has 2000 tuples. The obtained results show

² <http://swat.cse.lehigh.edu/projects/lubm/>

that our system executes efficiently queries involving a small set of classes (less joins) (e.g. $Q_{14}(x):- \text{UndergraduateStudent}(x)$), but, for queries involving large number of classes (e.g. $Q_9(x):- \text{Student}(x), \text{Faculty}(y), \text{Course}(z), \text{advisor}(x, y), \text{takesCourse}(x, z), \text{takesCourse}(y, z)$), the response time is quite high, but still reasonable. An interesting issue from this result is to separate the query response time into two parts: mediator processing time (including finding the functional dependencies that hold in the query, deriving the reconciliation key and reconciliation of result) and local query evaluation.

In the same direction of the previous experiment, we conduct another one, by considering low costly and high costly queries (Q7 and Q9) and varying the number of instances of 10 used sources. Figure 3 shows the obtained results. An interesting result came from the query 10 considered as low costly query in the first experiments, but when the number of instances increases, join operation becomes costly where it becomes costly. This shows the query response time depends heavily on the sources and their ability of processing queries and not on the mediator.

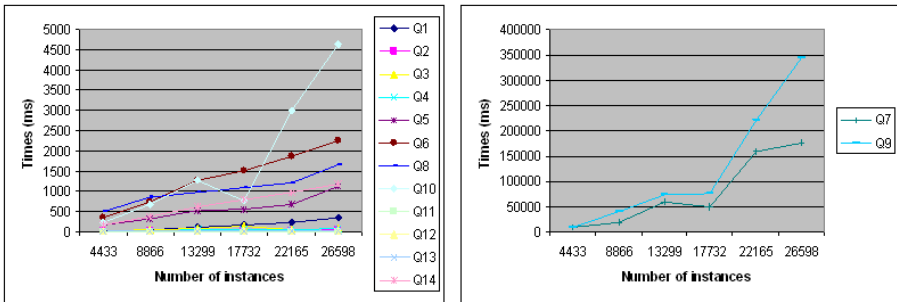


Fig. 3. Query Response Time vs. Number of Sources

6 Conclusion

The need for developing semantic integration systems increases with the evolution of domain ontologies in various systems such as engineering, medicine, etc. The presence of ontologies contributes largely in solving heterogeneity of sources. Some actual integration systems suppose that the manipulated sources have similar keys to ensure data integration which violates the autonomy characteristic of sources. Others use statistical techniques to reconcile data. In sensitive domains, such techniques cannot be used. In this paper, we proposed a complete ontology-based integration method that covers most important phases of system integration life cycle. The presence of ontology contributes for both reducing heterogeneity and offering mechanisms for data reconciliation, since it is enriched by functional dependencies defined on each ontology class. Our approach is evaluated using the dataset of Lehigh University Benchmark. The obtained results show its efficiency and feasibility.

Two main issues that arise from our *preliminary work* should be explored: (i) conducting of a large scale evaluation to measure the real efficiency of our system and (ii) defining of metrics to measure the quality of our integration system.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases (1995)
2. Bellatreche, L., Ait Ameur, Y., Chakroun, C.: A design methodology of ontology based database applications. *Logic Journal of the IGPL*, 1–18 (2010)
3. Bellatreche, L., Xuan, D.N., Pierra, G., Dehainsala, H.: Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry Journal Elsevier* 57(8-9), 711–724 (2006)
4. Bleiholder, J., Naumann, F.: Data fusion. *ACM Computing Surveys* 41(1), 1–41 (2008)
5. Calbimonte, J.P., Porto, F., Maria Keet, C.: Functional dependencies in owl abox. In: *Brazilian Symposium on Databases (SBBDB)*, pp. 16–30 (2009)
6. Calvanese, D., Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: *Proc. of IJCAI*, pp. 155–160 (2001)
7. Dehainsala, H., Pierra, G., Bellatreche, L.: OntoDB: An ontology-based database for data intensive applications. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) *DASFAA 2007. LNCS*, vol. 4443, pp. 497–508. Springer, Heidelberg (2007)
8. Dong, X.L., Naumann, F.: Data fusion - resolving data conflicts for integration. *PVLDB* 2(2), 1654–1655 (2009)
9. Fan, W.: Dependencies revisited for improving data quality. In: *PODS*, pp. 159–170 (2008)
10. Goh, C.H., Bressan, S., Madnick, E., Siegel, M.D.: Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems* 17(3), 270–293 (1999)
11. Hakimpour, F., Geppert, A.: Global Schema Generation Using Formal Ontologies. In: Spaccapetra, S., March, S.T., Kambayashi, Y. (eds.) *ER 2002. LNCS*, vol. 2503, pp. 307–321. Springer, Heidelberg (2002)
12. Halevy, A.Y., Ashish, N., Bitton, D., Carey, M.J., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise information integration: successes, challenges and controversies. In: *SIGMOD*, pp. 778–787 (2005)
13. Hong, J., Liu, W., Bell, D.A., Bai, Q.: Answering queries using views in the presence of functional dependencies. In: Jackson, M., Nelson, D., Stirk, S. (eds.) *BNCOD 2005. LNCS*, vol. 3567, pp. 70–81. Springer, Heidelberg (2005)
14. Mena, E., Kashyap, V., Sheth, A.P., Illarramendi, A.: Observer: An approach for query processing in global information systems based on interoperability across pre-existing ontologies. In: *CoopIS*, pp. 14–25 (1996)
15. Mohania, M.K., Radha Krishna, P., Pavan Kumar, K.V.N.N., Karlapalem, K., Vincent, M.W.: Functional dependency driven auxiliary relation selection for materialized views maintenance. In: *COMAD* (2005)
16. Romero, O., Calvanese, D., Abello, A., Rodriguez-Muro, M.: Discovering functional dependencies for multidimensional design. In: *ACM 12th Int. Workshop on Data Warehousing and OLAP* (2009)
17. Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. *Journal of Data Semantics* 12, 66–94 (2009)
18. Toman, D., Weddell, G.E.: On keys and functional dependencies as first-class citizens in description logics. *J. of Automated Reasoning* 40(2-3), 117–132 (2008)

A Comprehensive Framework on Multidimensional Modeling

Oscar Romero and Alberto Abelló

Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{aabello,oromero}@essi.upc.edu

Abstract. In this paper we discuss what current multidimensional design approaches provide and which are their major flaws. Our contribution lays in a comprehensive framework that does not focus on *how* these approaches work but *what* they do provide for usage in real data warehouse projects. So that, we do not aim at comparing current approaches but set up a framework (based on four criteria: the role played by end-user requirements and data sources, the degree of automation achieved and the quality of the output produced) highlighting their drawbacks, and the need for further research on this area.

1 Introduction

Developing a data warehousing system is never an easy job, and raises up some interesting challenges. One of these challenges focuses on modeling multidimensionality. OLAP tools are conceived to exploit the data warehouse for analysis tasks based on the multidimensional (MD) paradigm and therefore, the data warehouse must be structured according to the MD model. Lots of efforts have been devoted to MD modeling, and several models and design methods have been developed and presented in the literature. Consequently, we can nowadays design a MD conceptual schema, create it physically and later, exploit it through the model algebra or calculus (implemented in the exploitation tools).

MD modeling was first introduced by Kimball in [9]. Kimball's approach was well received by the industry and also introduced the first method to derive the data warehouse *logical* schema. Similar to traditional information systems modeling, Kimball's method is *requirement-driven*: it starts eliciting business requirements of an organization and through a step-by-step guide we are able to derive the MD schema. Only at the end of the process data sources are considered to map data from sources to target.

In short, Kimball's approach follows a traditional modeling approach (i.e., from requirements), but it set down the principles of MD modeling. MD modeling is radically opposite to OLTP systems modeling: the data warehouse conceptual schema is directly derived from the organization operational sources and provides a single, detailed, integrated and homogenized view of the business domain. Consequently, the data warehouse can be thought as a strategic view of the

organization data and for this reason, and unlike most information systems that are designed from *scratch*, the organization data sources must be considered as first-class citizens in the data warehouse design process. This major additional requirement has such interesting consequences so much so that it gave rise to a new research topic and up to now, several MD modeling methods have been introduced in the literature. With the perspective of time, we may now highlight those features that drew the attention of the community. The evolution of the modeling methods introduced in the literature pivots on a crucial aspect: the dichotomy requirements versus data sources (and how to deal with it).

In this paper we discuss what current approaches provide and which are their major flaws. Our contribution lays in a comprehensive framework that does not focus on *how* these approaches work but *what* they do provide. Importantly, note that by no means we aim at comparing current approaches but providing a comprehensive, general picture on MD modeling and identify what is (yet) missing on this area. The criteria used for this analysis can be summarized as follows: the role played by end-user requirements and data sources for each method, the degree of automation achieved and the quality of the output produced (i.e., which MD concepts and features do they really consider). The use of this criteria is justified by the conclusions drawn by a previous, exhaustive analysis of current design methods that can be found in [17].

The paper is structured as follows. Section 2 summarizes briefly our previous research on MD design and highlights how this area evolved with time. Next, Section 3 provides a detailed, comprehensive discussion of what can be achieved by using current approaches in real projects. We wrap up the discussion pointing out the main flaws that still need to be addressed in order to better support the data warehouse design.

2 Multidimensional Modeling

In this section we introduce the background of MD modeling. Our objective here is to provide an insightful view of how this area evolved with time. The interested reader is addressed to [17] for details.

Shortly after Kimball introduced his ad hoc modeling method for data warehouses [10], some other methods were presented in the literature (e.g., [4,6,2,7,12]). Like Kimball's method, these methods were originally regarded as step-by-step guides to be followed by a data warehouse expert who start gathering the end-user requirements. However, unlike Kimball's work, they give more and more relevance to the data sources. Involving the data sources in these approaches means that it is compulsory to have well-documented data sources (e.g., with up-to-date conceptual schemas) at the expert's disposal but it also entailed two main benefits: on the one hand, the user may not know all the potential analysis contained in the data sources and analyzing them we may find unexpected potential analysis of interest to the user; on the other hand, we should assure that the data warehouse can be populated with data available within the organization.

As said, to carry out these approaches manually it is compulsory to have well-documented data sources, but in a real organization, the data sources

documentation may be incomplete, incorrect or may not even exist [6] and, in any case, it would be rather difficult for a non-expert designer to follow these guidelines. Indeed, when automating this process is essential not to depend on the expert's ability to properly apply the method chosen and to avoid the tedious and time-consuming task (even unfeasible when working over large databases) of analyzing the data sources.

In order to solve these problems, several new methods automating the design task were introduced in the literature [14,21,8]. These approaches work directly over relational database logical schemas. Thus, despite they are restricted to a specific technology, they get up-to-date data that can be queried and managed by computers. They also argue that restricting to relational technology makes sense since nowadays it is the most widely used technology for operational databases. About the process carried out, these methods follow a *data-driven* process focusing on a thorough analysis of the data sources to derive the data warehouse schema in a reengineering process. This process consists of techniques and design patterns that must be applied over the data sources schema to identify data likely to be analyzed from a MD perspective.

Nevertheless, a requirement analysis phase is crucial to meet the user needs and expectations [3,5,22,15,11,1]. Otherwise, the user may find himself frustrated since he / she would not be able to analyze data of his / her interest, entailing the failure of the whole system. Today, it is assumed that the ideal scenario to derive the data warehouse conceptual schema embraces a hybrid approach (i.e., a combined data-driven and requirement-driven approach) [22]. Then, the resulting MD schema will satisfy the end-user requirements and it will have been conciliated with the data sources simultaneously.

According to [22], MD modeling methods may be classified within a *demand-driven*, a *supply-driven* or a *hybrid* framework:

- Supply-driven approaches (SDAs): Also known as *data-driven*, start from a detailed analysis of the data sources to determine the MD concepts in a reengineering process.
- Demand-driven approaches (DDAs): Also known as *requirement-driven* or *goal-driven*, focus on determining the user MD requirements (as typically performed in other information systems) to later map them onto data sources.
- Hybrid approaches: Combine both paradigms to design the data warehouse from the data sources but bearing in mind the end-user requirements.

Each paradigm has its own advantages and disadvantages. Carrying out an exhaustive search of dimensional concepts among all the concepts of the domain (like SDAs do) has a main benefit with regard to those approaches that derive the schema from requirements and later conciliate them with the data sources (i.e., DDAs): in many real scenarios, the user may not be aware of all the potential analysis contained in the data sources and, therefore, overlook relevant knowledge. Demand-driven and current hybrid approaches do not consider this and assume that requirements are exhaustive. Thus, knowledge derived from the sources not depicted in the requirements is not considered and discarded.

As a counterpart, SDAs tend to generate too many results (since they overlook the MD requirements, they must apply their design patterns all over the data sources) and mislead the user with non relevant information. Furthermore, DDAs (or demand-driven stages within a hybrid approach) are not automated whereas supply-driven stages tend to facilitate their automation. The main reason is that demand-driven stages would require to formalize the end-user requirements (i.e., translate them to a language understandable by computers). Unfortunately, most current methods handle requirements mostly stated in languages (such as natural language) lacking the required degree of formalization. Thus, matching requirements over the data sources must be performed manually. However, the time-consuming nature of this task can render it unfeasible when large databases are used.

In general, most approaches do not automate the process and just present a set of steps (i.e., a guideline) to be followed by an expert in order to derive the MD schema. Mainly, these methods introduce different patterns or heuristics to discover concepts likely to play a MD role and to carry out these approaches manually it is compulsory to have well-documented data sources at the expert's disposal. This prerequisite is not easy to fulfill in many real organizations and in order to solve this problem, current automatable methods directly work over relational databases (i.e., getting up-to-date data). To our knowledge, only three exceptions exist to this rule [20,19,13], which automate the process from ER schemas (the first one) and ontologies (the other two). Consequently, all these methods (or stages within hybrid approaches) follow a supply-driven paradigm and thus, rely on a thorough analysis of the sources.

All in all, *DDAs assume that requirements are exhaustive*, whereas *SDAs rely on discovering as much MD knowledge as possible*. As a consequence, *SDAs generate too many results*. Furthermore, *current automatable methods follow a SDA*, whereas *current DDAs overlook the process automation*, since they tend to work with requirements at a high level of abstraction. Finally, all *current hybrid approaches follow a sequential approach with two well-differentiated steps*: the supply-driven and the demand-driven stages. Each one of these stages, however, suffers from the same drawbacks as pure SDAs or DDAs do.

2.1 The State of the Art in a Nutshell

Previous experiences in the data warehouse field have shown that the data warehouse MD conceptual schema must be derived from a hybrid approach: i.e., by considering both the end-user requirements and the data sources, as first-class citizens. Like in any other system, requirements guarantee that the system devised meets the end-user necessities. In addition, since the data warehouse design task is a reengineering process, it must consider the underlying data sources of the organization: (i) to guarantee that the data warehouse must be populated from data available within the organization, and (ii) to allow the end-user discover unknown additional analysis capabilities.

Nowadays, we may find several methods for supporting the data warehouse conceptual design but, all of them, start from very different assumptions that

make them hardly comparable. For example, some approaches claim to fully automate the design task, but they do so by overlooking the end-user requirements in a fully SDA (and thus, making the user responsible for manually filtering the results obtained according to his / her needs). Similarly, exhaustive DDAs claim to derive high-quality outputs, but they completely overlook the task automation. For this reason, every approach fits to a narrow-ranged set of scenarios and do not provide an integrated solution for every real-world case, which, in turn, makes the data warehouse designers to come up with ad hoc solutions for each project. For example, we cannot follow the same approach in a scenario where the end-user requirements are clear and well-known, and in a scenario in which the end-user requirements are not evident or cannot be easily elicited (e.g., this may happen when the users are not aware of the analysis capabilities of their own sources). Clearly, this is the major flaw of current approaches, which do not suit well for the wide range of real projects a designer could meet. Interestingly, it has already been pointed out [18] that, given a specific design scenario, the necessity to provide requirements beforehand is smoothed by the fact of having semantically rich data sources. In lack of that, requirements gain relevance to extract the MD knowledge from the sources. In both cases, we can still achieve an acceptable degree of automation and output quality, as discussed later on.

3 The Big Picture

To overcome the situation above discussed, we aim to establish a clear framework in which to place the most relevant design methods introduced in the literature. This comprehensive picture of the state of the art will help to identify the major drawbacks of current approaches. As earlier introduced in this paper, the data warehouse design task must consider (i) the end-user requirements and (ii) the data sources. Furthermore, we also aim to analyze the (iii) automation degree achieved and (iv) the quality of the output produced. In the following, we rate the most relevant methods introduced in the literature with regard to these four criteria. However, we do not intend to classify nor rank approaches (for a detailed comparison on MD design methods, see [17]) but to identify, at first sight, the assumptions made by each kind of approach and moreover, analyze the consequences of the process proposed regarding the automation degree achieved and the quality of the outputs produced. In short, identify trends and flaws that should be addressed in the future.

Consider Figures 1 and 2. Axes x and y represent the assumptions of each method; i.e., the use of the requirements and the data sources in each approach. The x axis measures how important requirements are in the approach, and if the method proposes to formalize them somehow, to facilitate their analysis. The y axis assesses how important the analysis of the data sources is for the approach, and if detailed patterns are provided to exploit them. Finally, axes z measure either the automation degree achieved (see Figure 1) and the quality of the output produced (see Figure 2) regarding the assumptions made by the method (i.e., axes x and y). In the 3D-space formed, every approach is identified

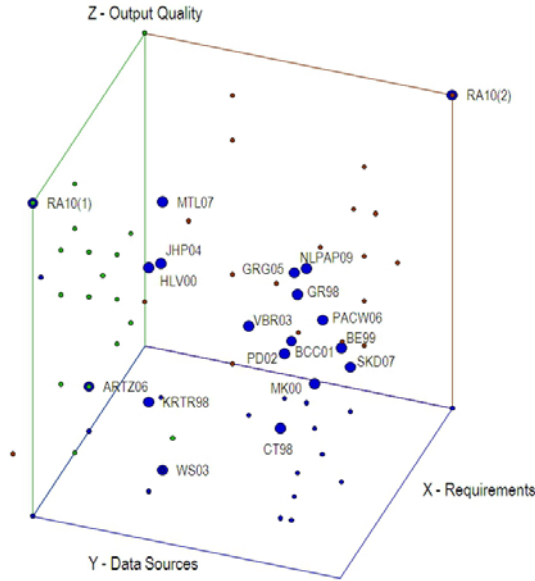


Fig. 1. Output quality analysis

as a rhombus labeled with the first initial of each author plus the year of the bibliographical item it represents. Furthermore, for the sake of understandability, we provide the projection of each rhombus in the three planes (green points for the XZ plane projections; blue points for the XY plane and red points for the XZ plane). Each approach is placed in the 3D-space according to the conclusions extracted from [17]. The first conclusion is that the duality requirements / data-sources is clearly shown in both figures, as SDAs and DDAs are placed in opposite axis (to better appreciate it, check the plane projections of each point).

Requirements Specificity: DDAs integrate the end-user requirements, which lead the whole process by exploiting the knowledge of the requirements. Therefore, the quality and expressiveness of the input requirements must be high. On the contrary, at the beginning of the process, SDAs do not need the end-user requirements to work. Indeed, results provided by SDAs, are eventually shaped by the end-user needs a posteriori. In this latter step, the end-user just state his / her requirements by choosing his / her concepts of interest regarding the results provided. Therefore, the user would even be able to state them on-the-fly regarding the output presented to the user by SDAs.

Data Source Expressiveness: SDAs lead the process from a thorough analysis of the data sources and, in general, they ask for high quality inputs capturing the data sources (i.e., relational schema, ER diagram, domain ontology, etc.). In case of inputs at the conceptual level, a mapping between the conceptual schema and the sources as well as means to access the data sources at the instance level are also required. Regarding DDAs, the quality of the inputs is not that relevant

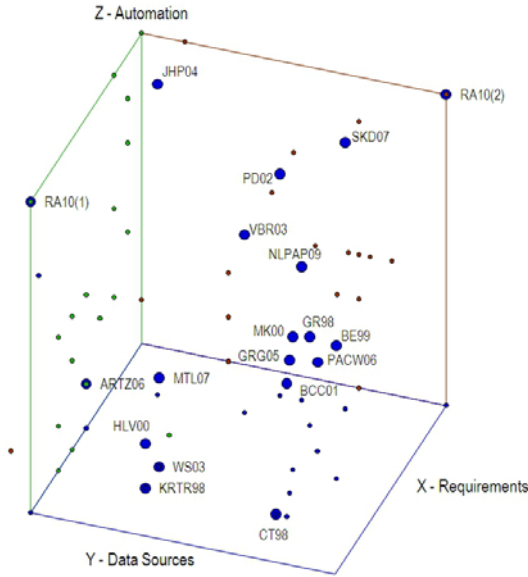


Fig. 2. Automation analysis

given that the requirements provide the lack of semantics captured in the sources. Thus, they could even handle not well-formed data sources (e.g., denormalized sources).

Automation: The first figure shows that the automation degree achieved, in the general case, is medium or low. Only 6 approaches automate the design task up to a fair degree. Regarding DDAs, new techniques to automatically manipulate requirements during the design phase are needed. In this sense, [16] sets a basis on this direction but it can only deal with relational sources. Oppositely, SDAs achieve an interesting degree of automation, but most of them happen not to be useful in practice due to the big set of assumptions made. For example, the kind of sources (normally, only relational sources are allowed but this is clearly unsatisfactory nowadays, where unstructured data and the Web is a relevant source of knowledge) or additional ones (such as relational schemas in, at least, 3NF). Furthermore, filtering techniques based on objective evidences are a must. SDAs tend to generate too many results. Consequently, they unnecessarily overwhelm users with blindly generated combinations whose meaning has not been analyzed in advance. Eventually, they put the burden of (manually) analyzing and filtering results provided onto the designer's shoulder, but the time-consuming nature of this task can render it unfeasible when large data sources are considered. To our knowledge, only [19] filters out results obtained prior to show the results to the user. Furthermore, all these facts directly affect the computational complexity of SDAs.

Quality Output: Both SDAs and DDAs are able to extract valuable knowledge from the requirements / data sources but only a few of them deal with concepts

such as *factless facts*, *MD space definitions*, *aggregate measures* and *semantic relationships* between the MD concepts identified. However, this is not tied to the kind of framework used but a decision made by the authors. Indeed, both can produce quality outputs if the appropriate inputs are provided. This is obviously sound, since, given the same scenario, we would expect to obtain the same result regardless of the approach chosen. The difference between both approaches is just on *how* we obtain the final result depending on our current inputs and if they provide enough semantics. Thus, DDAs are exhaustive regarding requirements so in case of disposing of quality requirements a DDA suits better. However, if requirements are not clear but we dispose of quality data sources then a SDA is mandatory. For example, DDAs can compute *derived measures* or *concept specializations* not explicitly captured in the data sources, but present in the end-user requirements. However, this kind of measures and specializations can only be identified by SDAs if they are captured in the input representation of the data sources (and thus, only if they are of enough quality). Analogously, this also happens regarding *semantic relationships* between MD concepts, *aggregate measures* and *MD space definitions*. The case of the *factless facts*, however, is slightly different. DDAs might identify them by means of requirements, but in SDAs, some kind of *quality function* used to identify facts would be needed [19].

4 Discussion and Conclusions

Several methods for supporting the data warehouse modeling task have been provided. However, they suffer from some significant drawbacks, which need to be addressed. In short, DDAs assume that requirements are exhaustive (and therefore, do not consider the data sources to contain alternative interesting evidences of analysis), whereas SDAs (i.e., those leading the design task from a thorough analysis of the data sources) rely on discovering as much MD knowledge as possible from the data sources. As a consequence, SDAs generate too many results, which misleads the user. Furthermore, the design task automation is essential in this scenario, as it removes the dependency on an expert's ability to properly apply the method chosen, and the need to analyze the data sources, which is a tedious and time-consuming task (which can be unfeasible when working with large databases). In this sense, current automatable methods follow a SDA, whereas current DDAs overlook the process automation, since they tend to work with requirements at a high level of abstraction. Indeed, this scenario is repeated regarding SDA and DDA stages within current hybrid approaches, which suffer from the same drawbacks than pure DDA and SDA approaches.

Consequently, previous experiences in this field have shown that the data warehouse MD conceptual schema must be derived from a truly hybrid approach: i.e., by considering both the end-user requirements and the data sources, as first-class citizens. Currently, several methods (i.e., detailed design approaches) and dissertations (i.e., high level discussions highlighting the necessities in each real scenario) for supporting the data warehouse design task have been introduced in the literature, but none of them provides an integrated and automated solution

embracing both aspects. On the one hand, dissertations about how the design task must be adapted to every real-world scenario provide an insightful idea of how to proceed in each case. However, they fail to provide detailed algorithms to undertake this task (thus, ad hoc solutions are needed). On the other hand, detailed methods introduced tend to focus on a narrow-ranged set of scenarios. For example, today, it is assumed that the approach to follow in a scenario where the end-user requirements are clear and well-known is completely different from that in which the end-user requirements are not evident or cannot be easily elicited (for example, this may happen when the users are not aware of the analysis capabilities of their own sources). Similarly, the necessity to provide requirements beforehand is smoothed by the fact of having semantically rich data sources. In lack of that, requirements gain relevance to extract the MD knowledge from the sources. Indeed, a combined and comprehensive framework to decide, according to the inputs provided in each scenario, which is the best approach to follow, is missing. This framework should be built considering that:

- If the end-user requirements are well-known beforehand, we can benefit from the knowledge captured in the data sources, but we should guide the design task according to requirements and consequently, we will be able to work and handle semantically poorer data sources. In other words, providing high-quality end-user requirements, we can guide the process and overcome the fact of disposing of bad quality (from a semantical point of view) data sources.
- As a counterpart, a scenario in which the data sources available are semantically richer, the approach should be guided by a thorough analysis of the data sources, which eventually will be properly adapted to shape the output result and meet the end-user requirements. In this context, disposing of high-quality data sources we can overcome the fact of lacking of very expressive end-user requirements.

Acknowledgements. This work has been partly supported by the Ministerio de Ciencia e Innovación under project TIN2008-03863.

References

1. Annoni, E., Ravat, F., Teste, O., Zurfluh, G.: Towards multidimensional requirement design. In: DaWaK 2006. LNCS, vol. 4081, pp. 75–84. Springer, Heidelberg (2006)
2. Böhnlein, M., vom Ende, A.U.: Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems. In: Proc. of 2nd Int. Wksp on Data Warehousing and OLAP, pp. 15–21. ACM, New York (1999)
3. Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A., Paraboschi, S.: Designing Data Marts for Data Warehouses. *ACM Trans. Soft. Eng. Method* 10(4), 452–483 (2001)
4. Cabibbo, L., Torlone, R.: A Logical Approach to Multidimensional Databases. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) *EDBT 1998*. LNCS, vol. 1377, pp. 183–197. Springer, Heidelberg (1998)

5. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented Requirement Analysis for Data Warehouse Design. In: Proc. of 8th Int. Wksp on Data Warehousing and OLAP, pp. 47–56. ACM Press, New York (2005)
6. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Int. Journal of Cooperative Information Systems* 7(2-3), 215–247 (1998)
7. Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual Data Warehouse Modeling. In: Proc. of 2nd Int. Wksp on Design and Management of Data Warehouses, p. 6. CEUR-WS.org (2000)
8. Jensen, M.R., Holmgren, T., Pedersen, T.B.: Discovering Multidimensional Structure in Relational Data. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) *DaWaK 2004*. LNCS, vol. 3181, pp. 138–148. Springer, Heidelberg (2004)
9. Kimball, R.: *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley & Sons, Inc., Chichester (1996)
10. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M.: *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. John Wiley & Sons, Inc., Chichester (1998)
11. Mazón, J., Trujillo, J., Lechtenborger, J.: Reconciling Requirement-Driven Data Warehouses with Data Sources Via Multidimensional Normal Forms. *Data & Knowledge Engineering* 23(3), 725–751 (2007)
12. Moody, D., Kortink, M.: From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. In: Proc. of 2nd Int. Wksp on Design and Management of Data Warehouses. CEUR-WS.org (2000)
13. Nebot, V., Llavori, R.B., Pérez-Martínez, J.M., Aramburu, M.J., Pedersen, T.B.: Multidimensional integrated ontologies: A framework for designing semantic data warehouses. *J. Data Semantics* 13, 1–36 (2009)
14. Phipps, C., Davis, K.C.: Automating Data Warehouse Conceptual Schema Design and Evaluation. In: Proc. of 4th Int. Wksp on Design and Management of Data Warehouses., vol. 58, pp. 23–32. CEUR-WS.org (2002)
15. Prat, N., Akoka, J., Comyn-Wattiau, I.: A UML-based Data Warehouse Design Method. *Decision Support Systems* 42(3), 1449–1473 (2006)
16. Romero, O., Abelló, A.: Automatic Validation of Requirements to Support Multidimensional Design. *Data & Knowledge Engineering* 69(9), 917–942 (2010)
17. Romero, O., Abelló, A.: A Survey of Multidimensional Modeling Methodologies. *Int. J. of Data Warehousing and Mining* 5(2), 1–23 (2009)
18. Romero, O.: Automating the Multidimensional Design of Data Warehouses. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain (2010), <http://www.tdx.cat/handle/10803/6670>
19. Romero, O., Abelló, A.: A Framework for Multidimensional Design of Data Warehouses from Ontologies. *Data & Knowledge Engineering* 69(11), 1138–1157 (2010)
20. Song, I., Khare, R., Dai, B.: SAMSTAR: A Semi-Automated Lexical Method for Generating STAR Schemas from an ER Diagram. In: Proc. of the 10th Int. Wksp on Data Warehousing and OLAP, pp. 9–16. ACM, New York (2007)
21. Vrdoljak, B., Banek, M., Rizzi, S.: Designing Web Warehouses from XML Schemas. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) *DaWaK 2003*. LNCS, vol. 2737, pp. 89–98. Springer, Heidelberg (2003)
22. Winter, R., Strauch, B.: A Method for Demand-Driven Information Requirements Analysis in DW Projects. In: Proc. of 36th Annual Hawaii Int. Conf. on System Sciences, pp. 231–239. IEEE, Los Alamitos (2003)

Preface to Variability@ER'11

As software requirements constantly increase in size and complexity, the need for methods, formalisms, techniques, tools and languages for managing and evolving software artifacts become crucial. One way to manage variability when dealing with a rapidly growing variety of software products is through developing and maintaining families of software products rather than individual products. Variability management is concerned with controlling the versions and the possible variants of software systems. Variability management gained a special interest in various software-related areas in different phases of the software development lifecycle. These areas include conceptual modeling, product line engineering, feature analysis, software reuse, configuration management, generative programming and programming language design. In the context of conceptual modeling, the terminology of variability management has been investigated, yielding ontologies, modeling languages, and classification frameworks. In the areas of software product line engineering and feature analysis, methods for developing core assets and efficiently using them in particular contexts have been introduced. In the software reuse and configuration management fields, different mechanisms for reusing software artifacts and managing software versions have been proposed, including adoption, specialization, controlled extension, parameterization, configuration, generation, template instantiation, analogy construction, assembly, and so on. Finally, generative programming deals with developing programs that synthesize or generate other programs and programming language design provides techniques for expressing and exploiting commonality of source code artifacts, but also for specifying the allowed or potential variability, whether it is static or dynamic.

The purpose of this workshop is to promote the theme of variability management from all or part of these different perspectives, identifying possible points of synergy, common problems and solutions, and visions for the future of the area. The workshop accepted 4 papers dealing with variability management related issues:

1. Mohammed Eldammagh and Olga De Troyer. Feature Modeling Tools: Evaluation and Lessons learned.
2. Ateeq Khan, Gunter Saake, Christian Kaestner and Veit Koeppen. Service Variability Patterns.
3. Angela Lozano. An Overview of Techniques for Detecting Software Variability Concepts in Source Code.
4. Jaap Kabbedijk and Slinger Jansen. Variability in Multi-tenant Environments: Architectural Design Patterns from Industry.

The workshop also had an invited talk given by Timo K. Käkölä and entitled "ISO Initiatives on Software Product Line Engineering: Vision and Current Status."

For more information about the workshop, please visit our web site at <http://www.domainengineering.org/Variability@ER11/>

July 2011

Iris Reinhartz-Berger
Arnon Sturm
Kim Mens

ISO Initiatives on Software Product Line Engineering: Vision and Current Status

Invited Talk for Variability@ER2011

Timo K. Käkölä

University of Jyväskylä, Finland
timo.kakola@jyu.fi

Abstract. Software product line engineering and management is still a relatively young discipline and its industrial diffusion is somewhat limited, partly due to the lack of international standards. It involves sub-disciplines that may not necessarily be mature enough yet for standardization. On the other hand, well-focused standardization efforts contribute to the diffusion of best product line practices and help raise the maturity of these practices. Careful roadmapping and execution of the standardization initiative is thus vital. This talk discusses the current roadmap for the international standardization initiative of software product line engineering and management methods and tools and the status of the work done so far in this area in the International Organization for Standardization (ISO). One of the purposes of the talk is to invite discussion about the most critical areas to be standardized and the time frames involved. The role of variability management in the standards will be especially emphasized.

Feature Modeling Tools: Evaluation and Lessons Learned

Mohammed El Dammagh and Olga De Troyer

Vrije Universiteit Brussel, Pleinlaan 2,
1050 Brussels, Belgium

{Mohammed.ElDammagh, Olga.DeTroyer}@vub.ac.be

Abstract. This paper presents an evaluation of feature modeling tools. The purpose of the evaluation was to gain insight in the aspects that influence the quality and more in the particular usability. The evaluation focused on the quality criteria: usability, safety, and the support for functional usability requirements. The study involved 9 feature-modeling tools and was done using an experimental evaluation and an investigation by the authors of the paper. From the results, recommendations are formulated that can be taken into consideration in future tool design for these kind of modeling tools.

Keywords: feature modeling tool, quality, usability, evaluation.

1 Introduction

During the past years, several variability modeling techniques as well as tools have been developed to model variability during domain analysis [1]. Example modeling techniques are: COVAMOF [2], OVM (Orthogonal Variability Modeling) [1], VSL (Variability Specification Language) [3] and FAM (Feature Assembly Modeling) [4]. Notwithstanding the wide range of possible approaches to support variability modeling, feature modeling remains the most commonly used technique for identifying and capturing variability requirements and dependencies between product characteristics. Also, most variability modeling tools support feature modeling [6].

Industrial software product lines are characterized by a rapid growth of the number of variation points, associated variants and dependencies. Previous research points out that there is a lack of adequate tools supporting this increase in variability information [5,6,16]. Moreover, in the case of large-scale software systems, usability and scalability issues quickly grow and become a major source of frustration [7].

Because of the importance of adequate tool support, we want to investigate how the quality of feature modeling tools can be improved. As a first step in this research, the quality of existing tools will be investigated in order to gain insight in the aspects that influence the quality. Later on, we can then try to improve on these aspects. In a first study performed in this context, we have concentrated on evaluating the quality of existing tools from a usability point of view. Scalability issues will be investigated in later work. In this paper, we present the results of this first study. For this study, we selected 9 feature modeling tools [8] using a graphical user interface. These tools are evaluated against the criteria Usability, Safety, and Functional Usability Features.

The rest of the paper is organized as follows. We start by identifying the quality criteria that we will use (Section 2). Next, we describe the methodology of our quality evaluation in Section 3. Whereas Section 4 concerns the setup of the evaluation, Section 5 concentrates on the results. Section 6 discusses the differences in quality compliance of the tools and highlights the most important lessons learned. Next, Section 7 mentions the related work. Finally, Section 8 recaps the main ideas and presents further research.

2 Quality Criteria

For our quality evaluation of tools, we can use the *system quality in use* model, as defined by ISO/IEC 9126, ISO 9241–11 and the ISO/IEC 25000 series of standards. This model offers a broad framework of quality requirements. It consists of three characteristics: *usability*, *flexibility* and *safety*. These are on their turn subdivided into sub-characteristics. Accordingly, usability can be measured by the degree to which a specified group of users conducts certain tasks with *effectiveness*, *efficiency* and *satisfaction*. Whereas effectiveness is defined in terms of “accuracy and completeness”, efficiency is defined in terms of “resources expended” in relation to effectiveness. Satisfaction describes the extent to which users are satisfied. Similarly, flexibility deals with the context within which the tool operates. Safety includes all the potential negative effects resulting from incomplete or incorrect output [9].

For this first study, we decided to focus on usability and safety. We omit flexibility, as in this study the context of use will be fixed. However, some other aspects of flexibility will be measured by considering *functional usability features* (FUFs), which are usability requirements with a major impact on the functionality. According to Juristo [10], these are important to consider because of their high functional implications. Since a relationship between usability and functional requirements has been proven, functional requirements should be taken into consideration in the case that they have an impact on usability attributes.

3 Methodology

The purpose of our study is to measure the quality of existing feature modeling tools using the criteria identified in Section 2. Please note that, in no way, it is our purpose to rank or give a judgment on the tools evaluated. The purpose of the study is to use the information collected to draw some “lessons learned”, and to use this to make recommendations on how to improve the quality of feature modeling tools in general.

3.1 Tools Identification

Firstly, tools were selected on their ability to support feature modeling. Secondly, we selected tools with an interactive graphical user interface (GUI), in which the features, types, feature group and feature dependencies are visualized. We decided to concentrate on tools with a GUI, as the use of a GUI has many advantages [11], especially in the communication with non-technical domain experts.

To select the tools, we mainly based us on the study “A systematic review of domain analysis tools” [6]. Nine of these 19 tools were executable. Seven of the executable tools use feature modeling and thus were selected (CaptainFeature [19], FeatureIDE [21], Pure::Variants [20], RequiLine [22], XFeature [23], GEARS [28] and FeaturePlugin [29]). Of these seven tools, GEARS and FeaturePlugin were abandoned. GEARS was not available to openly evaluate and FeaturePlugin missed an interactive graphical interface. Thus, from the original list of 19 tools, 5 were selected.

However, the study in [6] dated from January 2008, therefore, following the same search strategy as provided in that review, we have added six new tools (MOSKitt [24], Feature Modeling Tool [25], Feature Model DSL [26], CVM Tool [27], FORM [20] and ToolDay [31]). FORM and ToolDay were excluded based on their non-availability. Finally, nine tools were retained, as is shown in the Table 1.

It may be noted that most of the selected tools, except Pure::Variants, were developed for an academic purpose. Three of these tools are designated to industrial as well as academic environments. The fact that most tools are developed for academic purposes, and therefore, should be considered more as prototypes rather than as full-fledged tools, is not an issue. On the contrary, they may better reveal the aspects influencing the quality of a tool, which is the main purpose of the study.

Table 1. Tools

	Tool	Developed by	Used	Released
T1	CaptainFeature	Fachhochschule Kaiserslautern in Germany.	A	2002
T2	Pure::Variants	Pure-Systems' company in Germany.	I	2003
T3	FeatureIDE	Otto-von-Guericke-University Magdeburg in Germany.	A	2005
T4	RequiLine	Research Group Software Construction in Germany.	B	2005
T5	XFeature	An association of P&P Software Company with the Swiss Federal Institute of Technology	B	2005
T6	MOSKitt	gvCASE project by the Valencian Regional Ministry of Infrastructure and Transport in Spain.	A	2008
T7	Feature Modeling Tool	Research Group of GIRO at the University of Valladolid and Burgos in Spain.	A	2008
T8	Feature Model DSL	Gunther Lenz and Christoph Wienands in Practical Software Factories in .NET book.	A	2008
T9	CVM Tool	European project ATESSST in Germany.	B	2009

I. Industrial, A. Academic, B. Both.

3.2 Evaluation Criteria

This study will evaluate whether and to which extent the nine tools support the selected quality criteria, i.e. Usability and Safety, and whether the FUFs are present.

Usability

Efficiency: The “expended resources” as referred to in ISO 9241–11, are in our experiment defined as follows. On one hand, task completion time (TCT) is measured as the time that is needed to model the features, feature groups and feature dependencies. On the other hand, the user’s effort (e) that is needed to complete the task is measured as well. In order to calculate effort, we rely on [12]. Accordingly to this work, effort (e) equals the number of mouse clicks (mc) + the number of keyboard strokes (mk) + the number of mouse pixels traversed by the user - mouse trajectory - (mic). Note that the time and effort to arrange the model according to individual preferences (esthetic aspect) was excluded from the results.

Satisfaction: In order to evaluate “the degree to which users are satisfied”, we base ourselves on the studies of [9] and [13]. Hence, participants were asked to rate the following three statements: (Q1) “You like very much to use the tool”, (Q2) “You find the tool very easy to use” and (Q3) “You strongly recommend the tool to a friend”. A 5-point Likert scale has been used for this (1—strongly disagree to 5—strongly agree).

Note that we will not measure effectiveness. In this study “the accuracy and completeness with which users achieve specified goals” will not be considered as all tools selected, support the creation of an accurate feature model.

Safety. We rely on the work of [14] to measure “all the potential negative effects resulting from incomplete or incorrect output” [9]. The author draws a parallel between inconsistent product configuration (output) and the disability of the tools to provide automatic redundancy, anomaly and inconsistency checks, defined as follows:

Redundancy: Refers to information that is modeled in multiple ways. Redundancy can decrease the maintainability of the model (negative effect), while on the other hand it can increase the readability and understandability of the feature model (positive effect). Hence, redundancy is considered as a light issue.

Anomalies: Refers to the modeling of senseless information. As a consequence, potential configurations are being lost, although these configurations should be possible. This can be considered as a medium issue.

Inconsistency: Refers to contradictory information within a model, i.e. information that is conflicting with some other information in the model. Since inconsistent product configuration can be derived from such a model, it is a severe issue.

Complementary to these three checks, a fourth measure of Safety has been added:

Invalid semantics: Since all the given tools support a feature model based on a tree structure, the feature model should be modeled with respect to certain semantic rules inherent to this tree structure, e.g., each child has only one parent.

In our study, the tools are tested on their compliance with these four safety checks by using a rating scale from 0 to 3 (0—not supported to 3—fully supported).

Functional Usability Features. As motivated in Section 2, functional requirements having a major impact on usability should be included in our evaluation. As a result, 7 FUFs have been selected from the initial lists of [10] according to their relevance:

- Feedback: To inform the user about actions with important consequences.
- Undo: To undo the user’s action in order to save time to recover from mistakes and to make it more pleasant to work with the tools.

- Form/Field Validation: To improve data input and software correction as soon as possible.
- Shortcuts: To allow the user to activate a task with one quick gesture.
- User Expertise: To customize the functionalities of the tool to the users' preferences and experience. E.g., does the user have more than one way to perform his action, or is the user able to create his own shortcuts.
- Reuse information (cut, copy and paste): To allow the user to easily move data.
- Help: To provide different support.

To evaluate the FUFs, the same rating scale as to measure Safety is used.

4 Setup of the Evaluation

The evaluation has been achieved in two steps: an experimental evaluation followed by a questionnaire, and an investigation performed by the authors of this paper.

As our main purpose was to obtain a better understanding of how and to which extent the existing tools offer quality support during the modeling process, we opted for a small-scale experiment. Five persons were asked to create a small feature diagram having 20 features, two feature groups and two feature constraints. Although five participants is a small number, it has been shown that five participants are sufficient to discover 85% of the usability problems [15]. Next, we opted for a small feature diagram, as scalability will be studied separately.

We divided the experiment into three main sub processes: (1) specifying features (refers to “feature and its type”), (2) specifying feature groups (“alternative” or “or-relationship”), and (3) specifying feature constraints. Each of these sub processes is measured against the criteria for efficiency, i.e. task completion time and effort. In order to record all the activities of the participants, programs recording screen activities have been used, i.e. ALLCapture [32] and WinOMeter [33].

The participants were 4 PhD students, and 1 PhD graduated in computer science, having good modeling experience and skills. To make them acquainted with the tools, a demo for each tool was made and handed over in advance to the participants. This demo explains all the functionalities to know in order to do the experiment.

The 9 tools were evaluated in 3 sessions (three tools in one session). Each day one session took place in order not to overload the participants. As mentioned earlier, satisfaction was measured by using a questionnaire that was handed over after the usage of the tool. The participants also had the opportunity to justify their answers.

Safety and the FUFs were evaluated by the authors themselves. As to measure Safety, the authors tested in the first place if the tools possess a check for redundancy, anomalies, inconsistency, and invalid semantics. Secondly, a situation analogue to each of the four deficiencies has been modeled with each tool. The results were given using a rating scale from 0 to 3. The same rating scale has been used for the FUFs. The authors examined if each of the FUFs, as listed earlier, were present in the tools.

5 Results

We discuss the results of the evaluation following the evaluation criteria:

- Usability

Efficiency: One-way analyses of variance (ANOVAs) were conducted to outline whether the tools differ significantly from each other in terms of ‘task completion time’ and effort spent. The independent variable consists of the different tools, while the dependent variables are task completion time (TCT) and effort. Both dependent variables are measured for each of the sub processes. The results of these ANOVAs indicate that there is a significant difference among the tools.

Additionally, Tukey’s HSD [35] range test was used to identify homogeneous subsets of means that are not significant differently from each other. It allows us to group tools together and draw parallels between non-literally similar tools. Equally, it enables us to search for a justification why a certain group of tools scores better in efficiency than another group. The latter will be further explored in Section 6.

Table 2 depicts TCT and effort required to model one feature, one feature group and one feature constraint. The groups are arranged in a sequential manner, with G1 being the most efficient and G4 being the least. Table 2 suggests that all tools differ in offering support to each of the three sub processes. Unfortunately, no tool scores well (G1) in all the three sub processes.

Satisfaction: None of the tools was given the maximum rating by each of the participants. Nevertheless, RequiLine scores the best on satisfaction. Subsequently, the satisfaction ratings of Feature Modeling Tool, CVM Tool, Feature Model DSL and Pure::Variants are close to each other. The worst results are obtained for XFeature, CaptainFeature, FeatureIDE and MOSKitt.

From participants’ comments given, it is clear that most of the tools failed in satisfying them. The reasons brought up mainly are: (a) the inability to adapt to the participants’ modeling skills, i.e. a lack to support user expertise; (b) the unfitness to adapt to participants’ preferences; (c) the non-availability of functionalities like copy/paste, shortcuts and redo/do; (d) requiring more steps than needed; (e) restrictions/lack of flexibility, e.g. features, like parent and child are displayed and locked into levels on top of each other; and (f) bugs.

Table 2. Efficiency

Task Completion Time				Effort		
	Feature	Feature Group	Feature Constraints	Feature	Feature Group	Feature Constraints
G1	T3, T9	T4	T9, T7, T6, T8, T4	T3	T3, T4	T9, T4, T8, T7
G2	T4, T7, T2	T8, T5	T3, T2, T5	T4, T9, T7, T2	T9, T1, T5	T6, T3, T2, T1
G3	T8, T1, T6	T9, T7, T3	T1	T1, T5, T6	T8, T7, T2, T6	T5
G4	T5	T2, T1, T6	---	T8	---	---

- Safety

The tools were investigated on their compliance with the checks for redundancy, anomalies, inconsistency, and invalid semantics. Table 3 shows the results.

- **Functional Usability Feature**

The availability of the selected FUFs was evaluated for each tool; the results are given in Table 4. Analyzing Table 4, we observe that all tools lack support for the FUF ‘user expertise’. Feedback and shortcuts are slightly supported. Although reuse information is available in four tools, it was not working properly.

Table 3. Safety

Tools	R ^a	A ^b	I ^c	S ^d
T7	3	3	3	3
T2	0	0	3	3
T4	0	0	3	3
T9	0	0	0	3
T3	0	0	0	3
T8	0	0	0	3
T5	0	0	0	1
T1	0	0	0	0
T6	0	0	0	0

a. Redundancy, b. Anomalies, c. Inconsistency, d. Invalid semantics.

Table 4. Functional Usability Features

Tools	F ^a	U ^b	V ^c	E ^d	S ^e	R ^f	H ^g
T2	1	3	3	0	1	2	3
T7	1	3	3	0	1	2	0
T9	0	3	3	0	1	0	3
T4	1	2	3	0	1	2	0
T8	1	3	3	0	1	0	0
T6	0	3	0	0	1	2	0
T1	0	0	3	0	0	0	3
T5	0	3	1	0	1	0	0
T3	1	0	3	0	0	0	0

a. Feedback, b. Undo, c. Field Validation, d. User Expertise, e. Shortcuts, f. Reuse information, g. Help.

6 Discussion and Lessons Learned

The objective of the evaluation was to analyze whether and to which extent the existing tools offer the necessary quality support to model variability, and subsequently, to learn from the imperfections of the existing tools. We have grouped our finding according to the three criteria considered: efficiency, safety, and FUFs.

Regarding the first criteria, we deduce from the results shown in Table 2 in the previous Section, that no tool meets the efficiency demands on all aspects. Table 2 unveils the strength and the weakness of each tool in each of the three sub processes.

Based on an analysis and a thorough comparison of these tools, following considerations were made in an effort to justify the differences in efficiency.

First of all, we consider the input of information. In this respect four main approaches are distinguished:

- By an external window: i.e. information is not added directly on the design surface; instead, an external window pops up in which the user writes the name of the feature and in which he selects its type – e.g. Pure::Variants and CaptainFeature use this technique for a feature, a feature group and a feature constraint.
- By using syntax: i.e. a kind of formal language, like for instance Extended Backus-Naur Form (EBNF) [18], is used – e.g., CaptainFeature and FeatureIDE use this technique to specify feature constraints.
- By using a toolbox: i.e. the user makes his selection from a list of buttons, e.g., RequiLine, Feature Modeling Tool, Feature Model DSL, and MOSKitt use this technique to create a feature as well as a feature group and a feature constraint.

- By using a context menu: e.g., by right clicking on a feature a menu pops up to edit it or to create a sub feature/feature group (CVM Tool uses this technique to create a feature group, and FeatureIDE uses it to create a feature and a feature group).

Specifying information by using an external window or/and by syntax are the most effort and time consuming. The use of syntax also requires good knowledge of the syntax to avoid errors.

The best results in efficiency were obtained when input can be done by a context menu. It owes its success to the fact that it is concentrated directly on the design surface, where the diagram is located. Although a toolbox is commonly used in modeling tools, its use turns out to be very counterproductive because of the trajectory of going forth and back from the toolbox to design surface.

Secondly, with relevance to efficiency, we encountered three different ways of editing. Editing was done by either (a) a properties menu, or (b) a context menu or (c) a single- and double-click of the mouse. Editing by mouse clicks appears to be least effort and time consuming, whereas editing by a properties menu should be avoided.

Thirdly, unnecessary steps should be omitted. The cumbersome way for modeling a feature group in 3 separated steps (like done in most tools) aptly illustrates this. Since the feature group logically consists of one parent and at least two children, modeling the feature group can be easily reduced to one single step.

And last but not least, the use of defaults has a negative effect on the efficiency, e.g., default feature type, default position (i.e. position of the feature on the design surface after its creation), and default feature size (i.e. the size of its symbol). A default carries the disadvantage that it may need to be adjusted afterwards, which requires twice as much effort and time. Therefore, defaults must be used with care.

With respect to safety, any kind of inconsistencies, anomalies and invalid semantics should be avoided. With respect to the redundancy check, the user should have the option to allow it since in some cases redundancy can add readability and understandability to the model.

Concerning the Functional Usability Features (FUFs), it is undisputable that these should be supported. The comments of the participant's post-questionnaire unveil a significant influence of the FUF on satisfaction.

To summarize, we can give the following advices:

- Any action – like input, editing, arranging features – taking place outside of the design surface should be avoided as much as possible. The use of a context menu meets these needs the best. Another modeling technique worthwhile to explore is one that visualizes features, feature group and feature constraints on the design surface (with small icons) and where the user can create these by a single mouse click (cf. CmapTool [34]). Although such a technique looks promising, it should be evaluated to verify if indeed it results in better efficiency.
- At the same time, input and editing should take as few steps as possible. Shortcuts and a kind of single- and double-click method can be advised.
- In general, defaults should only be used if in most cases the user does not need to change them. In case of default position, we advise to position the created features either under the parent or at the position of the mouse. As to default size, we propose to 'AutoFit' the size of the feature to its name.

- Moreover, the safety checks (except redundancy check) as well as the Functional Usability Features have to be provided.
- Furthermore, allowing the user to configure the tool according to his personal expertise and preferences is also considered important (e.g., define his own shortcuts and to (de)activate safety checks).

7 Related Work

To the best of our knowledge, no previous work has evaluated these specific quality requirements regarding tools supporting modeling variability.

In [6] the author presented a systematic functionality review of 19 tools, which support the domain analysis process. However, the review only included functional requirements. [16] provides an evaluation report of three tools on the importance of tool support for functionality, usability and performance within software product lines. The tools were completely reviewed based on the author's own opinion without a scientific experiment. An industry survey of 4 tools was conducted in [17], where usability had been categorized as a technical criterion. The literature review in [8] highlights the need for an inconsistency, redundancy and anomalies check.

8 Conclusion and Future Work

In order to investigate how to improve the quality of feature modeling tools in general, we conducted a quality evaluation of 9 feature modeling tools. These tools were measured against the evaluation criteria Usability, Safety and Functional Usability Features. From the results, we could extract justifications for the differences in quality support of the tools. Some lessons could be drawn to improve the quality support of such tools in the future. The next step in our research is to investigate quality requirements related to scalability. Many tools work fine for small models, but when the size of the models' increases, they suffer from severe scalability problems.

References

1. Pohl, K., Böckle, G., van der Linden, F.: *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, New York (2005)
2. Sinnema, M., Deelstra, S., Nijhuis, J., Bosch, J.: COVAMOF: A Framework for Modeling Variability in Software Product Families. In: Nord, R.L. (ed.) *SPLC 2004*. LNCS, vol. 3154, pp. 197–213. Springer, Heidelberg (2004)
3. Becker, M.: Towards a General Model of Variability in Product Families. In: *Proceedings of the 1st Workshop on Software Variability Management*, Netherlands (2003)
4. Abo Zaid, L., Kleinermann, F., De Troyer, O.: Feature Assembly Framework: Towards Scalable and Reusable Feature Models. In: *Fifth International Workshop VaMoS (2011)*
5. Chen, L., Babar, M.A.: A Systematic Review of Evaluation of Variability Management Approaches in Software Product Lines. *Information and Software Technology* 53, 344–362 (2011); Elsevier Journal
6. Lisboa, L.B., Garcia, V.C., Almeida, E.S., Meira, S.L., Lucrédio, D., Fortes, R.P.: A Systematic Review on Domain Analysis Tools. *Information and Software Technology* 52, 1–13 (2010)

7. Chen, L., Babar, M.A.: Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 166–180. Springer, Heidelberg (2010)
8. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Information Systems* 35, 615–636 (2010)
9. Bevan, N.: Extending Quality in Use to Provide a Framework for Usability Measurement. In: *Proceedings of HCI International*, San Diego, California, USA (2009)
10. Juristo, N.: Impact of usability on software requirements and design. In: De Lucia, A., Ferrucci, F. (eds.) *ISSSE 2006-2008*. LNCS, vol. 5413, pp. 55–77. Springer, Heidelberg (2009)
11. Baecker, R.M.: *Readings in Human-Computer Interaction: Toward the year 2000*. Morgan Kaufmann, San Francisco (1995)
12. Tamir, D., Komogortsev, O.V., Mueller, C.J.: An Effort and Time Based Measure of Usability. In: *The 6th International Workshop on Software Quality*. ACM, Leipzig (2008)
13. Hornbæk, K.: Current Practice in Measuring Usability: Challenges to Usability Studies and Research. *International Journal of Human-Computer Studies* 64, 79–102 (2006)
14. Massen, T.V.D., Lichter, H.: Deficiencies in Feature Models. In: *Workshop on Software Variability Management for Product Derivation - Towards Tool Support* (2004)
15. Lazar, J.: *Research methods in Human-Computer Interaction*. Wiley, Chichester (2010)
16. Chong, S.: *An Evaluation Report for Three Product-Line Tools (Form, Pure::Variants and Gear)*. NASA Software Assurance Research Program (2008)
17. Djebbi, O., Salinesi, C., Fanmuy, G.: Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues. In: *15th IEEE International Requirements Engineering Conference, RE 2007*, pp. 301–306 (2007)
18. Wirth, N.: *Extended Backus-Naur Form (EBNF)*. ISO/IEC 14977:1996 (2996)
19. CaptainFeature, <http://sourceforge.net/projects/captainfeature/>
20. Pure::Variants, http://www.pure-systems.com/pure_variants.49.0.html
21. FeatureIDE, http://www.witi.cs.uni-magdeburg.de/iti_db/research/featureide/
22. RequiLine, <http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/index.php>
23. XFeature, <http://www.pnp-software.com/XFeature/>
24. MOSKitt, <http://www.moskitt.org/eng/moskitt0/>
25. Feature Modeling Tool, <http://giro.infor.uva.es/index.html>
26. Feature Model DSL, <http://featuremodeldsl.codeplex.com/releases/view/20407>
27. CVM, <http://www.cvm-framework.org/index.html>
28. BigLever's Gears SPLE Too, <http://www.biglever.com/solution/product.html>
29. Feature Modeling Plug-in, <http://gsd.uwaterloo.ca/projects/fmp-plugin/>
30. FORM CASE Tool, <http://selab.postech.ac.kr/form/>
31. ToolDay - Tool for Domain Analysis, http://www.rise.com.br/english/products_toolday.php
32. ALLCapture, <http://www.balesio.com/allcapture/eng/index.php>
33. WinOMeter, <http://www.tjelinek.com/main.php?section=w>
34. CmapTools Knowledge Modeling Kit, <http://cmap.ihmc.us/>
35. Tukey, J.W.: *The Problem of Multiple Comparisons*. Princeton University, USA (1953)

Service Variability Patterns

Ateeq Khan¹, Christian Kästner², Veit Köppen¹, and Gunter Saake¹

¹ University of Magdeburg, Germany

² Philipps Universität Marburg, Germany

{ateeq, vkoeppen, saake}@iti.cs.uni-magdeburg.de,

kaestner@informatik.uni-marburg.de

Abstract. Service-oriented computing (SOC) increases flexibility of IT systems and helps enterprises to meet their changing needs. Different methods address changing requirements in service-oriented environment. Many solutions exist to address variability, however, each solution is tailored to a specific problem, e.g. at one specific layer in SOC. We survey variability mechanisms from literature and summarize solutions, consequences, and possible combinations in a pattern catalogue. Based on the pattern catalogue, we compare different variability patterns and their combinations. Our catalogue helps to choose an appropriate technique for the variability problem at hand and illustrates its consequences in SOC.

1 Introduction

Service-Oriented Computing (SOC) is a paradigm to create information systems and provides flexibility, interoperability, cost effectiveness, and higher quality characteristics [1]. The trend of service-usage is increasing in enterprise to support processes.

However, even in the flexible world of services, *variability* is paramount at all layers. Variability is the ability of a system to extend functionality, modify, customise or configure the system [2]. We do not want to provide the same service to all consumers but need to provide customised variants. Consumers want to fine tune services according to their needs and will get a unique behaviour, which is tailored (personalised) for their requirements. Fine-tuning depends on available *features* of the services, where a feature is a domain-abstraction used to describe commonalities and differences [3].

However, variability approaches in SOC are ad-hoc. Many solutions exist; however, each one is tailored and aimed for a specific problem or at a specific layer. Some approaches use simple mechanisms for variability, such as, using if-else structure implementations for variability in services. Others try to prevent bloated results of putting all variability into one service (which also violates the service principle that each service should be an atomic unit to perform a specific task) with various strategies, such as frameworks ([4,5,6]) and languages-based approaches ([7,8]). A single and perfect-for-all solution does not exist in variability. Such a solution is also unrealistic, due to very different requirements and technologies at different layers. Still, we believe that there are common patterns, and developers do not need to rule out inefficient solutions and reinvent better solutions again and again.

We contribute a catalogue of common variability pattern, designed to help developers to choose a technique for specific variability needs. We survey the literature and

abstract from reoccurring problems and individual implementation strategies and layers. We summarise our results in six common patterns for variability in the SOC domain (in general many patterns are even transferable to other domains). The patterns are general enough to describe the problem and the solution strategy including its trade-offs, different implementation strategies at different SOC layers, but also concrete enough to guide a specific implementation. To help developers decide for the appropriate solution to a variability problem at hand, we discuss trade-offs, limitations and possible combinations of different patterns. To aid understanding, we discuss example scenarios of each pattern with their consequences.

2 Variability Patterns in SOC

We use the pattern template by Gamma et al. [9] with modification to describe our patterns in SOC domain. Our pattern structure is simple and consists of a pattern name, pattern motivation or recurring problem, applications, examples, implementation technique or solution for the pattern, and consequences of the pattern.

In our pattern catalogue, we include some general patterns which are used in various variability situations. We discuss some implementation techniques and examples. Discussion of all implementation techniques for each pattern is out of scope of this paper (for details see [10]). In contrast to related pattern catalogues [2,11], which are focused on product lines, we focus on the SOC domain. We use examples from a sports SaaS application, which is used to manage a sports club. The SaaS application contains different services, e.g. to display the matches' results, managing players and members. Our sports application can be used in other sports domains by using variability approaches.

2.1 Parameter Pattern

Motivation: Service providers offer different implementations of a service and selection of services are based on the parameters. Service consumers have different kinds of preferences for a service or need a specific behaviour from services.

Application: This is a simple and widely used pattern. This pattern provides variability solutions based on parameters. Service providers offer variability depending on parameters (depicted in Figure 1) e.g. who is calling the service (consumer id). Access to specific services is decided using this pattern. Service providers plan for variability at design time and this result in variability for a consumer at runtime. Parameters and consumer specific configurations may be bundled together and stored. There are different options to store the configuration of the consumers, mostly stored at the service provider side (although, storage does not have an impact, e.g. at the consumer side or at the service provider side). When a consumer accesses the service, consumer specific configuration is accessed for variability and unique behaviour. We can use parameter pattern for user-interface or workflow preferences, database attributes or for domain specific extensions.

Example: We can use the parameter pattern for sorting, rendering, or for different layouts in a sports service scenario, e.g. offering text commentary of a match based on the consumer language or changing scoring fields for different sports domain.

Solution: We can store consumer specific settings and parameters as configuration files, e.g. as XML files or stored in a database for each consumer. Parameter storage is also not necessary; a possible extension is passing all required parameters every time when a consumer accesses the SaaS application. There are two types of data associated with this pattern, one is configuration specific data (values configured by consumers for different options) and other is application specific data for each consumer (contain database, values, and users). Configuration data is usually small and less updated as compared to application specific data. For general needs or requirements, configuration data for each consumer can be stored as key-value pair, e.g. consumer id and configuration values (for user-interface favourite colour, selected endpoint, or fields to display).

Consequences: This pattern provides an easy approach to provide variability from the same source code by storing and accessing consumer-specific behaviour based on parameters. Services are selected based on attribute values. Such approach is simple to program and does not require a lot of expertise. This pattern provides flexibility but consumer can choose only from the provided set. Management will be an issue in larger scenarios if parameter conditions are scattered within the code.

2.2 Routing Pattern

Motivation: Even if requirements are same between two consumers, business rules can vary between them. Consumers want to change the business rules to follow a specific behaviour. This pattern routes the request based on the rules or consumers requirements.

Application: We can use this pattern for routing requests to different targets, selection of services, changing application behaviour using rules or based on consumer description. Changes can be made at runtime. Flexibility is provided by consumer, provider or by both depending on the scenario and can be used at different layers. Service providers offer consumers to change the business rules of an application. Rules are used to handle complex scenarios and different conditions. Such conditions are due to user preferences. Meta rules or algorithms can be used to choose which rule has to be executed. Service providers can also allow to use specific operators, e.g. allowing consumers to add if-else branches in the business rules (shown in Figure 2) to control the business logic or

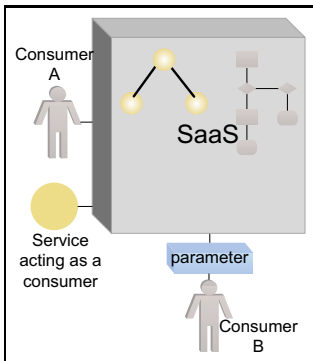


Fig. 1. Parameter pattern

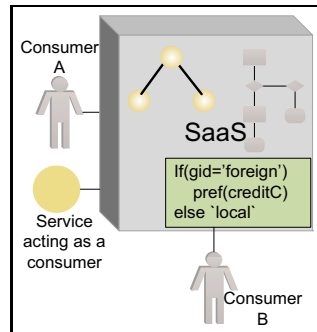


Fig. 2. Routing Pattern

using logical operators. Logical operators can also be source of variability, e.g. some consumers may use simple operators and others prefer or require more flexible rules for business logic. We can use this pattern to to handle exceptions. This pattern is similar to the façade or proxy pattern, discussed in [9].

Example: In our sports system, members pay club membership fees. For payments different options, or routing of services are possible, e.g. local members pay using credit card, bank transfer or both, and foreign members can only pay using credit card.

Solution: Different solutions for implementation do exist for routing. These approaches range from simple if-else statements to complex Aspect-Oriented Programming (AOP) based approaches [12]. Message interception can also be used for routing. A message is intercepted and analysed to add user-specific behaviour. Different techniques are used to intercept message. Rahman et al. [12] use an AOP-based approach to apply business rules on the intercepted message in SOA domain.

A web service request is intercepted, rules are applied on the request, and then result is forwarded. Routing can be done by analysing SOAP header or SOAP body (may carry extra data for routing) and request is routed accordingly.

Consequences: Routing allows consumer to use application which suits to their requirements. It also allows to separate business logic from service implementation (for easy modification in rules at runtime). It is also easy to change routing rules and only few changes are necessary. Consumers influence the application behaviour by changing rules.

Adding new business rules or logical operators may add unnecessary loop in an application or inconsistency in application. Validation rules or validators are applied before adding branching rule [13,14]. Higher complexity of involved services may lead to inconsistency in application due to rules. Algorithms for validation [15] can also be used to find inconsistent or contradictory rules. Scalability is also an issue for complex applications, routing rules may increase in size and their management become difficult. Routing pattern may introduce single point of failure or decrease in performance.

2.3 Service Wrapping Pattern

Motivation: We use this pattern when service is incompatible to use (due to technical or business issue) or provider want to add/hide functionality in services. So, modification is required to use the service in a scenario.

Application: We can use this pattern (as depicted in Figure 3) for the wide variety of changes, e.g. from technical perspective interface mismatch, message or data transformation, protocols transformation, or for business modifications (content modification). This pattern helps to delegate, modify or extend the functionality for consumers [16,11]. Service wrapping can be used to modify existing services and to resolve incompatibilities between services or service interfaces. Services are wrapped and arranged together so that a service delegates the request to other services or component, which implement the service logic. Composite, decorator, wrapper, proxy, and adaptor patterns [9] are similar patterns with the service wrapping pattern. We can also use this pattern to offer a group of services (from different providers, platforms, or languages) as a composite service to provide sophisticated functionality and vice versa. Consumers use services through the provided interface without knowing whether the service provider adds or

hides the functionality. We can also use this pattern to support legacy systems without major modification of existing code of the system and exposing functionality as a service [1, 17, 18]. The consumers may want to expose her existing systems as a service for other consumers, and restrict the access of some private business logic.

Example: An example from our sports system is offering email and SMS message services (wrapped together as a *notify match* composite service) to send reminder about change in match schedule to members and players.

Solution: We can use different solutions, e.g. using intermediate service, middleware solutions or tools for variability. To expose legacy systems as service, different techniques are possible, e.g. service annotations in Java. Intermediate service acts as an interface between incompatible services and contains required logic to overcome the mismatch.

Using SAP Process Integration (SAP PI) as a middleware, different service implementation, workflows, or client interfaces can be used to provide variability. We can use different types of adapters to solve interface mismatch or to connect different systems. When a request from a consumer side is sent to SAP PI, different service implementations, business rules, and interfaces can be selected based on the request. We also use middleware for synchronous-asynchronous communication, in which results are stored at middleware and delivered to the consumers based on their requests. The consumer or provider both (not necessary service owner, could be third party providers) are responsible for variability in this pattern.

Consequences: Using this pattern, we offer different variability solutions. Service wrapping hides the complexity of the scenario from the consumer and simplifies the communication between consumer and composite service (consumers do not care about different interfaces or number of underlying services). Addition or removal of a service becomes easy for the consumer (considered as include/exclude component in case of component engineering). Services are reused and become compatible without changing their implementation details by using service wrapping.

Composite services increase the complexity of the system. Adding services from other providers may effect non-functional properties. Service wrapping increases the number of services (depending on the scenarios composite, adapters or fine-grained) offered from the provider and management of such a system becomes complex.

2.4 Variant/Template Pattern

Motivation: We assume that providers know consumers variability requirements for services. Therefore, providers offer static variants of services, and consumers configure these variants according to their needs, e.g. variants based on the consumer geographical location, cultural aspects, subscription, consumer group, and devices.

Application: Providers offer a set of service variants to consumers (as illustrated in Figure 4). Service providers plan for the variability and provide variants at design time and consumers select these variants, mostly at runtime. Service providers select features and varying options based on industry best practices, as variants, with a pre-defined set of configuration options. Consumers choose options. In [13, 14], authors suggest to offer a set of templates, so consumers can choose a template in a process or workflow. In [19], authors discuss different workflow patterns.

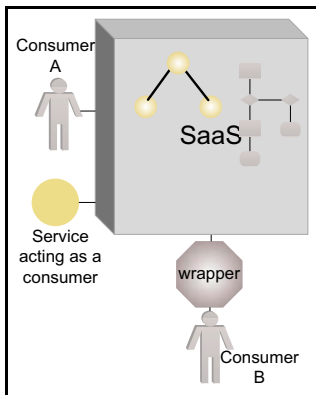


Fig. 3. Service Wrapping Pattern

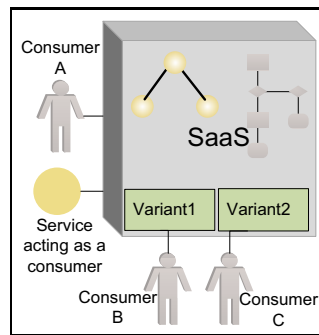


Fig. 4. Variant Pattern

Example: In our sports system, different user-interface variants are used to display match scores (suppose in Figure 4, text commentary is displayed in *Variant1* for the consumer B, online video streaming in *Variant2* from provider side for consumer C, while the consumer A sees the score only).

Solution: The variant pattern is a general pattern and used in various scenarios. Consumers choose from a set of variants and use options to configure it, e.g. for unique look and feel, workflows, or for viewing/hiding data fields in interface. These variants are typically generated from the same source code at provider side. We can use generators, inheritance, polymorphic, or product line approaches to generate variants of a service at design time [3, 20, 21, 9]. In [3], we discuss how different variants can be offered based on a feature set from the same code base and benefits achieved using variability. WSDL files can also be tailored and used for representing different variants. The consumer specific options are stored as configuration files.

Consequences: It pattern allows to offer optimal solutions in the form of variants. Industry best practices help consumers to choose right options and result in higher quality.

This pattern does not allow full flexibility to consumers. Developers provide variants in advance and consumers have to choose only from given set. Managing different variants of a service increases the complexity. Additional information is needed to decide which variant of a service is useful or compatible. Complex scenarios need a flexible platform or architecture, which allows handling of different variants (challenges mentioned in [3]).

2.5 Extension Points Pattern

Motivation: Sometimes, consumers have specific requirements which are not fulfilled by the above mentioned patterns. For instance, consumers want to upload their own implementation of a service, replace part of a process, to meet the specific requirements. Therefore, providers offer extension points in a SaaS application.

Application: This pattern requires pre-planning. Service providers prepare the variability as extension points at design time. Consumers share the same code base and provide behaviour at those extension points at runtime. Other consumers access the service

without any change. It is similar to the strategy design pattern [9], frameworks, or callbacks (can use inheritance methods at design time). The consumer modifies the application behaviour by uploading implementations, rules, or fine-tuning services (changing service endpoints). Extension points allow consumers to add consumer-specific implementations or business logic in the system at runtime as shown in Figure 5.

Example: In our sports system, a consumer configures extension point for alternative scoring services from different providers using web service endpoint binding method.

Solution: In SOC, service interfaces (WSDL files), service implementations, service bindings, and ports (endpoints) act as extension points in the architecture [22, 23, 24]. Consumers change these extensions points for variability. We can use physical separation of instances or virtualisation as solutions for this pattern. A provider allocates a dedicated hardware or a virtual instance for consumer-specific code execution separately. In case of malicious code or failure, only the tenant-specific instance or virtual image will be effected instead of the whole system. The consumer can perform modifications for service binding in WSDL. Endpoint modification is a method to modify the service address in a WSDL or in a composite service, e.g. adding an end-point service as an alternative in a web service binding. Endpoint modification can be done at runtime.

Consequences: Extension points offer flexibility to the consumer and allow customisation of application behaviour. There are some potential risks due to offering flexibility through extension points. In a workflow, by allowing a consumer to add activities, it is possible that adding new activities in a workflow introduce loops in application, consuming resources or might result in never ending loops. Another problem is in allowing a consumer to insert her own code, which may lead to failure of the whole system or instance, e.g. in case of malicious code or virus uploading. Once variability is realised by consumers, the system must check for the modification (extension points) and test scenarios for correctness of the system, e.g. for resource consumption or effect on the whole process (availability, time constraints for response, etc.)

2.6 Copy and Adapt Pattern

Motivation: Offering variability from the same code base in SaaS is not always a best choice. Sometimes, available patterns or approaches fail to fulfil consumers demands from the same code base. Another reason is, if we apply those patterns, management become complex or result in higher costs as compared to separate service instances.

Application: We use this pattern when shared instance modifications for a consumer harm other consumers. Therefore, a developer copies the service code and modifies it for individual consumer as depicted in Figure 6. This pattern requires source code access for modification. Mostly, the consumer is responsible for managing changes or updating the new version of service with own modifications. We also use this pattern where consumers have data privacy issues, e.g. in some countries, data storing, or processing in the shared environment is not feasible.

Example: We use this pattern in scoring service. Scoring is different for football (for consumer group A) as compared to baseball (for consumer group B) and a lot of changes are required, which makes the scenario complex.

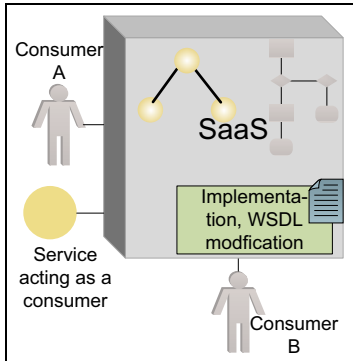


Fig. 5. Extension Points Pattern

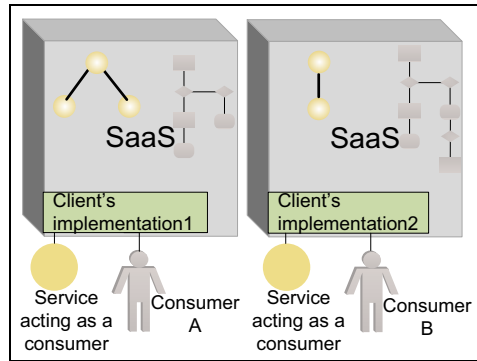


Fig. 6. Copy and Adapt Pattern

Solution: Service providers offer a separate instance for a consumer to keep the solution simpler, although it may introduce services with similar codes and functionalities. The consumer introduces her own implementation and exposes as a service or modifies the provided solution. In such a case, every consumer gets an independent customised service instance. We use this pattern at the process or database layer as well, where a consumer adds or develops her own process in SaaS. In such cases, at the database layer, a consumer uses a separate database instance to accommodate new database relations and different business requirements.

Consequences: SOC benefits are achieved in this pattern, although for some parts the application service provider (ASP [25]) model is used, in which each consumer shares the infrastructure facilities (shifting infrastructure and management tasks to providers) but separate service instances. Legacy systems or other applications can be shifted to SOC using this pattern easily. This pattern allows full flexibility, and consumers can modify or customise respective services freely.

From service provider perspective, this pattern does not scale. It is expensive in terms of costs for the large number of consumers and service instances. Hardware costs also increase in such cases due to separate instances. Code replication increases the effort for management and decreases productivity. Software updates or new version of software must be updated for each instance manually or individually. Due to these main problems, it is often not advisable to use this pattern and sometimes considered as anti-pattern.

3 Patterns Comparison and Combinations

We discuss different patterns for variability in SOC. In Table 1, we compare these patterns with each other against evaluation factors for variability. Our pattern catalogue covers the common variability problems and solutions in SOC and by no means a comprehensive pattern catalogue. We identify that some patterns can also be combined together for a better solution or to solve variability problems. For example, the *parameter pattern* can be combined with the *extension points pattern* to keep the consumer

Table 1. Pattern comparison and combinations

Patterns	Required changes	Flexibility	Scalability	Risk	Maintenance	Responsibility
Parameters (P)	low	medium	high	low	easy	provider
Routing (R)	low	medium	medium	medium	easy	both
Service Wrapping (SW)	medium	high	medium	medium	medium	both
Variants (V)	very low	low	medium	low	medium	provider
Extension Points (E)	medium	medium	low	high	difficult	provider
Copy and Adapt (CA)	very high	very high	high	low	difficult	both
Combining P + E	medium	medium	high	low	low	provider
Combining R + SW	medium	high	high	low	medium	consumer
Combining R + V	low	high	medium	low	medium	both
Combining R + E	medium	low	low	medium	difficult	both
Combining SW + V	medium	medium	high	low	medium	provider
Combining V + E	medium	high	medium	low	medium	provider

implementation separate from other consumers. Consumer's implementations are stored in configuration files and retrieved when consumers access the service.

We can also combine the *routing pattern* with the *variant pattern* or the *service wrapping pattern* to select different variants of services and protocols or messages transformation based on some criteria. The *routing pattern* is used with the *extension points pattern* to inject routing rules in application (e.g. uploading code containing routing logic). We can also use the *routing pattern* to offer a set of valid rules based on variants. The *service wrapping pattern* can be mixed with the *variant pattern* or the *routing pattern* to offer different variants of services. These variants are shared between consumers and used for different service flows or to overcome a mismatch at middleware level. The *variant pattern* with the *extension points pattern* allows us to restrict the extension points options to valid combinations instead of giving consumers flexibility to add random activities. So, consumers can add activities or rules from offered templates. An example of such an activity in our sports system is a *notification activity* where a consumer can send an email for a match notification but other consumers want to add additional SMS message activity for notification. So, SMS message activity can be added in the workflow from templates activity.

It is possible that different patterns fit in a particular environment or problem. Choosing a pattern depends on many factors, e.g. patterns consequences, application scenarios, business needs, architectures, and customers business models. In some organisation and countries, consumers have legal or organisational issues, restrictions for shared access of applications (despite the efforts for data and processes confidentiality in multi-tenant applications), so the consumer may prefer other patterns.

4 Summary and Outlook

We contributed six variability patterns for SOC that can guide developers to solve different variability problems in practice. We discuss trade-offs according to several

evaluation criteria to help deciding for the right solution strategy for a problem at hand. Our pattern catalogue helps to reuse solutions strategies in a manageable way.

In future work, we plan to extend our pattern catalogue into a framework that contains decision criteria to choose and manage variability in SOC with specific implementation techniques. We will also evaluate our pattern catalogue further in practice to compare performances where more than one patterns can be used at the same time.

Acknowledgement. Ateeq Khan is supported by a grant from the federal state of Saxony-Anhalt in Germany. This work is partially supported by the German Ministry of Education and Science (BMBF), within the ViERforES-II project No. 01IM10002B.

References

- [1] Papazoglou, M.P., van den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. *VLDB* 16(3), 389–415 (2007)
- [2] Svahnberg, M., van Gurp, J., Bosch, J.: A taxonomy of variability realization techniques. *Software - Practice and Experience* 35(8), 705–754 (2005)
- [3] Apel, S., Kästner, C., Lengauer, C.: Research challenges in the tension between features and services. In: *ICSE Workshop Proceedings SDSOA*, pp. 53–58. ACM, NY (2008)
- [4] Cámara, J., Canal, C., Cubo, J., Murillo, J.M.: An Aspect-Oriented Adaptation Framework for Dynamic Component Evolution. *Electr. Notes Theor. Comput. Sci.* 189, 21–34 (2007)
- [5] Guo, C.J., Sun, W., Huang, Y., Wang, Z.H., Gao, B.: A framework for native multi-tenancy application development and management. In: *The 9th IEEE International Conference on E-Commerce Technology*, pp. 551–558 (2007)
- [6] Kongdenfha, W., Saint-Paul, R., Benatallah, B., Casati, F.: An aspect-oriented framework for service adaptation. In: Dan, A., Lamersdorf, W. (eds.) *ICSOC 2006*. LNCS, vol. 4294, pp. 15–26. Springer, Heidelberg (2006)
- [7] Charfi, A., Mezini, M.: AO4BPEL: An aspect-oriented extension to BPEL. *WWW* 10(3), 309–344 (2007)
- [8] Zur Muehlen, M., Indulska, M.: Modeling languages for business processes and business rules: A representational analysis. *Information Systems* 35, 379–390 (2010)
- [9] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Reading (1995)
- [10] Khan, A., Kästner, C., Köppen, V., Saake, G.: Service variability patterns in SOC. Technical Report 05, School of Computer Science, University of Magdeburg, Magdeburg, Germany (May 2011), http://www.witi.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/KKKS11.pdf
- [11] Topaloglu, N.Y., Capilla, R.: Modeling the Variability of Web Services from a Pattern Point of View. In: Zhang, L.J. (ed.) *ECOWS 2004*. LNCS, vol. 3250, pp. 128–138. Springer, Heidelberg (2004)
- [12] ur Rahman, S.S., Khan, A., Saake, G.: Rulespect: Language-Independent Rule-Based AOP Model for Adaptable Context-Sensitive Web Services. In: *36th Conference on Current Trends in Theory and Practice of Computer Science (Student Research Forum)*, vol. II, pp. 87–99. Institute of Computer Science AS CR, Prague (2010)
- [13] Chong, F.T., Carraro, G.: Architecture strategies for catching the long tail, Microsoft Corporation (April 2006), <http://msdn.microsoft.com/en-us/library/aa479069.aspx> (last accessed June 24, 2011)

- [14] Carraro, G., Chong, F.T.: Software as a service (SaaS): An enterprise perspective, Microsoft Corporation (October 2006), <http://msdn.microsoft.com/en-us/library/aa905332.aspx> (last accessed June 24, 2011)
- [15] Bianculli, D., Ghezzi, C.: Towards a methodology for lifelong validation of service compositions. In: Proceedings of the 2nd International Workshop on Systems Development in SOA Environments, SDSOA, pp. 7–12. ACM, New York (2008)
- [16] Mügge, H., Rho, T., Speicher, D., Bihler, P., Cremers, A.B.: Programming for Context-based Adaptability: Lessons learned about OOP, SOA, and AOP. In: KiVS 2007 - Kommunikation in Verteilten Systemen, vol. 15. ITG/GI-Fachtagung (2007)
- [17] Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and managing web services: issues, solutions, and directions. The VLDB Journal 17(3), 537–572 (2006)
- [18] Mughrabi, H.: Applying SOA to an ecommerce system, Master thesis (2007), <http://www2.imm.dtu.dk/pubdb/p.php?5496> (last accessed May 5, 2011)
- [19] Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
- [20] Papazoglou, M.P., Kratz, B.: Web services technology in support of business transactions. Service Oriented Computing and Applications 1(1), 51–63 (2007)
- [21] Pohl, C., Rummler, A., et al.: Survey of existing implementation techniques with respect to their support for the requirements identified in m3. 2, AMPLE (Aspect-Oriented, Model-Driven, Product Line Engineering), Specific Targeted Research Project: IST- 33710 (July 2007)
- [22] Jiang, J., Ruokonen, A., Systa, T.: Pattern-based variability management in web service development. In: ECOWS 2005: Proceedings of the Third European Conference on Web Services, p. 83. IEEE Computer Society, Washington, DC, USA (2005)
- [23] Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for WS-BPEL. In: WWW, pp. 815–824. ACM, New York (2008)
- [24] Erradi, A., Maheshwari, P., Tosic, V.: Policy-Driven Middleware for Self-adaptation of Web Services Compositions. In: van Steen, M., Henning, M. (eds.) Middleware 2006. LNCS, vol. 4290, pp. 62–80. Springer, Heidelberg (2006)
- [25] Lacity, M.C., Hirschheim, R.A.: Information Systems Outsourcing; Myths, Metaphors, and Realities. John Wiley & Sons, Inc., Chichester (1993)

An Overview of Techniques for Detecting Software Variability Concepts in Source Code

Angela Lozano*

Université catholique de Louvain (UCL), ICTEAM,
Place Sainte Barbe 2, B-1348 Louvain La Neuve, Belgium

Abstract. There are two good reasons for wanting to detect variability concepts in source code: migrating to a product-line development for an existing product, and restructuring a product-line architecture degraded by evolution. Although detecting variability in source code is a common step for the successful adoption of variability-oriented development, there exists no compilation nor comparison of approaches available to attain this task. This paper presents a survey of approaches to detect variability concepts in source code. The survey is organized around variability concepts. For each variability concept there is a list of proposed approaches, and a comparison of these approaches by the investment required (required input), the return obtained (quality of their output), and the technique used. We conclude with a discussion of open issues in the area (variability concepts whose detection has been disregarded, and cost-benefit relation of the approaches).

1 Introduction

Today's companies face the challenge of creating customized and yet affordable products. Therefore, a pervasive goal in industry is maximizing the reuse of common features across products without compromising the tailored nature expected from the products. One way of achieving this goal is to delay customization decisions to a late stage in the production process, which can be attained through software. For instance, a whole range of products can be achieved through a scale production of the same hardware, and a software customization of each type of product.

The capability of building tailored products by customization is called variability. Software variability can be achieved through combination and configuration of generic features.

Typical examples of variability can be found in embedded software and software families. Embedded software facilitates the customization of single purpose machines (i.e. those that are not computers) such as cars, mobile phones, airplanes, medical equipment, televisions, etc. by reusing their hardware while

* Angela Lozano is funded as a post-doc researcher on a an FNRS-FRFC project. This work is supported by the ICT Impulse Program of ISRIB and by the Inter-university Attraction Poles (IAP) Program of BELSPO.

varying their software to obtain different products. Software families are groups of applications that come from the configuration and combination of generic features. Both embedded software and software families refer to groups of applications related by common functionality. However, variability can also be found in single applications when they delay design decisions to late stages in order to react to different environments e.g. mobile applications, games, fault tolerant systems, etc.

Motivations for Detecting Variability: When companies realize that slight modifications of a product could enlarge their range of clients they could migrate to a product-line development. Detecting variability opportunities in the current product is the first step to assess which changes have a higher return in terms of potential clients. This return assessment is crucial for the success of such migration because the reduction on development cost may not cover the increase in maintenance cost if the variation introduced is unnecessary.

Once a product-line architecture is in place, it will degrade over time. Degradation of the architecture is a reality of software development [7]. In particular, product-lines have well-known evolution issues that can degrade their architecture [3][12]. Therefore, at some point it might become necessary to restructure the product-line; and the first step is reconstructing its architecture from the source code.

Both previous scenarios start from the source code of the (software) system as source of information. Source code mining is a reverse engineering technique that extracts high-level concepts from the source code of a system. In order to mine for variability concepts, we need to clearly define these high-level concepts.

Although mining for variability in source code is a common step towards the successful adoption of variability-oriented development, there exists no compilation nor comparison of existing approaches available to attain this task. The goal of our paper is two-fold. First, we identify high-level concepts targeted by variability mining approaches and describe the intuition behind them. Second, we use these concepts to classify the approaches that detect variability as they uncover the intuition and assumptions behind the technique used.

2 Variability Concepts

In order to compare variability mining approaches we establish a common ground of terms found in literature. The purpose of this common ground is not to provide a formal definition of each term, but to fix its meaning in an intuitive manner. In particular, we consider four variability concepts central to variability mining: features, variability dependencies, variation points, and variants.

Features: Variability is described in terms of *features*. There are two types of features: mandatory and variable. To illustrate these kinds of features we present an example of the possible configurations of features in a car. Figure 1 depicts the car feature diagram, illustrating the relations between variability concepts.

A *feature* represents a unit of software functionality i.e. an increment of behavior (w.r.t. some standard car behavior). The features in Figure 1 are the

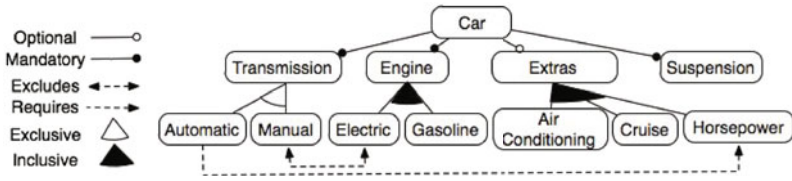


Fig. 1. A feature diagram

suspension, transmission, engine, and extra options. *Mandatory features* can be implemented directly in the application because they do not depend on the usage environment i.e. they do not require customizations. The mandatory features of Figure 1 are the suspension, transmission and engine. Mandatory features represent functionality that must be included in all products.

Variants: As mentioned in the introduction, the goal of variability is to allow for flexibility by delaying customization decisions. *Variable features* are those features that require customizations. The transmission, engine, and extras are the variable features of Figure 1, because they can vary from one car to another. The options available for a variable feature are called its *variants*. The transmission is a variable feature with two variants: automatic and manual (Fig. 1).

Depending on its variants a feature can be classified into optional or alternative [1]. *Optional features* do not require a variant and are included only in certain products, e.g. the extra options in a car (see example on Fig. 1). *Alternative features* implement an obligatory feature in the product family in different ways depending on the product developed. Alternative features are composed of mutually inclusive or mutually exclusive variants [1]. *Mutually inclusive* variants depend on each other, so including one of them requires to include the rest of them. Having mutually inclusive variants turns an alternative feature into a *multiple feature* because it can have several values assigned. *Mutually exclusive* variants are interchangeable. Having mutually exclusive variants make an alternative feature a *single feature* because it can have only one value assigned. In the example of Fig. 1, the transmission is a single feature because the manual and the alternative transmissions are mutually exclusive, while the engine is a multiple feature because the gasoline and electric engines are mutually inclusive.

Variable features can be *single* if they must have one value assigned, *multiple* if they can have several values assigned, and *optional* if they do not require a value assigned. The example shown in Figure 1 presents a single feature (the transmission), a multiple feature (the engine), and an optional feature the extra.

Variability Dependencies: There are two types of *variability dependencies* which dictate constraints among variable features: when including a feature *requires* including another feature, and when including a feature *excludes* or prohibits the inclusion of another feature. For instance, automatic transmission requires extra horsepower while the manual transmission excludes use of an electric engine (Fig. 1).

Variation Points: Variable features are implemented in the application in the form of *variation points*. A *variation point* is the placeholder for a value that has associated variable functionality. Variation points make explicit delayed design decisions. Associating a value to a variation point is called *binding*. Variation points with a value are *bound*, while those without a value are *unbound*.

3 Mining for Feature Diagrams

Feature diagrams describe the commonalities and differences of some domain model. Therefore feature diagram mining requires to analyze multiple applications of the same domain.

Antkiewicz et al. [2] analyzed applications that extend the same framework. These applications can be viewed as instances of the same domain because their usage of the framework shows configurations of the concepts provided by the framework. The authors aim at detecting variable features. The feature to be detected is described by an abstract concept that is the starting point of the analysis. For instance, Applet will look for all instances of Applet in the applications analyzed. A feature is extracted in the form of sets of distinguishing structural and behavioral patterns that allow discriminating configurations of the same concept. The patterns are detected with source code queries. The results of the queries are facts that describe the structural and run-time relations among source code entities. That is, variables/parameters/returned objects and types, methods implemented and understood, calls sent and received, precedence relations between methods called, etc. The mining builds a tree of source code facts obtained from the source code entity that is the starting point of the analysis. The algorithm then calculates commonalities and differences among the facts to establish mandatory, optional, multiple and indispensable patterns when configuring a feature of the framework.

Yang et al. [24] analyzed open source applications with similar functionality. The assumption is that data models (entity-relationship) of these applications are similar and uncover a basis for mapping the common concepts among different applications. The starting point of the mining is a reference domain data model. The data schema of the applications is then mapped to the reference domain model by detecting entities or fields with similar names in the applications. The approach is based on detecting consistent data access semantics (i.e., similar usage of data entities by the methods in the applications). The data access semantics are obtained by detecting SQL statements using aspects which intercept invocations to the database library and store SQL run-time records. In order to verify that different methods have a similar data access semantics, the SQL records must contain the tables, fields and constraints involved in the queries. These records describe the data access semantics of each method in each application. The records are then analyzed using Formal Concept Analysis (FCA), a classification technique that aims at finding the maximal set of objects that share a maximal set of properties or attributes. The resulting concepts represent the usage of data-entities mapped to the reference domain model. Depending

on the level of the concept, it is merged with neighbor concepts or pruned so that each concept represents one feature. The result is a domain feature diagram comprising mandatory features, variable features, whether they are alternative or multiple, its variants, and whether they are inclusive or exclusive. However, the approach is incapable of detecting variability dependencies.

4 Mining for Variability Dependencies

Variability dependencies are constraints between features that establish the valid products of a feature diagram. In terms of source code, variability dependencies (requires/excludes) are related with control flow relations between features. Nevertheless, other types of feature overlaps have been found by analyzing implementations of features regardless of being variable or mandatory.

Czarnecki et al. [5] define Probabilistic Feature Models for feature diagrams, organized using the probability of having a feature given the presence of another feature. The approach is based on counting the legal configurations of a feature diagram. The legal configurations are used to determine a set of legal samples (of legal configurations), which are in turn used to calculate the frequency of co-existence of all possible combinations of features. These frequencies are then used to obtain conditional probabilities, that is, the probability of requiring a feature given the presence in the application of another feature. Conditional probabilities are used to structure the feature diagram and to decide when a variable feature is optional, alternative, mandatory, inclusive or exclusive. Finally, a Bayesian Network with a Directed Acyclic Graph of the features, and the conditional probability table is used to learn variability dependencies. The Bayesian Network is capable of detecting require relations ($probability(feature1|feature2)$) and exclude relations ($probability(\overline{feature1}|feature2)$) among features. A disadvantage of this approach is that it does not analyze source code but it requires an intermediate approach to detect the Directed Acyclic Graph of features in the domain and their mapping to several products.

Parra et al. [19] propose an analysis of feature constraints based on the assumption that variable features are implemented with aspects. The approach detects that one feature requires a second feature when the pointcut that defines the variation point for the first feature references source code elements referred to by the aspect that defines the second feature. The approach can detect when one feature excludes a second feature, when the pointcuts (that define the variation points) for both features refer to the same source code elements. This analysis is used to detect the correct order (if it exists) to compose the variants represented by aspects in the source code. The disadvantage of this approach is that presupposes an aspect-oriented implementation of variability.

Egyed [6] assumes that calling a similar set of source code entities when executing different features implies that one feature is a sub-feature of the other. Similarly, Antkiewicz et al. [2] consider that a sub-feature is essential if it is common to all applications (of the same domain) analyzed. This approach locates a feature by identifying similar patterns in entities that may be related to the

feature analyzed, a sub set of these patterns is considered a sub-feature. Aside from detecting essential sub-features, their approach is also capable of detecting incorrect features due to missing essential sub-features. However, this approach is incapable of detecting variability dependencies across features that do not have a containment relation.

Lai and Murphy [16] described two situations in which several features are located in the same source code entity: overlap and order. Overlap occurs when there is no control or data flow dependency between the features in the source code entity. Overlap is expected and encouraged in sub-features; however, other types of overlap should be documented and handled during maintenance. Order occurs when the features in the source code entity have a control or data flow dependency, i.e. they require a partial execution order. Usually wherever many features were related, there was no partial order.

In his PhD thesis [11], Jaring proposes four types of variability dependencies depending on the binding of variation points. The categories are: dependencies between variation points, dependencies between a variation point and a variant, dependencies between a variant and a variation point, and dependencies between variants. This characterization is used to uncover hidden variability dependencies in a legacy application (that uses C macros to implement its variability). However it is not clear to what extent the analysis is automated and to what extent it requires user-input.

5 Mining for Variation Points and Variants

Variation points capture the functionality areas in which products of the same product family differ. Mining for variation points aims at detecting source code entities that allow diverging functionality across different products of the same domain. Although there exists literature describing how to implement variability [13,14,22,17], we could only find one approach to detect variation points and variants. The lack of approaches may be due to the wide variety of possibilities to translate a conceptual variation point (i.e. a delayed decision) to the implementation of a variation point, as well as to the difficulty to trace this translation [20,14].

Thummalapenta and Xie [23] analyze applications that extend the same framework. They calculate metrics that describe the amount and type of extensions per class and method of the framework. These metrics allow to classify the methods and classes of the framework into variation points (hotspots/hooks) and coldspots/templates. Authors, analyzed the level of variability of the frameworks analyzed by counting the percentage of classes and methods identified as variation points. The disadvantage of this approach is that it requires a high variety of applications from the domain to give reliable results.

A variant is an option available for a variable feature. Mining for variants implies assigning a high level concept to the values (or objects) used in the *variation points* to decide when to change the implementation of a variable feature. This means that mining for variants requires detecting variation points

and linking them to their corresponding variable feature, so it is possible to trace them to a single feature. We could not find any approaches to mine for variants. However this lack of results is predictable because of the lack of approaches to mine for variation points. Given that the approach described above [23] was designed to analyze the flexibility of frameworks, it lacks references to variability, and therefore, to variants.

6 Mining for Products of the Same Domain

Snelting [21] analyzes macros (C's preprocessor directives like `#ifndef`). The lines of code corresponding to each macro are characterized with the global variables (or configuration variables) that affects them. This characterization produces a table where each line has the lines of code of each macro and each column the variables configured in such macro. The table is analyzed using Formal Concept Analysis (FCA). The concepts of the lattice resulting from the analysis represent a possible configuration of the application. The intent of the concepts provides the variables that affect that configuration, and the extent the lines of code affected by that configuration. Moreover, the relations between concepts indicate subsumed configurations. Crossed relations between chains of concepts would indicate interference among those configurations. If the application is implemented separating concerns and anticipating changes the lattice would have disjoint sublattices in the middle of the lattice (i.e. high configuration coupling). Otherwise configuration variables of different features would be merged in single concepts in the middle area of the lattice, indicating dependencies and low coupling.

Hummel et al. [10] analyzed the signatures of the methods of several open source applications to recommend methods that classes in the same domain implement but that are missing from the user's implementation. The approach assumes that the domain concept that a class represents is summarized as the set of signatures of its methods. Using this information, their technique finds all classes with a similar set of signatures to the user's class, and detects the signatures missing from the user implementation. The disadvantage of this technique is that it is limited to the domain concepts that are implemented in open source applications, and therefore it might be difficult to use for business-specific domain concepts.

7 Mining for Variable vs. Mandatory Features

Several approaches mine for features. However, this section focuses on approaches to differentiate between mandatory (common functionality across products of the same domain) and variable features (divergent functionality across products of the same domain). This is usually achieved by analyzing the results of clone detection.

Faust and Verhoef [8] were the first ones to propose the analysis of diverse products of the same domain to explore development paths for mandatory features. They use bonsai maintenance as a metaphor to keep product lines under

control. The approach is called grow and prune because it aims at letting products of the domain evolve (grow), and then merge as mandatory features (prune) the successful source code entities of these products. For that reason, they proposed metrics to evaluate the success of a source code entity. A source code entity is defined as successful if its implementation contained a large amount of code, a low number of decisions, and a low frequency of changes, was highly used and cloned. Nevertheless the approach is incapable of offering further assistance for restructuring the product line.

Mende et al. [18] use clone detection to measure the level of commonalities across directories, files and functions of different products of the same domain. The idea is to assess to what extent a product (p1) can be merged into a basic product (p2), and to what extent the functionality of the product to be merged (p1) is included in the other product (p2). Depending on the number of identical functions (similarity = 1) and of similar functions (similarity between 0 and 1) of the product p2 into the product p1, it is possible to say if the correspondence is identical or potential, and if such correspondence is located in one function or not. Therefore the analysis proposed helps to detect potential mandatory features parts in an existing product line (i.e., product-specific functions identical across several products), as well as variability points that may require some restructuring to separate better mandatory and variable features (i.e., product-specific functions similar across several products).

Frenzel et al. [9] extract the model of a product-line architecture based on several products of the same domain. To extract the model they use reflexion models, which gives them the static components, their interfaces, their dependencies, and their grouping as layers and sub-layers in the system. The model is then compared with the implementation of the products by checking whether different products have corresponding components (based on clone similarity). Clone detection is also used to transfer common implementation areas to the common design. The latter two approaches are merged and further developed in [15]. However, given that these approaches do not mine for features, they are unable infer the feature diagram and its correspondence with the inferred architecture .

8 Conclusions

This paper compiles techniques to propose refactorings from variable to mandatory features, and to improve the comprehension of variable products by discovering the decomposition of features in a domain, the hidden links among implementations of variable features, the source code entities in charge of the variation, and valid products of a domain. Although we found several approaches to mine for most of the variability concepts the approaches can be improved in several ways. Some of the techniques have demanding requirements. For instance, reconstructing feature diagrams require an initial domain model [24] or entity that implements the feature to analyze [2], while detecting *excludes/requires* dependencies may require an aspect-oriented implementation of variable features [19] or an initial feature diagram and its mapping to several applications

5]. Other techniques have a restricted automated support. For example, reconstructing feature diagrams may require manual post processing [24]. Finally, the usefulness of the output of some techniques is limited to address architectural degradation of product-line design from single products. For instance, knowing the configurations of a product line and the variables involved [21] does not provide any hints on how to restructure it or on how to map this implementation details to domain concepts; which limits the usage of the approach to deal with architectural degradation due to evolution.

Mining for variation points is the area with highest potential because there are several papers that describe how variability should be implemented. Mining for variants also has potential given that is a neglected area, and that it is complimentary to the detection of variation points. Nevertheless, detecting variation points is not enough to detect variants because each variation point needs to be linked to a variable feature. However, the assignment of a variation points to a variable feature could be technically challenging because the majority of order interactions are due to control flow dependencies [16]. Another area open for future work is extending the approaches to mine for feature diagrams, and for variable and mandatory features to analyze the flexibility of single products in order to support the migration towards product-line development.

References

1. Anastasopoulos, M., Gacek, C.: Implementing product line variabilities. In: SSR 2001: Proc. of the 2001 Symposium on Software Reusability, pp. 109–117. ACM, New York (2001)
2. Antkiewicz, M., Bartolomei, T.T., Czarnecki, K.: Fast extraction of high-quality framework-specific models from application code. *Autom. Softw. Eng.* 16(1), 101–144 (2009)
3. Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J.H., Pohl, K.: Variability issues in software product lines. In: Revised Papers from the 4th Int'l Workshop on Software Product-Family Engineering, PFE 2001, pp. 13–21. Springer, Heidelberg (2002)
4. Brown, T.J., Spence, I., Kilpatrick, P., Crookes, D.: Adaptable components for software product line engineering. In: Chastek, G.J. (ed.) SPLC 2002. LNCS, vol. 2379, pp. 154–175. Springer, Heidelberg (2002)
5. Czarnecki, K., She, S., Wasowski, A.: Sample spaces and feature models: There and back again. In: SPLC 2008: Proc. of the 2008 12th Int'l Software Product Line Conference, pp. 22–31. IEEE Computer Society, Washington, DC, USA (2008)
6. Egyed, A.: A scenario-driven approach to traceability. In: ICSE 2001: Proc. of the 23rd Int'l Conference on Software Engineering, pp. 123–132. IEEE Computer Society, Washington, DC, USA (2001)
7. Eick, S.G., Graves, T.L., Karr, A.F., Marron, J.S., Mockus, A.: Does code decay? assessing the evidence from change management data. *IEEE Trans. Softw. Eng.* 27, 1–12 (2001)
8. Faust, D., Verhoef, C.: Software product line migration and deployment. *Software: Practice and Experience* 33(10), 933–955 (2003)

9. Frenzel, P., Koschke, R., Breu, A.P.J., Angstmann, K.: Extending the reflexion method for consolidating software variants into product lines. In: WCRE 2007: Proc. of the 14th Working Conference on Reverse Engineering, pp. 160–169. IEEE Computer Society, Washington, DC, USA (2007)
10. Hummel, O., Janjic, W., Atkinson, C.: Proposing software design recommendations based on component interface intersecting. In: Proc. of the 2nd Int'l Workshop on Recommendation Systems for Software Engineering, RSSE 2010, pp. 64–68. ACM, New York (2010)
11. Jaring, M.: Variability Engineering as an Integral Part of the Software Product Family Development Process. PhD thesis, Rijksuniversiteit Groningen (2005)
12. Johansson, E., Höst, M.: Tracking degradation in software product lines through measurement of design rule violations. In: Proc. of the 14th Int'l Conference on Software Engineering and Knowledge Engineering, SEKE 2002, pp. 249–254. ACM, New York (2002)
13. Keepence, B., Mannion, M.: Using patterns to model variability in product families. *IEEE Softw.* 16, 102–108 (1999)
14. Kim, S.D., Her, J.S., Chang, S.H.: A theoretical foundation of variability in component-based development. *Inf. Softw. Technol.* 47, 663–673 (2005)
15. Koschke, R., Frenzel, P., Breu, A.P., Angstmann, K.: Extending the reflexion method for consolidating software variants into product lines. *Software Quality Control* 17, 331–366 (2009)
16. Lai, A., Murphy, G.C.: The structure of features in Java code: An exploratory investigation. In: Ossher, H., Tarr, P., Murphy, G. (eds.) *Workshop on Multi-Dimensional Separation of Concerns (OOPSLA 1999)* (November 1999)
17. Maccari, A., Heie, A.: Managing infinite variability in mobile terminal software: Research articles. *Softw. Pract. Exper.* 35(6), 513–537 (2005)
18. Mende, T., Beckwermert, F., Koschke, R., Meier, G.: Supporting the grow-and-prune model in software product lines evolution using clone detection. In: Proc. of the 2008 12th European Conference on Software Maintenance and Reengineering, CSMR 2004, pp. 163–172. IEEE Computer Society, Washington, DC, USA (2008)
19. Parra, C., Cleve, A., Blanc, X., Duchien, L.: Feature-based composition of software architectures. In: Babar, M.A., Gorton, I. (eds.) *ECSA 2010. LNCS*, vol. 6285, pp. 230–245. Springer, Heidelberg (2010)
20. Salicki, S., Farcet, N.: Expression and usage of the variability in the software product lines. In: *Revised Papers from the 4th Int'l Workshop on Software Product-Family Engineering, PFE 2001*, pp. 304–318. Springer, London (2002)
21. Snelting, G.: Reengineering of configurations based on mathematical concept analysis. *ACM Trans. Softw. Eng. Methodol.* 5(2), 146–189 (1996)
22. Svahnberg, M., van Gurp, J., Bosch, J.: A taxonomy of variability realization techniques: Research articles. *Softw. Pract. Exper.* 35, 705–754 (2005)
23. Thummalapenta, S., Xie, T.: Spotweb: detecting framework hotspots via mining open source repositories on the web. In: Proc. of the 2008 Int'l Working Conference on Mining Software Repositories, MSR 2008, pp. 109–112. ACM, New York (2008)
24. Yang, Y., Peng, X., Zhao, W.: Domain feature model recovery from multiple applications using data access semantics and formal concept analysis. In: WCRE 2009: Proc. of the 2009 16th Working Conference on Reverse Engineering, pp. 215–224. IEEE Computer Society, Washington, DC, USA (2009)

Variability in Multi-tenant Environments: Architectural Design Patterns from Industry

Jaap Kabbedijk and Slinger Jansen

Utrecht University

Department of Information and Computing Sciences

Princetonplein 5, 3584CC, Utrecht, Netherlands

{j.kabbedijk, s.jansen}@cs.uu.nl

Abstract. In order to serve a lot of different customers in a SaaS environment, software vendors have to comply to a range of different varying requirements in their software product. Because of these varying requirements and the large number of customers, a variable multi-tenant solution is needed to achieve this goal. This paper gives a pragmatic approach to the concepts of multi-tenancy and variability in SaaS environments and proposes three architectural patterns that support variability in multi-tenant SaaS environments. The *Customizable Data Views* pattern, the *Module Dependent Menu* pattern and the *Pre/Post Update Hooks* pattern are explained and shown as good practices for applying variability in a multi-tenant SaaS environment. All patterns are based on case studies performed at two large software vendors in the Netherlands who are offering an ERP software product as a service.

Keywords: Software-as-a-Service, Variability, Multi-tenancy, Architectural Patterns.

1 Introduction

Increasingly, product software vendors want to offer their product as a service to their customers [1]. This principle is referred to in literature as Software as a Service (SaaS) [2]. Turning software into a service from a vendor's point of view means separating the possession and ownership of software from its use. Software is still maintained and deployed by the vendor, but used by the customer. The problem of moving a software product from different on-premises locations to one central location, is the fact that it becomes really difficult to comply to specific customer wishes. In order to serve different customers' wishes, variability in a software product is needed to offer specific functionality. By making use of variability in a software product, it is possible to supply software functionality as optional modules, that can be added to the product at runtime. Applying this principle can overcome many current limitations concerning software use, deployment, maintenance and evolution in a SaaS context [3]. It also reduces support costs, as only a single instance of the software has to be maintained [4].

Besides complying to specific customer requirements, a software vendor should be able to offer a service to a large number of customers, each with their own

requirement wishes, without running into scalability and configuration problems [5]. The solution to this problem is the use of multi-tenancy within a SaaS product. Multi-tenancy can be seen as an architectural design pattern in which a single instance of a software product is run on the software vendors infrastructure, and multiple tenants access the same instance [6]. It is one of the key competencies to achieve higher profit margins by leveraging the economy of scale [7]. In contrast to a model incorporating multiple users, multi-tenancy requires customizing the single instance according to the varying requirements among many customers [8]. Currently, no well documented techniques are available on how to realize the variability needed in multi-tenant SaaS environments.

First the research method is discussed in section 2, after which the most important concepts in this paper will be explained and discussed in section 3. Then, the trade-off between the level of variability needed and the number of customers is discussed in section 4, followed by three architectural design patterns for variability in SaaS product in section 5. The paper ends with a conclusion and future research in section 6.

2 Research Method

In this research, the variability realization techniques (VRTs) currently described in literature and in place in large SaaS providers are observed. In order to do this, a thorough literature study has been performed, in which the combinations of *variability, saas* and *variability, multi-tenancy* were used as keywords in Google Scholar¹. The VRTs discussed in the papers having the previous mentioned keywords in their title or abstract were collected and patterns in those patterns were put in a pattern database. A total of 27 papers was collected this way. Besides the literature study, two independent case studies were performed at large ERP providers who recently launched their enterprise resource planning (ERP) software as a service through the Internet (referred to as *ErpCompA* and *ErpCompB* from here on). *ErpCompA* has a turnover of around 250 million euros and around 20,000 users using their online product, while *ErpCompB* has a turnover of around 50 million euros and around 10,000 users. The case studies were performed using the case study research approach by Yin [9].

The VRTs are presented as architectural design patterns and created based on the Design Science principles of Hevner [10], in which a constant design cycle consisting of the construction and evaluation of the VRTs takes place. The initial model is constructed using a exploratory focus group (EFG) [11], consisting out of participants from academia and the case companies, and a systematic literature review [12]. The focus group has been carefully selected and all participants have experience in the area of variable multi-tenant SaaS-environments. Additional validation of the VRTs was done conducting interviews with software architects within the two case companies [13].

¹ Google Scholar (www.scholar.google.com indexes and searches almost all academic publishers and repositories world-wide.

3 Related Work and Definitions

To explain the multi-faceted concepts used in this paper, this section will discuss *multi-tenancy*, *design patterns* and *variability* in more depth. The definitions proposed are meant to enable researchers to have one shared lexicon on the topic of multi-tenancy and variability.

Multi-tenancy

Multi-tenancy can be defined as the ability to let different tenants “share the same hardware resources, by offering them one shared application and database instance, while allowing them to configure the application to fit their needs as if it runs on a dedicated environment” [5]. A tenant refers to an organization or part of an organization with their own specific requirements, renting the software product. We define different levels of multi-tenancy:

- **Data Model Multi-tenancy:** All tenants share the same database. All data is typically provided with a tenant specific GUID in order to keep all data separate. Even better is native support for multi-tenancy in the database management system [14].
- **Application Multi-tenancy:** Besides sharing the same database, all tenants also share the same instance of the software product. In practice, this could also mean a couple of duplications of the same instance, coupled together with a tenant load balancer [8].
- **Full Multi-tenancy:** All tenants share the same database and software instances. They can also have their own variant of the product, based on their tenant requirements. This level of multi-tenancy adds variability to the software product.

All items above are sorted on ascending implementation complexity.

Variability

The concept of variability comes from the car industry, in which different combinations of for example chassis, engine and color were defined as different *variants*. In software the concept is first introduced in the area of software product lines [15], in which variability is defined as “the ability of a software system or artefact to be efficiently extended, changed, customized or configured for use in a particular context” [16]. Within the area of software product lines, software is developed by the software vendor and then shipped to the customer to be run on-premises. This means variants have to be compiled before product shipping. Within the area of Software-as-a-Service, software is still developed by the software vendor, but the product is served to all customers through the internet from one central place [3,8]. In principle, all variants can be composed the moment customers ask for some specific functionality, so at run-time.

We identify two different types of variability within multi-tenant SaaS deployments:

- **Segment Variability:** Product variability based on the segment a tenant is part of. Examples of such variability issues are different standard currencies

or tax rules per country or a different layout for SMEs and sole proprietorships.

- **Tenant-oriented Variability:** Product variability based on the specific requirements of a tenant. Examples of such variability issues are different background colors or specific functionality.

We also identify different levels of variability in tenant oriented variability:

- **Low: Look and Feel:** Changes only influencing the visual representation of the product. These changes only occur in the presentation tier (tier-based architecture [17]) or view element (MVC-base architecture [18]). Examples include different background colors or different element sorting in lists.
- **Medium: Feature:** Changes influencing the logic tier in tier-based architecture or the model or controller element in a MVC-based architecture. Examples include the changes in workflow or the addition of specific functionality.
- **High: Full:** Variability of this level can influence multiple tiers at the same time and can be specific. Examples of this level of variability includes the ability for tenant to run their own program code.

The scope of this research is focussed on *runtime tenant-oriented low and medium variability in multi-tenant SaaS deployments*.

Design Patterns

The concept of patterns was first introduced by Christopher Alexander in his book about the architecture of towns [19]. This concept was quickly picked up in the software engineering world and led to the famous ‘Gang of Four’ pattern book by Gamma et al. [20]. This book describes several patterns that are still used today and does this in a way that inspired a lot of subsequent pattern authors. The definition of a pattern used in this paper originates from the Pattern Oriented Software Architecture series [21,22,23,24,25] and reads: “A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.”

Patterns are not artificially created artifacts, but evolve from best practices and experiences. The patterns described in this paper result from several case studies and discussions with experienced software architects. All patterns have proven to be a suitable solution for the problems described in section 5, since they are applied in successful SaaS products at our case companies. Also, all patterns are described language or platform independent, so the solution can be applied in various situations in the Software-as-a-Service domain. More information on future research concerning the patterns proposed can be found in section 6.

4 User-Variability Trade-off

The best solution for deploying a software product from a software vendor's perspective depends on level of resources shared and the level of variability needed to keep all users satisfied. In figure 1 four deployment solutions are introduced, that are considered best practices in the specific situations shown. In this section, the need for multi-tenant deployment models is explained.

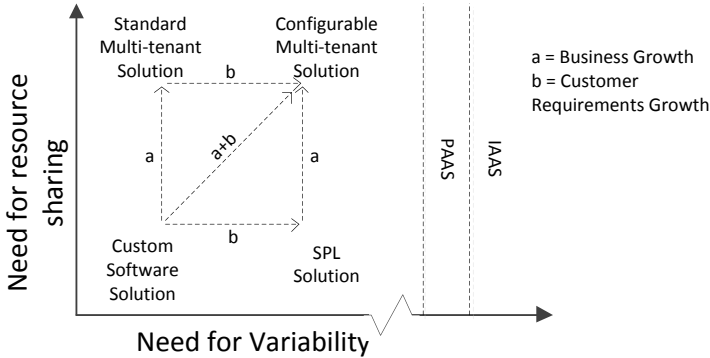


Fig. 1. Level of variability versus Number of users

By using the model shown in figure 1, software vendors can determine the best suited software deployment option. On the horizontal axis, the need for variability in a software product is depicted and the number of customers is shown on the vertical axis. For a small software vendor who does not have a lot of customers with specific wishes, a standard *custom software solution* is sufficient. The more customers software vendors get (business growth), the higher the need for a *standard multi-tenant solution* because of the advantages in maintenance. When the amount of specific customer wishes grows, software vendors can choose the software product line (SPL) approach to create variant for all customers having specific requirements. This solution can lead to a lot of extra maintenance issues as the number of customers grows. In case of a large number of customers having specific requirements, a *configurable multi-tenant solution* is the best solution for software vendors, keeping an eye on performance and maintenance.

5 Variability Patterns

In this section three patterns are described that were observed in the case studies that were conducted. The patterns are designed based on patterns observed within the case companies' software product, extended by patterns already documented in literature [20,16]. All patterns will be explained by an UML-diagram, together with descriptive topics proposed by Buschmann et al. [24] and Gamma et al. [20]

5.1 Customizable Data Views

In this section, a variability pattern is discussed, enabling developers to give tenants a way to indicate their preferences on the representation of data within the software product.



Fig. 2. Customizable Data Views Pattern

Intent - To give the tenant the ability to indicate and save his preferences on the representation of data shown.

Motivation - In a multi-tenant solution it is important to give tenants the feeling they can customize the product the way they want it. This customizability is most relevant in parts of the product where data is presented to the tenant.

Solution - In this variability pattern (cf. figure 2), the representation of data is performed at client side. Tenants can for example choose how they want to sort or filter their data, while the data-queries do not have to be adapted. The only change needed to a software product is the introduction of tenant-specific representation settings. In this table, all preferred font colors, sizes and sort option can be stored in order to retrieve this information on other occasions to display the data again, according to the tenant's wishes.

Explanation - As can be seen in the UML representation of the pattern in figure 2, the *DataRepresentation* class can manipulate the appearance of all data by making use of a *FunctionalComponent* able of sorting, filtering, etcetera. All settings are later stored by a *DataComponent* in a specific *UserSettings* table. Settings can later be retrieved by the same *DataComponent*, to be used again by the *DataRepresentation* class and *FunctionalModule*.

Consequences - By implementing this pattern, one extra table has to be implemented. Nothing changes in the way data selection queries have to be formatted. Representation of all data has to be formatted in a default way, except if a tenant changes this default way and stores his own preferences.

Example - In a bookkeeping program, a tenant for example, can decide what columns he wants to display and how he wants to order them. By clicking the columns he wants to display, his preferences are saved in the database. When the tenant uses the product again later, his preferences are fetched from the database and applied to his data.

5.2 Module Dependent Menu

This section describes a pattern to create dynamic menus, based on the modules associated to a tenant.

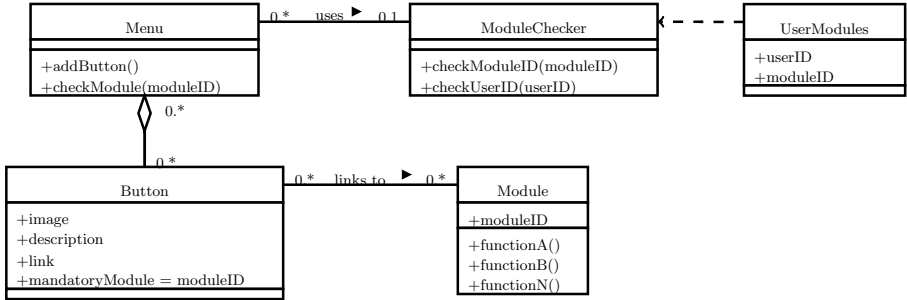


Fig. 3. Module Dependent Menu

Intent - To provide a custom menu to all tenants, only containing links to the functionality relevant to the tenant.

Motivation - Since all tenants have specific requirements to a software product, they can all use different sets of functionality. Displaying all possible functionality in the menu would decrease the user experience of tenants, so menus have to display only the functionality that is relevant to the tenant.

Solution - The pattern proposed (cf. figure 3), creates a menu out of different buttons based on the modules associated to the tenant. Every time a tenant displays the menu, the menu is built dynamically based on the modules he has selected or bought.

Explanation - The *Menu* class aggregates and displays different *buttons*, containing a link a specific module and the prerequisite for displaying this link (*mandatoryModule*). The selection of buttons is done, based on the results of the *ModuleChecker*. This class checks whether an entry is available in the *UserModules* table, containing both the ID of the tenant (user) and the mandatory module. If an entry is present, the *Menu* aggregates and displays the button corresponding to this module.

Consequences - To be able to use this pattern, an extra table containing user IDs and the modules available to this user has to be implemented. Also, the extra class *ModuleChecker* has to be implemented. All buttons do need a notion of a mandatory module that can be checked by the *ModuleChecker* to verify if a tenant wants or can have a link to the specific functionality.

Example - In a large bookkeeping product, containing several modules that can be bought by a tenant, the menus presented to the tenant can be dynamically composed based on the tenant’s license.’

5.3 Pre/Post Update Hooks

In this section a pattern is described, capable of implementing modules just before or after a data update.

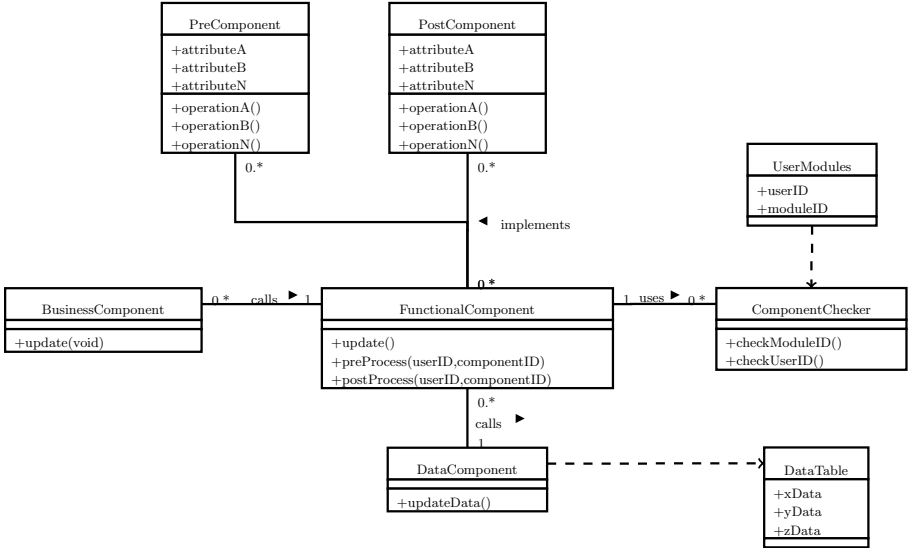


Fig. 4. Pre- Post Update Hooks

Intent - To provide the possibility for tenants to have custom functionality just before or after an event.

Motivation - In business oriented software, workflows often differ per tenant. To let the software product fit the tenants business processes best, extra actions could be made available to tentants before or after an event is called.

Solution - The pattern introduced here (cf. figure 4), makes use of a component able of calling other components before and after the update of data. The tenant-specific modules are listed in a separate table, similar to the pattern described in section 5.2.

Explanation - Before the *FunctionalComponent* calls the *BusinessComponent* in order to perform an update, the *ComponentChecker* is used to check the *UserModules* table if a tenant wants and may implements an extra component before the update is performed. After this, the *BusinessComponent* is called and the update is performed. The *DataComponent* takes care of the writing of data to a specific data table. After this, the *ComponentChecker* again checks the *UserModules* table and a possible *PostComponent* is called.

Consequences - Extra optional components have to be available in the software system in order to be able to implement this pattern. The amount and type of components available depends on the tenants' requirements.

Example - In a bookkeeping program, tenants can choose, whether they want to update a third party service as well by using a component that uses the API of a third party service to make changes there. If so, the `FunctionalComponent` can call the third party communicator after an internal update is requested.

6 Conclusion and Future Research

This paper gives a classification of different types of multi-tenancy and variability, enabling researchers to have one shared lexicon. Satisfying the need for a pragmatic overview on how to comply to the specific requirements of large numbers of customers, while keeping a product scalable and maintainable, this paper showed an introduction to the concept of variability in multi-tenant Software-as-a-Service solutions and presented three patterns gathered from industry case studies. By applying these patterns, companies can better serve customers and keep their software product maintainable and scalable. All three patterns are proven to be effective within the case companies and are reviewed by experts from the case companies, but still need to be analyzed more in terms of performance and effectiveness.

More VRTs still have to be identified and the effectiveness, maintainability, scalability and performance of all VRTs still has to be tested in future research. Currently a preliminary VRT evaluation model and testbed are being developed enabling researchers to test all identified VRTs and draw conclusions on their effectiveness. Also more case companies from other domains will be examined, enabling us to identify and test more VRTs.

Acknowledgment. This research is part of the Product as a Service project. Thanks goes to both case companies and their helpful architects, experts and developers.

References

1. Ma, D.: The Business Model of "Software-As-A-Service". In: IEEE International Conference on Services Computing, SCC 2007, pp. 701–702. IEEE, Los Alamitos (2007)
2. Gold, N., Mohan, A., Knight, C., Munro, M.: Understanding service-oriented software. *IEEE Software* 21(2), 71–77 (2005)
3. Turner, M., Budgen, D., Brereton, P.: Turning software into a service. *Computer* 36(10), 38–44 (2003)
4. Dubey, A., Wagle, D.: Delivering software as a service. *The McKinsey Quarterly* 6, 1–12 (2007)
5. Bezemer, C., Zaidman, A.: Multi-tenant SaaS applications: maintenance dream or nightmare? In: Proceedings of the International Workshop on Principles of Software Evolution (IWPSE), pp. 88–92. ACM, New York (2010)
6. Bezemer, C., Zaidman, A., Platzbeecker, B., Hurkmans, T., Hart, A.: Enabling multi-tenancy: An industrial experience report. In: 26th IEEE Int. Conf. on Software Maintenance, ICSM (2010)

7. Guo, C., Sun, W., Huang, Y., Wang, Z., Gao, B.: A framework for native multi-tenancy application development and management. In: The 9th IEEE International Conference on E-Commerce Technology, pp. 551–558 (2007)
8. Kwok, T., Nguyen, T., Lam, L.: A software as a service with multi-tenancy support for an electronic contract management application. In: IEEE International Conference on Services Computing, SCC 2008, vol. 2, pp. 179–186. IEEE, Los Alamitos (2008)
9. Yin, R.: Case study research: Design and methods. Sage Publications, Inc., Thousand Oaks (2009)
10. Hevner, A.R., March, S., Park, J., Ram, S.: Design science in information systems research. *Mis Quarterly* 28(1), 75–105 (2004)
11. Tremblay, M.C., Hevner, A.R., Berndt, D.J.: The Use of Focus Groups in Design Science Research. *Design Research in Information Systems* 22, 121–143 (2010)
12. Cooper, H.: *Synthesizing research: A guide for literature reviews* (1998)
13. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14(2), 131–164 (2009)
14. Schiller, O., Schiller, B., Brodt, A., Mitschang, B.: Native support of multi-tenancy in RDBMS for software as a service. In: *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 117–128. ACM, New York (2011)
15. Pohl, K., Böckle, G., van der Linden, F.: *Software product line engineering: foundations, principles, and techniques*. Springer-Verlag New York Inc., Secaucus (2005)
16. Svahnberg, M., van Gurp, J., Bosch, J.: A taxonomy of variability realization techniques. *Software: Practice and Experience* 35(8), 705–754 (2005)
17. Eckerson, W.: Three Tier Client/Server Architectures: Achieving Scalability, Performance, and Efficiency in Client/Server Applications. *Open Information Systems* 3(20), 46–50 (1995)
18. Krasner, G., Pope, S.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of Object Oriented Programming* 1(3), 26–49 (1988)
19. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: *A pattern language*. Oxford Univ. Pr., Oxford (1977)
20. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design patterns: elements of reusable object-oriented software*, vol. 206
21. Buschmann, F.: *Pattern-Oriented Software Architecture: A System of Patterns*, vol. 1. John Wiley & Sons, Chichester (1996)
22. Schmidt, D.: *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, vol. 2. Wiley, Chichester (2000)
23. Kircher, M., Jain, P.: *Pattern-Oriented Software Architecture : Patterns for Resource Management*, vol. 3 (2004)
24. Buschmann, F., Henney, K., Schmidt, D.: *Pattern-Oriented Software Architecture: Pattern Language for Distributed Computing*, vol. 4. Wiley, Chichester (2007)
25. Buschmann, F.: *Pattern-Oriented Software Architecture : On patterns and Pattern Languages*, vol. 5 (2007)

Preface to Onto.Com 2011

This volume collects articles presented at the first edition of the International Workshop on Ontologies and Conceptual Modeling (Onto.Com 2011). This workshop was organized as an activity of the Special Interest Group on Ontologies and Conceptual Modeling of the International Association of Ontologies and Applications (IAOA). It was held in the context of the 30th International Conference on Conceptual Modeling (ER 2011), in Brussels, Belgium. Moreover, the workshop was designed with the main goal of discussing the role played by formal ontology, philosophical logics, cognitive sciences and linguistics, as well as empirical studies in the development of theoretical foundations and engineering tools for conceptual modeling.

For this edition, we have received 18 submissions from Belgium, Canada, France, Germany, Italy, New Zealand, Russia, South Africa, Spain, Tunisia, United Kingdom, and the United States. These proposals were carefully reviewed by the members of our international program committee. After this process, 7 articles were chosen for presentation at the workshop. In the sequel, we elaborate on these selected submissions.

In the paper entitled “*Experimental Evaluation of an ontology-driven enterprise modeling language*”, Frederik Gailly and Geert Poels discuss an experiment to evaluate the use of an enterprise modeling language which was developed using the Resource Event Agent (REA) enterprise ontology and the Unified Foundational ontology (UFO). The effect of using the ontology-driven modeling language is analyzed using a well-known method evaluation model which contains both actual and perception-based variables for measuring the efficiency and effectiveness of the used method.

In “*Levels for Conceptual Modeling*”, Claudio Masolo proposes a (non-exclusive) alternative to taxonomic structuring based on subtyping relations in conceptual modeling. The author’s proposal relies on two relations: factual existential dependence and extensional atemporal parthood. On the basis of these relations, the author elaborates on a strategy to stratify object types in different levels, and to manage inheritance in a manner that addresses some classical difficulties in the modeling of this notion (e.g. attribute overriding, attribute hiding, or dynamic and multiple classifications and specialization).

In “*Principled Pragmatism: A Guide to the Adaptation of Ideas from Philosophical Disciplines to Conceptual Modeling*”, David W. Embley, Stephen W. Liddle, and Deryle W. Lonsdale discuss the synergism among the traditional disciplines of ontology, epistemology, logic, and linguistics and their potential for enhancing the discipline of conceptual modeling. The authors argue that application objectives, rather than philosophical tenets, should guide the adaptation of ideas from these disciplines to the area of conceptual modeling.

In “*Ontology Usage Schemes: A Working Proposal for the Ontological Foundation of Language Use*”, Frank Loebe proposes three theses regarding the relations between formal semantics, ontological semantics and representation systems. Based on these theses, the author outlines and illustrates a proposal for establishing usage-specific and ontology-based semantic schemes. Moreover, the author establishes a relation

between these findings and works regarding the specific case of conceptual modeling languages. Finally, he discusses potential applications of these proposals, including semantics-preserving translations and the re-engineering of representations.

In “*Gene Ontology based automated annotation: why it isn't working*”, Matthijs van der Kroon and Ana M. Levin propose an analysis of the current practice of ontology-based annotation in genome sequence applications. In particular, they analyze the current approaches based on the Gene Ontology (GO) and elaborate on the use of ontological categories to reflect on the pitfalls of these approaches.

In “*Formal Ontologies, Exemplars, Prototypes*”, Marcello Frixione and Antonio Lieto discuss a fundamental notion in Knowledge Representation and Ontology Specification, namely, the notion of “Concept”. In the paper, they discuss problematic aspects of this notion in cognitive science, arguing that Concept it is an overloaded term, referring to different sorts of cognitive phenomena. Moreover, they sketch some proposals for concept representation in formal ontologies which take advantage from suggestions coming from cognitive science and psychological research.

Finally, in the paper entitled “*Unintended Consequences of Class-based Ontological Commitment*”, by Roman Lukyanenko and Jeffrey Parsons elaborate on a rather controversial thesis, namely, that what appears to be a clear advantage of domain-specific ontologies, i.e., the explicit representation of domain semantics, may in fact impede domain understanding and result in domain information loss. Moreover, the paper discusses what the authors claim to be unintended consequences of class-based ontological commitment and advocates instead for the adoption of an instance-and property ontological foundation for semantic interoperability support.

We would like to thank the authors who considered Onto.Com as a forum for presentation of their high-quality work. Moreover, we thank our program committee members for their invaluable contribution with timely and professional reviews. Additionally, we are grateful to the support received by the IAOA (International Association for Ontologies and Applications). Finally, we would like to thank the ER 2011 workshop chairs and organization committee for giving us the opportunity to organize the workshop in this fruitful scientific environment.

July 2011

Giancarlo Guizzardi
Oscar Pastor
Yair Wand

Experimental Evaluation of an Ontology-Driven Enterprise Modeling Language

Frederik Gailly and Geert Poels

Faculty of Economic, Political and Social Sciences and Solvay Business School,
Vrije Universiteit Brussel
Frederik.Gailly@vub.ac.be
Faculty of Economics and Business Administration, Ghent University
Geert.Poels@ugent.be

Abstract. In this workshop paper an experiment is designed which evaluates the use of an enterprise modeling language that was developed with the Resource Event Agent enterprise ontology and the Unified Foundational ontology as a theoretical base. The effect of using the ontology-driven modeling language is analyzed using Moody's Method Evaluation Model which contains both actual and perception-based variables for measuring the efficiency and effectiveness of the used method.

1 Introduction

In the conceptual modeling domain it is generally accepted that ontologies can be used as a theoretical base for conceptual modeling languages [1]. This has resulted in using both core and domain ontologies for the identification of shortcomings in existing languages [2], the proposal of improvements to existing modeling languages [3] and the development of new conceptual modeling languages [4]. In most cases the validation of this research has been limited to demonstration of the shortcomings, improvements or languages using a relevant conceptual modeling cases.

In this research we plan to empirically evaluate an enterprise modeling language that is based on both a core ontology and a domain ontology. It is generally accepted that ontologies can support communication by making some basic assumption explicit, enable the reuse of domain knowledge and allow the creation of interoperability by sharing some common understanding about a domain [5]. These potential benefits are also relevant when ontologies are used for the development of modeling languages but are in most cases only used as a reason for employing ontologies but not actually proven by empirical research [6].

The empirical study designed in this paper wants to provide evidence that using a domain-specific modeling language which is engineered using a core and a domain ontology will result in on the one hand a model which is easier to integrate with other models because some general ontology axioms are taken into account that are also relevant for other domain. On the other hand we also want to evaluate the impact of using the method on the domain-specific quality of the model which we define as the degree to which a model respects the invariant conditions of a domain as axiomatised

in the domain ontology [7]. Finally as we believe that ontology-driven modeling languages also facilitate the reuse of domain knowledge we also want to evaluate the efficiency and the perceptions of the modeling language user with respect to the used method.

The ontology-driven modeling languages that will be empirically evaluated is the Resource Event Agent (REA) enterprise modeling language. This domain-specific modeling languages was developed in previous research [8] and is based on the Resource Event Agent enterprise ontology (REA-EO) [9] and the Unified Foundational ontology [7]. In the next section the REA ontology-driven modeling language is introduced. Section 3 develops the research model and introduces the hypotheses that will be analyzed by the experiment. In section 4 the actual design of the experiment is presented. The paper ends with a short conclusion and an overview the risks and limitations of the experimental design.

2 Enterprise Modeling Language

The enterprise modeling language used in this experiment is based on the Resource Event Agent enterprise ontology and the Unified Foundational Ontology. The REA-EO focuses on the creation of value in an enterprise and specifies that a business process or transaction consists of Economic Events that transfer or consume Economic Resources and Economic Agents that participate in these Economic Events. Additionally the ontology differentiates between different layers that deal with (i) the actual past and present in enterprise reality (i.e., what occurred or is occurring), (ii) the known future (i.e., what will occur), and (iii) the allowable states of the unknown future (i.e., what should occur). UFO is a core ontology developed by Guizzardi [7] and represents a synthesis of a selection of core ontologies. It is tailored towards applications in conceptual modeling. During the creation of the REA enterprise modeling language UFO is used for the ontological analysis of the REA-EO concepts, relations and axioms.

The development of the REA enterprise modeling language is described in [8] and results in the creation of a UML profile (i.e. the REA profile) that transforms the ontologies in a useful modeling language. The ontological evaluation of the REA-EO using UFO supports the development of the REA profile in two ways. Firstly it allows to define the stereotypes of the enterprise modeling language as specializations of the stereotypes of the OntoUML profile which is a general purpose modeling language that is core ontology-driven because it extends the UML class diagram metamodel with stereotypes that have their origin in UFO [10]. Secondly the ontological analysis also influences the development of the structuring rules for the enterprise modeling language because it identifies the domain-independent axioms, the domain-specific specializations of general axioms and the domain-specific extensions that need to be translated into structuring rules. In the profile the domain-independent axioms are inherited from the OntoUML profile and domain-specific axioms are implemented using OCL.

The REA-EO has been empirically evaluated for business modeling in previous research projects. These experiments have shown that being able to consult the REA enterprise ontology as a reference model, has a significant impact on the pragmatic quality of the developed models [11]. Moreover model users perceive diagrams with REA pattern occurrences as easier to interpret than diagrams without [12]. The main difference between this research and this previous work, is that in this experiment the REA ontology is not used as a reference model but instead is implemented as a domain specific modeling language with accompanying structuring rules that are based on the ontology axioms which have their origin in both a domain ontology and a core ontology.

3 Research Model and Hypotheses

The goal of this research is evaluating the use of ontology-driven modeling method for the creation of conceptual models. This is done by comparing the process of creating a business model using the REA enterprise modeling language with creating a business model with a domain-independent modeling language where the REA-EO is used as a reference model. This is operationalized in the research model (see figure 2) by distinguishing two modeling methods: REA ontology-driven modeling and enterprise modeling using REA as a reference.

As the focus of this research lies on evaluating a conceptual modeling method, the research framework is derived from the Method Evaluation Model (MEM) [13]. According to the MEM the success of a modeling method is determined by the efficacy and the adoption in practice. In this paper we will focus on the efficacy because this is a new method that is not used in practice and following the MEM the actual usage has a causal relation with the intention to use which has a causal relation with the efficacy.

In the MEM the efficacy is determined using both performance based (i.e. actual efficacy) and perception based (i.e. perceived efficacy) variables. The actual efficacy measures whether the method improves the performance of the task and is determined by the effort required to apply the method (i.e. efficiency) and the degree to which a method achieves its objectives (i.e. effectiveness). The actual efficiency is operationalized using the development time, defined as the time needed to create the model. As the goal of the ontology-driven modeling method is creating models that are easier to integrate and have a higher domain-specific quality, the effectiveness of the method is operationalized using two variables: (1) integration quality which measures the degree in which the part of the model that uses the REA modeling language integrates concepts that are not within the scope of the domain-specific modeling language and (2) domain-specific quality which we define as the degree to which a model respects the invariant conditions of a domain as axiomatized in the domain ontology.

The concept of perceived efficacy is operationalized by measuring the perceived ease of use and the perceived usefulness. On the one hand the perceived ease of use is a perception-based variable that measures the efficiency of using the modeling method and is defined as the degree to which a person believes that using the method would be free of effort. On the other hand the perceived usefulness measures the

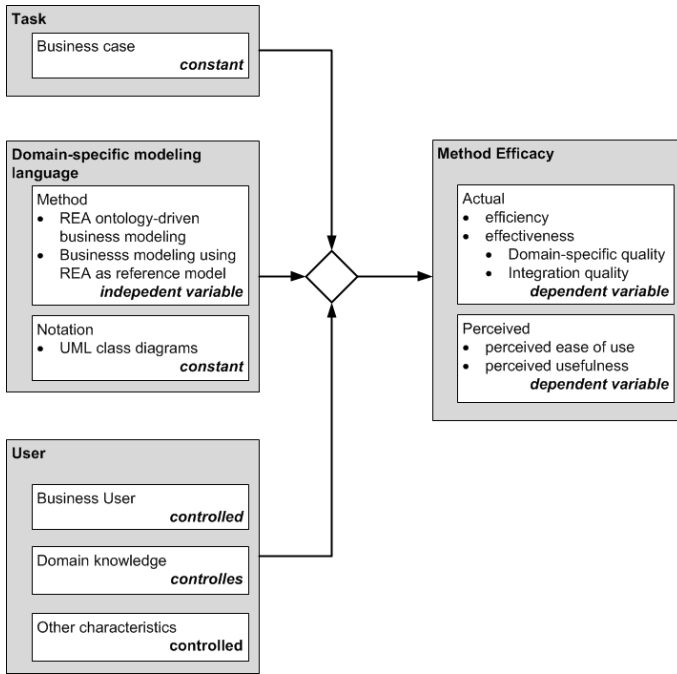


Fig. 1. Research model

effectives of using the modeling method and is defined as the degree to which a person believes that a particular method will be effective in achieving its attended objectives.

Following the work of Topi and Ramesh [14] the research model in figure 1 also contains the variables that need to be constant or controlled in the experiment because they could influence the performance and perception based variables and we only want to measure the influence of the used method. Firstly the characteristics of the modeler (e.g. domain knowledge, personal characteristics, modeling language knowledge) should be controlled. This will be done by randomly selecting the participants for a specific treatment (cfr. Infra). Secondly the task that should be performed is constant by making sure that all modelers must develop a business model for the same business case. Finally the used notation of the modeling language is also constant by using in both method the UML class diagram notation.

The specification of the experimental hypotheses is based on two different foundational theories and results in two different sets of hypotheses. On the one hand based on the contiguity principle of the multimedia learning theory of Mayer [15] we believe that when using an integrated tool that implements both the language and the rules instead of reference model with accompanying textual description of the axioms, the actual and perceived efficiency will be higher:

H_a: The efficiency of creating a business model using the REA profile is significantly higher when the model is created using the REA profile than when it is created using standard UML class diagrams with REA as a reference framework.

H_b: The perceived ease of use while creating a business model using the REA profile is significantly higher when the model is created using the REA profile than when it is created using standard UML class diagrams with REA as a reference framework.

On the other hand we also believe that the integration of the core ontology axioms and domain-specific axioms in the domain specific modeling language forces the modeler to take into account some general purpose and domain-specific modeling structuring rules which actually make it easier to create a high-quality model. This effect will be intensified by providing to the group that use the REA ontology as a reference model only the domain-specific axioms and not the domain independent axioms. Consequently the second hypothesis expects that modelers that use the profile will develop models with a higher-domain specific quality and higher integration quality, which should also result in a higher perceived usefulness:

H_c: The domain-specific quality of a business model created using the REA profile is significantly higher when the model is created using the REA profile than when it is created using standard UML class diagrams with REA as a reference framework.

H_d: The integration quality of a business model created using the REA profile is significantly higher when the model is created using the REA profile than when it is created using standard UML class diagrams with REA as a reference framework.

H_e: The perceived usefulness of creating a business model using the REA profile is significantly higher when the model is created using the REA profile than when it is modeled using standard UML class diagrams with REA as a reference framework.

4 Experimental Design

The potential users of an enterprise modeling language like REA are business people who want to model the value creation of the business processes. Consequently the participants of the experiments are students of a master in business engineering which have followed a business modeling course that contains topics such as conceptual data modeling (using ER modeling), business process modeling (using BPMN) and domain-specific modeling using profiles.

The dependent variable is operationalized by dividing the group of participants in two groups which will receive a different treatment. Group A will get one hour introduction into REA enterprise modeling where the REA-model is used as a reference model and the relevant axioms are described in text. Group B of students will also learn the REA enterprise modeling language but will learn this by means of the REA UML profile. Important to notice is that for both groups only a limited version of the REA enterprise modeling languages will be taught because the complete modeling languages is too complex to be taught in a limited time frame. As a consequence we decided to exclude the policy layer of the REA ontology and only take into account a limited set of structuring rules. The domain-specific structuring that will be thought to the all participants are represented in table 1. The domain-independent structuring rules that are only thought to group B and influence the integration quality of the model are presented in table 2.

Table 1. REA structuring rules based on domain-specific specializations

Structuring rule	Description
1	Every Economic Agent must be related to an Economic Event via an accountability or participation relationship
2	Instances of economic event must affect at least one instance of an economic resource by means of an inflow or outflow relation
3	Instances of economic event must be related to at least one instance of economic agent that is accountable for the event (i.e. inside agent) and to at least one instance of an economic agent that participates in the event (i.e. outside agent).
4	Instances of commitment must affect respectively at least one instance of an economic resource or economic resource type by means of a reserve or specify relationship.
5	Instances of commitment must be related to at least one instance of an economic agent that is accountable for the event (i.e. inside agent) and to at least one instance of an economic agent that participates in the event (i.e. outside agent).
6	A Commitment that is connected to an economic resource or economic resource type via a reserve-inflow or specify-inflow relation must be connected via a inflow-fulfill relation to an Economic Event that is connected to an inflow relation. A Commitment that is connected to an economic resource or economic resource type via a reserve-outflow or specify-outflow relation must be connected via an outflow-fulfill relation to an Economic Event that is connected to an outflow relation.

Table 2. REA structuring rules based domain-independent axioms

Structuring rule	Description
7	Instances of REA-EO Economic Resource and Economic Agents must always directly or indirectly be an instance of a Substance Sortal
8	Economic Resources and Economic Agents cannot include in their general collection more than one Substance Sortal
9	Economic Resources and Economic Agents cannot be specialized in an entity representing a Kind

After receiving the treatment the two groups will receive the same business case which is actually based on a existing business which sells trainings to their customers. The business case contains on the one hand the description of business process which contains both the planning and the actual execution of the training, can be modeled

using the REA profile or using the REA ontology as reference. Additionally the business case contains some information that is not within the scope of REA but is also relevant. The modeling of this information is given to the participants and should be integrated in the model. For instance the business case indicates that every available training is provided by a partner organization and a training is a special kind of product for the company. Moreover the company makes a distinction between a training and a planned training.

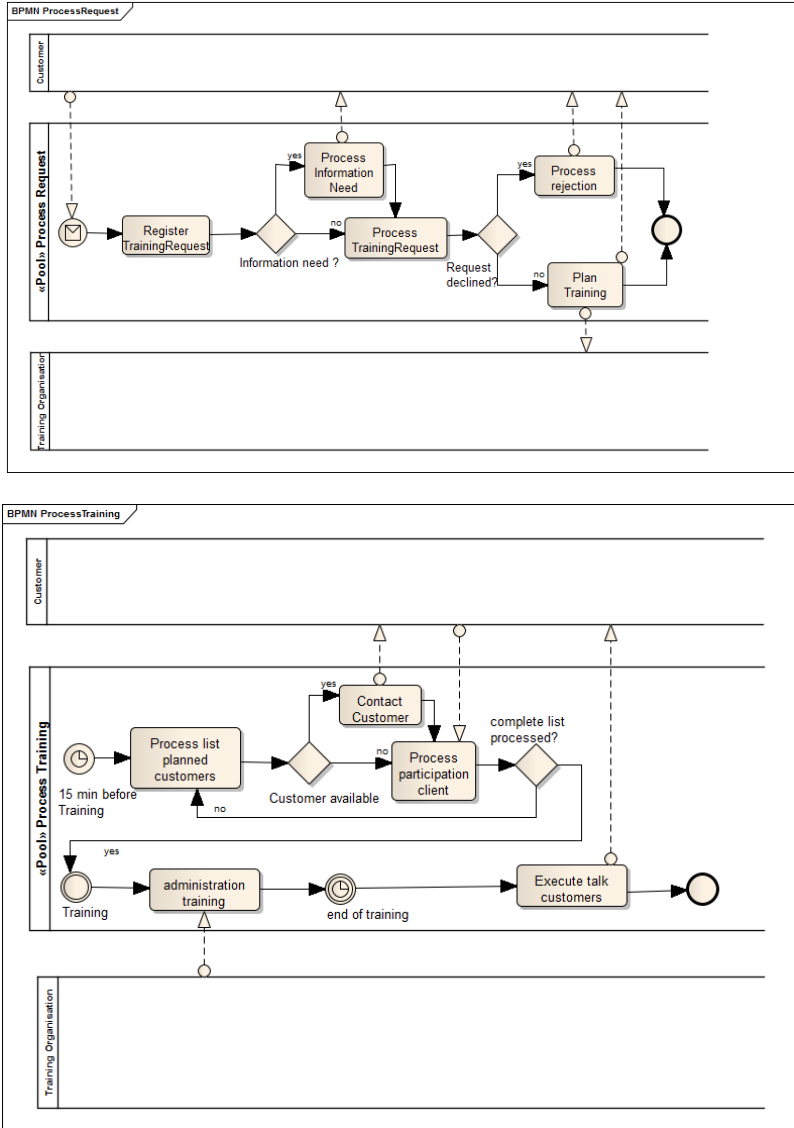


Fig. 2. Process models

Table 3. Perceived ease of use (PEOU)

PEOU1	I found the procedure for applying the method complex and difficult to follow
PEOU2	Overall, I found the method difficult to use
PEOU3	I found the method easy to learn
PEOU4	I found it difficult to apply the method to the business case
PEOU5	I found the rules of the method clear and easy to understand
PEOU6	I am not confident that I am now competent to apply this method in practice

Table 4. Perceived usefulness (PU)

PU1	I believe that the method reduces the effort required to create a business model
PU2	I believe that this method has improved my overall performance during the development of the business model
PU3	This method makes it easier for users to create business models
PU4	Overall, I found the method to be useful for the development of a business model
PU5	I believe that this method allows me to create business models more quickly
PU6	Overall, I think this method does not provide an effective solution to the problem of representing business models

5 Conclusion, Limitations and Risks

In this position paper the research model, the hypotheses and the design of an experiment is presented which has as goal the evaluation of an ontology-driven enterprise modeling language. It is expected that using language that has been developed using both a domain and core ontology has a positive impact on the efficiency and effectiveness of the method. The described experiment will be pre-tested in June 2011 and will be executed in October 2011.

The proposed experimental design has also some limitations and risks. First the external validity of the experiment might be limited by the fact that business students are used as modelers. This approach was followed because we want to control the characteristics of the users and because of difficulties in finding companies that want to participate in these type of experiments. However we believe the generalizability of the results is improved by using a real business case and by involving the business people that provides us with the case in the assessment of the quality of the models. A possible risk of the experiment is that no real differences will be found because a slim version of the ontology-driven modeling language is used. However we believe that this unavoidable because experience with previous experiments clearly showed that in a time frame of max 3 hours the complexity of the language must be bordered.

References

1. Wand, Y., Weber, R.: An Ontological Model of an Information System. *IEEE Transactions on Software Engineering* 16, 1282–1292 (1990)
2. Wand, Y., Storey, V.C., Weber, R.: An Ontological Analysis of the Relationship Construct in Conceptual Modeling. *ACM Transactions on Database Systems* 24 (1999)
3. Evermann, J., Wand, Y.: Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering* 10, 146–160 (2005)
4. Tairas, R., Mernik, M., Gray, J.: Using Ontologies in the Domain Analysis of Domain-Specific Languages. In: Chaudron, M.R.V. (ed.) *MODELS 2008*. LNCS, vol. 5421, pp. 332–342. Springer, Heidelberg (2009)
5. Gruninger, M., Lee, J.: *Ontology Applications and Design: Introduction*. *Communications of the ACM* 45, 39–41 (2002)
6. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. *Journal of Systems and Software* 84 (2011)
7. Guizzardi, G.: *Ontological Foundations for Structural Conceptual Models*. TelematicaInstituut cum laude. University of Twente, Twente (2005)
8. Gailly, F., Geerts, G., Poels, G.: Ontology-driven development of a domain-specific modeling language: the case of an enterprise modeling language. FEB working paper series. Ghent University (2011)
9. Geerts, G., McCarthy, W.E.: An Accounting Object Infrastructure for Knowledge Based Enterprise Models. *IEEE Intelligent Systems and Their Applications* 14, 89–94 (1999)
10. Benevides, A.B., Guizzardi, G.: A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML. In: Filipe, J., Cordeiro, J. (eds.) *Enterprise Information Systems*. LNBIP, vol. 24, pp. 528–538. Springer, Heidelberg (2009)
11. Poels, G., Maes, A., Gailly, F., Paemeleire, R.: The pragmatic quality of Resources-Events-Agents diagrams: an experimental evaluation. *Information Systems Journal* 21, 63–89 (2011)
12. Poels, G.: Understanding Business Domain Models: The Effect of Recognizing Resource-Event-Agent Conceptual Modeling Structures. *Journal of Database Management* 21 (2011)
13. Moody, D.: The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods. In: *11th European Conference on Information Systems*, ECIS 2003 (2003)
14. Topi, H., Ramesh, V.: Human Factors research on Data Modeling: A review of Prior Research, An extended Framework and Future Research Directions. *Journal Database Management* 13, 3–19 (2002)
15. Moreno, R., Mayer, R.E.: Cognitive Principles of Multimedia Learning: The Role of Modality and Contiguity. *Journal of Educational Psychology* 91, 358–368 (1999)
16. Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13, 319–340 (1989)
17. Maes, A., Poels, G.: Evaluating quality of conceptual modelling scripts based on user perceptions. *Data & Knowledge Engineering* 63, 701–724 (2007)

Levels for Conceptual Modeling

Claudio Masolo

Laboratory for Applied Ontology, ISTC-CNR, Trento, Italy
masolo@loa-cnr.it

Abstract. Usually object types are organized in taxonomies by means of a *specialization* relation (also called *subtyping* or *isa*) ‘implemented’ by means of *inheritance*. This paper proposes a (non-incompatible) alternative to taxonomies that relies on three primitives: *grounding*, a specific kind of factual existential dependence, extensional atemporal *parthood*, and *existence* at a time. On the basis of these relations, *specific*, *generic*, and *compositional* grounding relations between object types are introduced. By clearly separating the objects from the substrata on which they are grounded, these grounding relations allow to stratify object types in *levels* and to manage inheritance in a flexible way. In particular, this approach helps to avoid *isa* overloading and to overcome some classical difficulties related to inheritance, e.g. attribute overriding, attribute hiding, or dynamic and multiple classification and specialization, that are relevant aspects especially in modeling *roles*.

Keywords: Grounding, Dependence, Levels, Taxonomies, Inheritance.

Classification schemes – taxonomies based on subtyping (*isa*) among object types – and *inheritance* are central notions in conceptual modeling (CM) and in object-oriented modeling. By assuming, for instance, that Statue is a subtype of Amount Of Matter¹, Statue inherits all the attributes and associations² of Amount Of Matter. However, new attributes can be introduced. For instance, Statue, but not Amount Of Matter, could have the attribute Style. Similarly *roles*³ like Student, Customer, or President could be modeled as subtypes of Person. Student, but not Person, has a Matriculation. Customer, but not Person, has a Code, etc. This powerful mechanism of inheritance faces some well known problems. Statue and Amount Of Matter could have different values for the ‘same’ attribute, e.g. Price: a statue could be more expensive than a brute piece of bronze. Customer, differently from Person, could not have Weight or Place Of Birth. Attribute *overriding* and *hiding* try to manage these problems. Furthermore, roles can be played by objects with ‘incompatible’ attributes. For instance, both companies and persons can be customers, but Customer is neither

¹ Amounts of matter are concrete entities, pieces of matter, specific sums of molecules, e.g. the piece of gold that now constitutes my ring, not ‘the gold’ or ‘gold’.

² In this paper I will focus only on data modeling and not on behavior modeling.

³ The term *role* indicates here a specific kind of object types (properties). Roles intended as ‘parts in relationships’ are close to *relational roles* (see [12]).

a subtype nor a supertype of both Company and Person. “[W]e have the paradoxical situation that, from the extensional point of view, roles are supertypes statically, while dynamically they are subtypes” ([17], p.90). While keeping the same domain, this problem can be managed by adding new objects types, e.g. Private Customer (subtype of both Person and Customer) and Corporate Customer (subtype of both Company and Customer) [7], or by introducing dynamic and multiple classification and specialization (see [17] for a review). Alternatively, more *permissive* or *multiplicative* approaches extend the domain with new entities. Steimann [17] separates *natural types* (e.g. Person) from *role types* (e.g. Customer). Roles are *adjunct instances* linked by a *played-by* relation to their players (the persons or companies in the case of customers). The object and its roles form an aggregate and “the dynamic picking up of a role corresponds to the creation of a new instance of the corresponding role type and its integration in a compound, and dropping a role means releasing the role instance from the unit and destroy it” ([17], p.91). In object-oriented database management systems, by distinguishing *specialization*, an abstract concept, from *inheritance*, a mechanism that implements specialization, [1] systematically multiplies the instances in the presence of a subtype relation. If P is a subtype of Q , then the creation of an object p of type P produces the creation of an object q of type Q plus a link between them that allows p to inherit attributes from q . An object then is implemented “by multiple instances which represent its many faceted nature. Those instances are linked together through aggregation links in a specialization relation” ([1], p.561). The attributes are locally defined and stored but additional ones can be inherited via the links between the instances. From a more foundational perspective, multiplicative approaches have been investigated to solve the *counting problem* [9]. For instance, to count the Alitalia passengers (during 2010), one cannot just count the persons that flew Alitalia (during 2010). By adding *qua-entities* [12], (sum of) *relational tropes* [7], or *role-holders* [13] – entities that *inhere in* (a sort of existential specific dependence), but are different from, the players (see Section 1 for more details) – the counting problem is solved. In philosophy, multiplicativism is often considered also in the case of statues, organisms, tables, etc. (see [14] for a review and [3] for a recent defense). Interestingly, qua-entities have been originally introduced in this contest [6]. As in the case of roles, statues and amounts of matter have different properties (in particular causal properties) and different persistence conditions. The amount of matter that constitutes a specific statue can change through time. Or, an amount of matter can constitute some statue only during a part of its life, when it is statue-shaped. Therefore, some authors assume that statues are *constituted by* (a sort of existential dependence), but different from, amounts of matter.

Taxonomies are undeniably an important conceptual tool to organize object types according to the set-theoretical inclusion between their extensions. But it is not the only one. This paper proposes a *complementary* structuring mechanism founded on a specific kind of *existential dependence* called *grounding*. This mechanism allows to account for both roles and material objects with a flexible management of inheritance that helps to avoid isa overloading and misuse.

1 Statues, Customers, Presidents, and Tables

Let us assume that statues can change their material support through time while maintaining their shape, i.e. the shape, not the material support, is *essential* for statues. It follows that *Statue* is not a subtype of *Amount Of Matter*. One can represent ‘being a statue’ as a binary predicate with a temporal argument, $\text{Statue}_t x$ stands for “at the time t , the amount of matter x is a statue (is statue-shaped)” (c11)⁴. According to (d1), ‘being a statue’ becomes a sort of (relational) role played by amounts of matter. Counting seems unproblematic: the statues present at t are the amounts of matter that are statue-shaped at t . However, problems arise by considering a non atomic time, e.g. the whole 2010. A statue could change its material support during 2010, i.e. we could have two amounts of matter that are statue-shaped during 2010 but only one single statue. On the other side, if the same amount of matter, at a given time, is the support of two different statues, then we have one amount of matter but two statues. This sounds wrong because one usually excludes co-location of statues. Different are the cases of artifacts intended as (material) objects with an assigned (by the creator) functionality [4], and roles where, for example, at a given time the same person can be the customer of different companies or a multiple-customer of the the same company (see [12] for more details). The strategy to multiply predicates, e.g. one specialization of *Statue* for each statue, incurs in the problem of expressing what is the exact property that identifies the amounts of matter that, for instance, ‘are’ David at different times.

$$\text{d1} \quad \text{Statue}_t x \triangleq \text{AmountOfMatter} x \wedge x \text{HasShape}_{t,y} \wedge \text{StatueShaped}_t y$$

A multiplicative approach helps in managing these problems. In the literature, the nature and the relations among different kinds of entities are discussed.

Four-dimensionalism (see [15]) accepts spatio-temporal-worms. A statue, say *david*, and the amount of matter *m* that constitutes *david* only during a part of its life, are two overlapping but different *worms*: some *temporal slices* of *david* are not part of *m*. Problems can arise when *david* and *m* coincide (share the same slices) during their whole lives. Some approaches refuse spatio-temporal coincidence. Other approaches support a *modal* distinction founded on slices spreading across *possible worlds*: *david* and *m* are different world-spatio-temporal worms because *david* can exist without coinciding with *m* (and vice versa).

In a *three dimensionalist* perspective, *multiplicative* positions (see [18]) assume that statues (generically) *existentially depend* on, more precisely they are *constituted* by, amounts of matter without overlapping with them. In particular, Fine [6] analyzes *constitution* on the basis of the notion of *qua-entity*. If an object a , the *basis*, instantiates a property P , the *gloss*, then there exists an additional entity, $a\text{-qua-}P$ that is a sort of amalgam of a and P ⁵. The entity $a\text{-qua-}P$, e.g. $m\text{-qua-}s\text{-shaped}$ ($m\text{-qua-}having\text{-shape-}s$) exists at every time at which a instantiates P , it is uniquely determined by a and P , and it can inherit (not necessarily

⁴ To avoid reference to shapes one can consider $\text{StatueShaped}_t x$ where x is an object.

⁵ Qua-entities seem similar to *states of affairs* as defined in [2].

all) the properties of a . Therefore, by assuming that at a given time t an amount of matter m can have only an unique shape, say s , then only a single m -qua- s -shaped entity exists at t . On the other hand, a s -shaped amount of matter $m' \neq m$ generates, say at t' , a necessarily different qua-entity m' -qua- s -shaped. If m and m' constitute, respectively at t and t' , one single statue, then still we have two qua-entities and just one statue. According to [6], statues are *mereological sums* of qua-entities (m -qua- s -shaped + m' -qua- s -shaped) aggregated by (spatio-temporal) unity criteria [6].

Because of their relational nature, roles (and artifacts) are more controversial than material objects. While, at a given time t , amounts of matter have an unique shape, the same person, e.g. john, can be simultaneously a customer of different companies, e.g. alitalia and airfrance, i.e. both john-qua-customer-of-alitalia and john-qua-customer-of-airfrance can exist at t [7]. Moreover, differently from statues, customers always depend on the same person. This implies that different customers can share the same support, the same player [8]. Second, 'the president of Italy' and 'the president of Alitalia', are 'constituted-by' different persons through time and they can also share the same support at some time (somebody can be both the president of Italy and the president of Fiat). Therefore, in the case of roles, both the nature of the glosses [9] and the unity criteria are quite heterogeneous. Customers have always the same support (player) because they are discriminated on the basis of the glosses, e.g. 'being a customer of Alitalia' vs. 'being a customer of Airfrance' (see *saturated roles* in [12]) while presidents require unity criteria based on laws or social rules because spatio-temporal considerations are no relevant [10].

The same abstract mechanism works also for *structured* objects. For instance, one can think that (a specific kind of) tables necessarily have four legs and one top even though it is possible to substitute them during their life. In this case tables can be aggregates of qua-entities where the basis is a complex object, e.g.

⁶ Differently from classical temporal slices (see the definition in [15]), qua-entities persist through time when the basis instantiates the gloss during a whole interval.

⁷ Here 'customer-of' is a relation defined on persons and companies. Qua-entities are then identified by a person and a property like 'being a customer of company A '. DBs often add customer codes that, however, in general, are *keys* to identify *persons* not customers. This is due to the fact that DBs do not refer to persons, they just manage cluster of attributes (e.g. Name, Date Of Birth, etc.) that do not always identify persons. Customer codes could be conceptually necessary when the same person can have different customer roles inside the same company according to, for instance, his/her rights or obligations. In this case, the way qua-entities are identified is different because there is a third argument in 'customer-of'.

⁸ In this view customers coincide with single qua-entities, a limit case of mereological sum, that have the 'form' *person-qua-customer-of- A* . This explains why multiplicativist models of roles often consider only qua-entities and not general sums.

⁹ Some authors claim that roles are necessary based on *anti-rigid* properties. I will not address here this topic.

¹⁰ It is not clear to me whether unity criteria that involve *diachronic* constraints are part of the glosses.

a sum of four legs and one top, and the gloss is a structural property reducible to some spatial relations holding between the legs and the top. In this case there are two unity criteria. A synchronic one that establishes how the legs and the top must be structured, and a diachronic one that establishes the allowed substitutions of legs and tops through time.

Despite the differences previously discussed, I think that a unified view on (structured and unstructured) material objects and roles is possible. At the end, all these kinds of objects have an *intensional* dimension, to be identified, they rely on intensional rules.

2 Founding Modeling Primitives on a Theory of Levels

I consider a temporally qualified and factual primitive called *grounding*: $x \prec_t y$ stands for “ x grounds y at t ”. Following Husserl and Fine, Correia [5] bases his theory of dependence on *grounding*, informally interpreted as “at t , the existence of x makes possible the one of y ” or “ y owes its existence at t to x ”. Grounding is factual because the usual *definition* of the *specific* existential dependence of x on y , i.e. $\Box(\mathbf{E}x \rightarrow \mathbf{E}y)$ (where $\mathbf{E}x$ stands for “ x exists”)¹¹, reduces dependence to “the necessary truth of a material conditional whose antecedent is about x only and whose consequent is about y only; and given that any such material conditional fails to express any ‘real’ relation between the two objects, it is hard to see how prefixing it with a necessary operator could change anything in this connection” ([5], p.58). Grounding is temporally qualified because the usual definition of the *generic* existential dependence of an object on an type P , i.e. $\Box(\mathbf{E}x \rightarrow \exists y(\mathbf{E}y \wedge Py))$, does not allow to represent on which object an object depends at a given time.

Even though I completely agree on these remarks, I consider here a notion of grounding that is stricter than the one of Correia, a notion somewhere in between pure existential dependence and constitution. Let us come back to qua-entities. Fine considers *a-qua-P* as an amalgam of a and P . From a purely existential perspective, *a-qua-P* depends on both a and P . If P is a relational property, e.g. ‘being the customer of alitalia’, then *john-qua-customer-of-alitalia* existentially depends not only on *john* but also on *alitalia*. Intuitively, my grounding aims at capturing the specific existential dependence between *john* and *john-qua-customer-of-alitalia* by excluding the one between *alitalia* and *john-qua-customer-of-alitalia*. To add intuitions. Let us suppose that ‘supplier-for’ is the inverse of ‘customer-of’, i.e. *john* is a customer of *alitalia* if and only if *alitalia* is a supplier for *john*. Ontologically, there are reasons to identify ‘customer-of’ and ‘supplier-for’. However also in this case, *john-qua-customer-of-alitalia* is intuitively different from *alitalia-qua-supplier-for-john* because we are changing the ‘perspective’, we are changing the basis (and therefore the gloss). In particular, even though the first qua-entity existentially depends on *alitalia*, it is strictly linked to (*directed* to and *thicker* than) *john*. Approaches based on constitution, often advocate spatial co-location. The constituted entity is co-located with the constituent entity.

¹¹ This definition has been modified to answer some criticisms. See [16] and [5].

In the case of qua-entities, *john-qua-customer-of-alitalia* is intuitively co-located with *john* not with *alitalia*. However, my grounding is defined on objects that are not necessarily in space. In addition, constitution (and supervenience [10]) often excludes relational properties from the ones that can ‘generate’ new (kinds of) entities. Aiming at managing roles, this constraints is too strong for my goal.

Formally, I simplify the theory in [11] by avoiding the temporal qualification of *parthood* and by discarding the primitive of *being at the same level as*.

Grounding is asymmetric, transitive, down linear (a1), and implies existence (a2), where the primitive $E_t x$ stands for “the object x exists, is present, at time t ”, or, more neutrally, “ x is temporally extended through the time t ” [12] *Direct grounding* (d2) captures one single grounding step.

Parthood, xPy stands for “ x is part of y ”, is a purely formal notion on the basis of which *overlap* (O) is defined as usual [16]. More precisely, I consider a *classical extensional mereology*: parthood is reflexive, antisymmetric, transitive, implies existence, and satisfies the strong supplementation principle (a3) guaranteeing that two objects with the same parts are identical [16]. *Mereological sums*, $sSM\{a_1, \dots, a_n\}$ stands for “ s is the mereological sum of a_1, \dots, a_n ” (d3), refer to ‘multitudes’ of objects without a strong ontological commitment. For instance, four legs and one top exist if and only if their mereological sum exists, but if they are disassembled no table exists [13] Grounding is not a specific kind of parthood. Differently from (improper) parthood, grounding is irreflexive (directly from asymmetry). Differently from proper parthood, grounding does not satisfy the strong (and the weak) supplementation principle. For example, the fact that an amount of matter m grounds a statue does not require the statue to be grounded on additional entities disjoint from m , i.e. m could be the only grounding of the statue. More strongly, I assume that grounding and parthood are incompatible: $x \prec_t y \rightarrow \neg xPy$. Note however that a grounding object is not necessarily atomic, i.e. it can have proper parts.

- | | | |
|-----------|--|--|
| a1 | $y \prec_t x \wedge z \prec_t x \rightarrow y \prec_t z \vee y = z \vee z \prec_t y$ | <i>(down linearity)</i> |
| a2 | $x \prec_t y \rightarrow E_t x \wedge E_t y$ | |
| a3 | $\neg xPy \rightarrow \exists z(zPx \wedge \neg zOy)$ | <i>(strong supplementation)</i> |
| a4 | $x \prec_t y \rightarrow (Tx \leftrightarrow \neg Ty)$ | <i>(T is a leaf type)</i> |
| a5 | $T_1 x \wedge T_2 y \wedge x \prec_t y \rightarrow \neg \exists z u t'(T_1 z \wedge T_2 u \wedge u \prec_t z)$ | <i>(T₁, T₂ leaf types)</i> |
| d2 | $x \odot_t y \triangleq x \prec_t y \wedge \neg \exists z(x \prec_t z \wedge z \prec_t y)$ | <i>(direct grounding)</i> |
| d3 | $sSM\{a_1, \dots, a_n\} \triangleq \forall z(zPs \leftrightarrow (zPa_1 \vee \dots \vee zPa_n))$ | <i>(mereological sum)</i> |

Levels are partially captured by (a finite set of) types that are assumed to be *non-empty* and *rigid* properties formally represented by (non temporally qualified) unary predicates T_i . Types can be extensionally organized in a taxonomy. *Leaf* types, types with no subtypes, partition the domain. According to (a4), grounding always relies on a difference in type that is expressible in the theory,

¹² I will focus here only on objects present at some time.

¹³ Sums need to be carefully managed because not all the summands necessarily exist at every time at which the sum exists.

i.e. grounding does not hold between objects belonging to the same leaf type. Together with (a5), it avoids grounding loops. (a4) and (a5) are basic requirements for structuring (leaf) types in *levels* that assure also the *maximality* (with respect to parthood) of the grounds.¹⁴

After all these technical details, I will now introduce three grounding relations useful to organize types in levels. The formal definitions characterize the semantic of these grounding relations, but, once understood, they can be used as conceptual modeling primitives. In this sense, according to the following quote, they can be seen as an abstraction, simplification, and hiding of the previous analysis: “The theoretical notions which are required for suitable characterizations of domain conceptualizations are of a complex nature. This puts emphasis on the need for appropriate computational support for hiding as much as possible this inherent complexity from conceptual modeling practitioners.” ([8], p.9).

T_1 is (directly) *specifically grounded* on T_2 (a6), noted $T_1 \triangleright T_2$, if every T_1 -object is (directly) grounded on a single T_2 -object during its whole life, e.g. **Customer** \triangleright **Person**. It is often motivated by emergent properties. **Customer** is no more modeled as a subtype of **Person**. **Customer** is now a rigid type, i.e. a customer is necessarily a customer, with specific attributes. I think this is a quite simplifying CM technique. Furthermore, the temporal extension of a customer is included in the one of the person (a different object) that grounds him, i.e., to exist, a customer requires a grounding person while persons do not require customers. We will see how (some of) the attributes of **Person** can be inherited by **Customer** and vice versa.

T_1 is (directly) *generically grounded* on T_2 (a7), noted $T_1 \blacktriangleright T_2$, if every T_1 -object is (directly) grounded on some, but not necessarily the same, T_2 -object, e.g. **Statue** \blacktriangleright **AmountOfMatter**. It is often motivated by different persistence conditions.¹⁵ Note that the proposed framework does not commit on a specific ontological theory of persistence. One can quantify on both statues and amounts of matter without including in the domain temporal slices, qua-entities, states of affairs, events, or tropes. Indeed without being forced to, the modeler can, through axioms that links statues and amounts of matter, make explicit the underlying theory of persistence (in addition to the unity criteria).

T is (directly and generically) *compositionally grounded* on T_1, \dots, T_m if every T -object is (directly) grounded on some, but not necessarily the same, mereological sum of T_1, \dots, T_m -objects. It is often motivated by structural relations among T_1, \dots, T_m -objects. I distinguish *definite* compositional grounding (a8)¹⁶, noted

¹⁴ In general, levels are not necessarily linear and they can be conceived as collections of objects that obey the same laws of nature, have common identity criteria or persistence conditions. These are interesting points for CM that deserve future work.

¹⁵ Customer are not completely determined by persons, nor statues by amounts of matters. Grounding does not necessarily imply reduction, it differs from *determination* used to explain supervenience, e.g. “The mental is dependent on the physical, or the physical determines the mental, roughly in the sense that the mental nature of a thing is entirely fixed by its physical nature” ([10], p.11).

¹⁶ In (a8) and (a9) $\neg T_1(x + y)$ is a shortcut for $\exists s(sSM\{x, y\} \wedge \neg T_1s) \vee \neg \exists s(sSM\{x, y\})$.

$\mathbf{T} \blacktriangleright \langle (n_1)\mathbf{T}_1; \dots; (n_m)\mathbf{T}_m \rangle$, e.g. $\mathbf{Table} \blacktriangleright \langle \mathbf{Surface}; (4)\mathbf{Leg} \rangle$ ¹⁷, i.e. when a table exists it is grounded on exactly one surface and four legs, from (at least) *indefinite* compositional grounding (a9), noted $\mathbf{T}_1 \blacktriangleright (\geq n)\mathbf{T}_2$, e.g. $\mathbf{Organism} \blacktriangleright (\geq 2)\mathbf{Cell}$, i.e. organisms are grounded on at least two cells even though the exact number of grounding cells can vary in time¹⁸. To count the grounding objects one must rely on clear principles that identify unitary objects. For example, I would exclude $\mathbf{Statue} \blacktriangleright (\geq 2)\mathbf{AmountOfMatter}$ and $\mathbf{Statue} \blacktriangleright (2)\mathbf{AmountOfMatter}$. Here I just assume a mereological principle, i.e. the grounding \mathbf{T}_i -objects does not overlap and their sums are not of type \mathbf{T}_1 (see (a8) and (a9))¹⁹

- a6** $\mathbf{T}_1 x \rightarrow \exists y (\mathbf{T}_2 y \wedge \forall t (\mathbf{E}_t x \rightarrow y \otimes_t x))$ (specific direct grounding)
- a7** $\mathbf{T}_1 x \rightarrow \forall t (\mathbf{E}_t x \rightarrow \exists y (\mathbf{T}_2 y \wedge y \otimes_t x))$ (generic direct grounding)
- a8** $\mathbf{T} x \rightarrow \forall t (\mathbf{E}_t x \rightarrow \exists y_{11} \dots y_{1n_1} \dots y_{m1} \dots y_{mn_m} s$
 $\mathbf{E}_t y_{11} \wedge \dots \wedge \mathbf{E}_t y_{mn_m} \wedge s \mathbf{SM} \{y_{11}, \dots, y_{mn_m}\} \wedge s \otimes_t x \wedge$
 $\mathbf{T}_1 y_{11} \wedge \dots \wedge \mathbf{T}_1 y_{1n_1} \wedge \neg y_{11} \mathbf{O} y_{12} \wedge \dots \wedge \neg y_{1, n_1-1} \mathbf{O} y_{1n_1} \wedge \neg \mathbf{T}_1 (y_{11} + y_{12}) \wedge \dots$
 \dots
 $\mathbf{T}_m y_{m1} \wedge \dots \wedge \mathbf{T}_m y_{mn_m} \wedge \neg y_{m1} \mathbf{O} y_{m2} \wedge \dots \wedge \neg \mathbf{T}_m (y_{m1} + y_{m2}) \wedge \dots$
- a9** $\mathbf{T}_1 x \rightarrow \forall t (\mathbf{E}_t x \rightarrow \exists s (s \otimes_t x \wedge \forall z (z \mathbf{P} s \rightarrow \exists u (u \mathbf{P} z \wedge \mathbf{T}_2 u)) \wedge$
 $\exists y_1 \dots y_n (\mathbf{E}_t y_1 \wedge \dots \wedge \mathbf{E}_t y_n \wedge y_1 \mathbf{P} s \wedge \dots \wedge y_n \mathbf{P} s \wedge \mathbf{T}_2 y_1 \wedge \dots \wedge \mathbf{T}_2 y_n \wedge$
 $\neg y_1 \mathbf{O} y_2 \wedge \dots \wedge \neg y_{n-1} \mathbf{O} y_n \wedge \neg \mathbf{T}_2 (y_1 + y_2) \wedge \neg \mathbf{T}_2 (y_1 + y_3) \wedge \dots))$

Generic (or specific) grounding relations can be easily combined. For example, $\mathbf{Kitchen} \blacktriangleright \langle \mathbf{Table}; \mathbf{Oven}; (\geq 2)\mathbf{Chair} \rangle$. To mix specific and generic (compositional) grounding, one just needs to introduce more elaborate definitions. E.g., $\mathbf{Car} \blacktriangleright \langle \triangleright \mathbf{Chassis}; \blacktriangleright \mathbf{Engine}; \blacktriangleright (4)\mathbf{Wheel}; \blacktriangleright (\geq 1)\mathbf{WindscreenWiper} \rangle$ (\triangleright is heterogeneous grounding) stands for “cars specifically depend on a chassis and generically depend on an engine, four wheels, and at least one windscreen wiper”.

Methodologically, one can start from the *fundamental* types, types that are not grounded²⁰, and then, according to the grounding relations, progressively arrange the other (leaf) types in layers. Figure 1 depicts a simple example (with only a fundamental type, namely $\mathbf{AmountOfMatter}$) that shows the weakness of the notion of level: types can be grounded on types that have a different distance from the fundamental level as in the case of Exhibition.

Inheritance. All the types involved in grounding relations are *rigid* and *disjoint* from the ones on which they are grounded. Customers, statues, and tables are

¹⁷ I write $\mathbf{Table} \blacktriangleright \langle \mathbf{Surface}; (4)\mathbf{Leg} \rangle$ instead of $\mathbf{Table} \blacktriangleright \langle (1)\mathbf{Surface}; (4)\mathbf{Leg} \rangle$. This is consistent with the fact that $\mathbf{T}_1 \blacktriangleright \mathbf{T}_2$ is equivalent to $\mathbf{T}_1 \blacktriangleright \langle (1)\mathbf{T}_2 \rangle$, i.e. generic compositional grounding is an extension of generic grounding.

¹⁸ ‘At most’ indefinite compositional grounding, *cardinality* constraints (for example, $\mathbf{FootballTeam} \blacktriangleright (11\dots 22)\mathbf{FootballPlayer}$). Moreover, indefinite compositional grounding can also be used to introduce levels of *granularity*, even though additional constraints are necessary (see [11] for a preliminary discussion).

¹⁹ *Specific* compositional grounding can be defined starting from the corresponding generic case by considering the ‘form’ in (a6) instead of the one in (a7).

²⁰ The existence of a (unique) fundamental level is debated in philosophy. However, in applicative terms, I don’t see any drawback in accepting fundamental types.

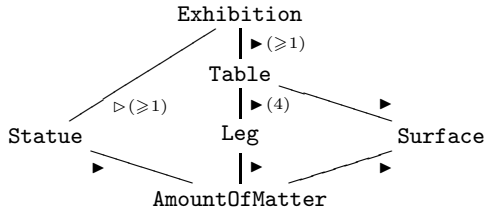


Fig. 1. Structuring object types according to grounding relations

such during their whole life. Grounding and subtyping are separate relations, therefore the problems due to isa overloading trivially disappear. As drawback, we loose the power of the inheritance mechanism. However, Baker [3] observes that constitution (a specific grounding) provides a *unity*, it allows the constituted entity to *inherit* (to derivatively have) *some* properties from the constituting one and *vice versa*.²¹ E.g. amounts of matter (persons) can inherit the *style* (*right to vote* for student representatives) from the statues (students) they ground.

On the basis of these observations, following [1], the inheritance of attributes of grounded types must be controlled. By default, $T_1 \triangleright T_2$ or $T_1 \blacktriangleright T_2$ implies that all the attributes of T_2 are inherited by T_1 . $T_1[A_1^1, \dots, A_n^1] \triangleright T_2[A_1^2, \dots, A_m^2]$ means that only the T_2 attributes A_1^2, \dots, A_m^2 are exported to T_1 while the T_1 attributes A_1^1, \dots, A_n^1 are exported to T_2 . Similarly in the case of generic grounding. $\text{Statue}[\text{Style}] \blacktriangleright \text{AmountOfMatter}$ means that *Statue* inherits all the attributes of *AmountOfMatter*, while the last type inherits only the attribute *Style* from *Statue*. In this way, attribute hiding can be trivially modeled. Attribute overriding can be approached by systematically override the attributes of the grounding type or by localizing all the attributes as in [1]. The case of compositional dependence is interesting. Some attributes of the grounded object can be obtained from a ‘composition’ of the attributes of the grounds. For example, the weight of tables is the sum of the weights of the grounding legs and surfaces. If necessary these rules can be explicitly added as constraints. Alternatively, one can add dependences among the values of attributes.

Grounding and Subtyping. It is trivial to prove that if $T_1 \Rightarrow T_2$ ²² and $T_2 \triangleright T_3$ then $T_1 \triangleright T_3$. Vice versa, from $T_1 \Rightarrow T_2$ and $T_1 \triangleright T_3$, $T_2 \triangleright T_3$ does not follow. Moreover, from $T_1 \triangleright T_2$ and $T_2 \Rightarrow T_3$ it follows that $T_1 \triangleright T_3$ but one loses the information about the specific subtype on which T_1 is grounded. A ‘parsimonious approach’ considers only maximally informative grounding relations $T_1 \triangleright T_2$: T_1 is *maximal* with respect to subtyping, while T_2 is *minimal*. This criterion (together with the fact that only direct grounding relations are considered) allows to clarify the nature of abstract types like *MaterialObject*. Let us assume $\text{Leg} \Rightarrow \text{MaterialObject}$, $\text{Surface} \Rightarrow \text{MaterialObject}$, and $\text{Table} \Rightarrow \text{MaterialObject}$ and compare the model that considers all the grounding relations in Figure [1]

²¹ However, high-level properties are not always reducible to properties of substrata.
²² \Rightarrow represents the subtyping relation. The following results hold also for generic dependence. Here I do not consider the composition of grounding relations.

with the one with only `MaterialObject`►`AmountOfMatter`. Given the same taxonomical information, only the first model makes explicit that `MaterialObject` is an abstract and multi-level type.

Acknowledgments. I would like to thank Emanuele Bottazzi, Nicola Guarino, Laure Vieu, and the reviewers for the helpful comments and discussions. This work has been carried out in the framework of the EuJoint project (Marie Curie IRSES Exchange Project n.247503) and the ICT4Law project (Converging Technologies project financed by Regione Piemonte).

References

1. Al-Jadir, L., Michel, L.: If we refuse the inheritance... In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.) DEXA 1999. LNCS, vol. 1677, pp. 569–572. Springer, Heidelberg (1999)
2. Armstrong, D.M.: *A World of States of Affairs*. Cambridge University Press, Cambridge (1997)
3. Baker, L.R.: *The Metaphysics of Everyday Life*. Cambridge University Press, Cambridge (2007)
4. Borgo, S., Vieu, L.: Artefacts in formal ontology. In: Meijers, A. (ed.) *Handbook of Philosophy of Technology and Engineering Sciences*, pp. 273–308. Elsevier, Amsterdam (2009)
5. Correia, F.: *Existential Dependence and Cognate Notions*. Ph.D. thesis, University of Geneva (2002)
6. Fine, K.: Acts, events and things. In: *Sixth International Wittgenstein Symposium, Kirchberg-Wechsel, Austria*, pp. 97–105 (1982)
7. Guizzardi, G.: *Ontological Foundations for Structural Conceptual Models*. Ph.D. thesis, University of Twente (2005)
8. Guizzardi, G., Halpin, T.: Ontological foundations for conceptual modeling. *Applied Ontology* 3(1-2), 1–12 (2008)
9. Gupta, A.: *The Logic of Common Nouns*. Phd thesis, Yale University (1980)
10. Kim, J.: *Mind in a Physical World*. MIT Press, Cambridge (2000)
11. Masolo, C.: Understanding ontological levels. In: Lin, F., Sattler, U. (eds.) *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pp. 258–268. AAAI Press, Menlo Park (2010)
12. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social roles and their descriptions. In: Dubois, D., Welty, C., Williams, M.A. (eds.) *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pp. 267–277 (2004)
13. Mizoguchi, R., Sunagawa, E., Kozaki, K., Kitamura, Y.: A model of roles within an ontology development tool: Hozo. *Applied Ontology* 2(2), 159–179 (2007)
14. Rea, M. (ed.): *Material Constitution*. Rowman and Littlefield Publishers (1996)
15. Sider, T.: *Four-Dimensionalism*. Clarendon Press, Oxford (2001)
16. Simons, P.: *Parts: a Study in Ontology*. Clarendon Press, Oxford (1987)
17. Steimann, F.: On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering* 35, 83–106 (2000)
18. Vieu, L., Borgo, S., Masolo, C.: Artefacts and roles: Modelling strategies in a multiplicative ontology. In: Eschenbach, C., Gruninger, M. (eds.) *Proceedings of Fifth International Conference on Formal Ontology and Information Systems (FOIS 2008)*, pp. 121–134. IOS Press, Amsterdam (2008)

Principled Pragmatism: A Guide to the Adaptation of Ideas from Philosophical Disciplines to Conceptual Modeling

David W. Embley¹, Stephen W. Liddle², and Deryle W. Lonsdale³

¹ Department of Computer Science

² Information Systems Department

³ Department of Linguistics and English Language,
Brigham Young University, Provo, Utah 84602, U.S.A.

Abstract. The synergism among the traditional disciplines of ontology, epistemology, logic, and linguistics and their potential for enhancing conceptual-modeling applications is not fully understood. Better understanding how to adapt ideas from these disciplines should lead to improved serviceability of conceptual modeling. We argue in this position paper, however, that application objectives, rather than philosophical tenets, should guide the adaptation of ideas from these disciplines. Thus, an appropriate balance of discipline-based theory and pragmatism should temper adaptations. We evaluate the principled pragmatism we advocate by presenting several case-study examples. Each illustrates that an appropriate adaptation of ideas from the disciplines of ontology, epistemology, logic, and linguistics can significantly guide conceptual-modeling research and help build successful conceptual-modeling applications.

1 Introduction

The applicability of ontology, epistemology, logic, and linguistics to conceptual modeling seems compelling. But what role should these disciplines play in facilitating conceptual-modeling applications? To what extent should conceptual-modeling researchers adopt or adapt philosophical ideas, positions, adages, and objectives from these disciplines? Must they be purists in their adaptation, or is it appropriate to leverage fundamental ideas and objectives and let the pragmatism of the application dictate the adoption and adaptation of theoretical tenets of the various perspectives within and surrounding these disciplines?

We argue in this position paper that application-dependent pragmatism should guide the adaptation of ideas from these disciplines to the various research directions within the conceptual-modeling community. In adapting ideas from these disciplines to conceptual modeling, we espouse the adage attributed to Einstein that everything should be made “as simple as possible, but no simpler.”

To establish our position, we first sketch our contextual view of ontology, epistemology, logic, linguistics, and conceptual modeling (Section [2.1](#)). We then argue that being solution-oriented requires appropriate selective adaptation of

ideas from these areas (Section 2.2). Appropriate selectivity requires tempering by two crucial, overarching considerations: the adaptation must be principled, and it must be pragmatic. From our perspective forcing purist views for adaptation may overly complicate the conceptual-modeling application in opposition to Einstein’s sufficiency-with-simplicity adage. To make our views concrete, we present several case-study examples to show how the principle of practical pragmatism has served and can further serve as a guide to adapting ideas to conceptual modeling (Section 3). Then, as we conclude (Section 4), we generalize and assert that the principled pragmatism we advocate is potentially more far-reaching in its implications than to just the case-study examples we use for illustration. It provides a vision and perspective for adapting philosophical disciplines to conceptual-modeling applications. Further, it answers in part the question about the relationship among “Ontology as an Artifact, Ontology as a Philosophical Discipline, Conceptual Modeling, and Metamodeling.”

2 Philosophical Disciplines and Conceptual Modeling

2.1 Contextual Overview

Ontology, as a field of philosophy, investigates problems of existence: what exists, how do we know what exists, how does an object’s existence relate to universal reality, and related questions. Deciding what reality is, and which relations characterize reality, are at the core of ontological investigation. A central theme of ontology is ontological commitment, which is about having enough evidence to commit to an object’s existence.

Epistemology studies knowledge and belief. It explores where knowledge comes from, how it is represented (including its structure), what its limits are, and how it can be used to refute assertions or support belief and discover truths. Important topics include how to quantify, describe, create, disseminate, and operationalize knowledge. For example, how much knowledge is necessary (or sufficient) for accomplishing a given task? How do we acquire and codify knowledge? What constitutes good evidence for justification of a belief?

Logic is about valid reasoning. It allows us to explore generalization, abstraction, and inferred relationships among objects that exist. Logic can be formalized in several different ways: proof theory, model theory, formal languages, linguistic expressions, mental representations, or graphical visualizations. Principles of correct reasoning guarantee the validity of inferences and set up systems of meaning that can be manipulated at higher levels of abstraction.

Linguistics investigates languages, either formal (e.g. logic or mathematics) or natural (i.e. human languages). Since language is crucial to communicating ideas, knowledge, beliefs, and logic, all of the above areas are of concern to linguists. So, too, are the structure, properties, contextual realities, and meaning of sounds, words, sentences, discourse, and dialog.

Conceptual modeling deals with computational representation of concepts and how to communicate these representations. Concept representation, from

a linguistically-informed semantics perspective, is in part referential or denotational: symbols relate to objectively verifiable, external, real-world objects. Communication about conceptualizations is social: symbols facilitate interpersonal or intergroup communication. Conceptual modeling comprises both perspectives: most model instances are explicitly denotational or referential in their content and serve a social function by facilitating communication among stakeholders.

2.2 Principled Pragmatism

Our approach to adapting ideas from philosophical disciplines to conceptual modeling is based on principled pragmatism. The overarching epistemological principle is that our work is ontology-driven. Our pragmatism views ontologies as formal specifications of some domain that are simple enough that users can encode them without extensive experience in knowledge engineering or the truth-maintenance systems typically used for large-scale ontology development. We maintain simplicity by minimally selecting and adapting ideas from each of the disciplines in a combination that appropriately enhances our application. By extension, we assert that these same principles hold for other conceptual-modeling applications. We believe, for example, in harmony with Smith [1], that more exacting and purer ontological descriptions would enhance integration and interoperability applications¹.

In summary, we assert our position: When adapting ideas from philosophical disciplines to conceptual modeling, we should find the right balance, being neither too dogmatic (insisting on a discipline-purist point of view) nor too dismissive (ignoring contributions other disciplines can make to conceptual-modeling applications).

3 Case Study Applications

To illustrate the principled pragmatism we advocate, we present five case-study examples. For each case study, we explain how the disciplines of ontology, epistemology, logic, and linguistics guide the theoretical conceptual-modeling underpinnings of these applications. We also show that as the applications become more complex, they draw more heavily on philosophical disciplines for

¹ Although we have engaged in a fair amount of research on integration (e.g., see [2] as a summary), we have observed (unfortunately) that other than having a formally defined conceptual-modeling language that encourages adherence to ontological principles, we do not know how to ensure that users will model in an ontologically correct way. (We thus omit integration and interoperability from our case-study examples.) We assert, however, that although applications should drive abstraction, abstraction should be proper—should maintain generally-agreed upon ontological descriptions for the application. Doing so will surely enhance integration applications since integration of conceptualizations across abstraction levels is less painful than integration across user-modeled peculiarities. Cost effectiveness in interoperability will likely drive us toward more exacting ontological descriptions, and ontology-oriented conceptual modelers can and should provide appropriate and needed guidance.

support. Thus, the relationship between philosophical disciplines and conceptual-modeling applications becomes blurred so that it is no longer clear whether the artifact produced is a conceptual model or whether it is an ontology drawing heavily on conceptualizations from epistemology, logic, and linguistics. Further, we argue: it does not matter. What matters is that the application is well-served by the artifact and that researchers are able to build serviceable applications by finding good synergistic combinations of ideas drawn from the overlapping disciplines of conceptual modeling, ontology, epistemology, logic, and linguistics.

3.1 Conceptual-Model-Based Information Extraction

The problem of information extraction is to turn raw text into searchable knowledge. It requires finding computational answers to the fundamental questions of ontology (“What exists?”), epistemology (“What is knowledge and how can it be represented?”), logic (“What facts are known or can be inferred from known facts?”), and linguistics (“What meaning do language symbols convey?”).

Answering these questions computationally leads naturally to a conceptual-modeling resolution of the information-extraction problem [3,4]. This should not be surprising since ontological conceptualizations have been the basis for formalizing information since the days of Aristotle [5], and they still provide the underlying schemas for today’s fact-filled databases. Our principled pragmatic approach to building an information-extraction system thus begins with a conceptual model that defines a narrow domain of interest. For example, the conceptual-model instance in Figure 1 models the information found in typical car ads. The conceptual-model instance answers the question of “what exists” in the narrow domain of interest and is therefore ontological in nature. Next, we linguistically ground the conceptual-model instance by saying what language symbols convey meaning for each of the object sets. In Figure 2, for example, the regular expression for the external representation of *Price* recognizes “\$11,995” in the car ad in Figure 3. The dictionary *CarMake.lexicon* includes an entry for “CHEVY”, and thus it, along with the recognized price and the other information in the car ad, are extracted and entered into a data repository (RDF triples in our implementation) whose schema (OWL in our implementation) corresponds to the conceptual-model instance in Figure 1. The populated conceptual-model instance is a model-theoretic interpretation and thus immediately supports inference via established logic systems (the Pellet reasoner in our implementation and SPARQL queries generated from free-form user input).

As for principled pragmatism, we emphasize that although the ideas we adapt for conceptual-model-based information extraction are foundational, we do not plumb the depths of these foundational ideas. Purists might not agree, for example, that the conceptualization we propose is even an ontology. We argue, however, that since our conceptualizations not only answer “What exists?” but also explain how we know what exists in terms of linguistic clues, they are ontologies. Similarly, purists might not agree that populating a data repository involves epistemology, but because it computationally answers a fundamental epistemological question (“How is knowledge acquired?”), it is epistemological.

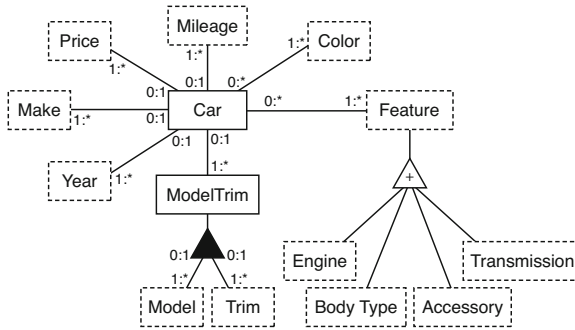


Fig. 1. Diagram for Conceptual-Model-Based Information Extraction

Price
internal representation: Integer
external representation: $\backslash\$\{1-9\}\backslash\text{d}\{0,2\},?\backslash\text{d}\{3\} \mid \backslash\text{d}?\backslash\text{d} \text{ [Gg]rand} \mid \dots$
context keywords: price|asking|obo|neg(\.|otiable) | ...
 ...
 LessThan(p1: Price, p2: Price) **returns** (Boolean)
context keywords: (less than | < | under | ...) \s*{p2} | ...
 ...
 Make
 ...
external representation: CarMake.lexicon
 ...

Fig. 2. Linguistic Grounding of *Price* and *Make* (partial)

Our use of logic and reasoning is likely not controversial, although we must be careful not to adopt so much that our system becomes undecidable or intractable. Our use of linguistics is intentionally simplistic: our lexical grounding computationally enables the conveyance of linguistic meaning without the necessity of the traditional depth of natural-language processing systems.

3.2 The Conceptual-Modeling Language OSM

Perspectives that motivate different approaches to conceptual-model language design are varied, but it is generally true that the target domain inspires the set constructs in the modeling language. In the context of database design, the ER model is natural, but in the context of real-time systems, a language like Statecharts is more appropriate. The challenge in modeling-language design is determining the “best” set of constructs to use: enough to cover the required concepts, but not so many that the language is too difficult to master. Also, it is important to define languages carefully and precisely, preferably with an underlying mathematical formalism that supports unambiguous definitions and even the ability to translate the model directly to an executing system.

Under the guise of the principled but pragmatic approach we advocate in this position paper, we created Object-oriented Systems Modeling (OSM), an

'97 CHEVY Cavalier, Red, 5 spd, only 7,000 miles on her.
 Previous owner heart broken! Asking only \$11,995.
 #1415 JERRY SEINER MIDVALE, 566-3800 or 566-3888

Fig. 3. Car Ad (unaltered from original)

object-oriented conceptual modeling language [6]. Figure 4 shows an example of a simple surveillance controller and illustrates how OSM describes objects (e.g. *Surveillance Controller*, *Dectector ID*, *Detection Event*, and *Timestamp* in Figure 4), relationships among objects (e.g. *Detector Event has Timestamp*), object behavior (e.g. event detection via *@detection* and controller reaction), and interaction among objects inside and outside the controller (e.g. *user abort*). It properly describes the behavior of a surveillance controller using a mixture of natural language and formal constructs, in real-world terms. And yet this model instance can directly execute in prototype fashion and can be fully formalized in terms of OSM to become operational. Thus, OSM is a conceptual-model programming language [7] as well as a conceptual-modeling analysis language.

In designing OSM, we tried to hold to “ontological” principles, believing that having the right abstractions and expressing concepts in natural terms (not “programming” terms) leads to better models. The ideas in Bunge’s ontology of physical systems [8] resonate nicely with our work on OSM. At the same time, we wove principles of epistemology and logic into the fabric of OSM. Populated OSM model instances constitute formal interpretations in model theory, and thus immediately provide a perspective on facts and knowledge (epistemology) and a basis for formal query and inference (logic). As a communication language, we used simple linguistic principles to make OSM human-readable. It is a small detail, but by convention we name object sets with spaces separating the words, not dashes or underscores as is more typical of many computer-friendly languages, and OSM provides for readable, sentence-like, relationship-set names (e.g. *Surveillance Controller has record of Detector Event* rather than more typical *detectedEvent* that is easier for a machine to parse).

3.3 Conceptualizations for Learning and Prediction

Recent initiatives by government agencies (e.g. IARPA-KDD [9]) and by academic think-tank groups (e.g. ACM-L [10]) require conceptualizations that track behavior, model what has happened and is currently happening, and analyze past and present behavior. The objectives of tracking, modeling, and analyzing include being able to predict future behavior, play out “what-if” scenarios, and warn of possible impending disasters. Although conceptual modeling can provide the foundation for storing the necessary information to support these initiatives, the lack of a unified, fact-oriented, conceptual-model formalism for temporal semantics, metamodeling, and reification leads to difficulties in reasoning about system properties and predicting future behavior from past behavior [11]. We can better serve these needs by a formalized conceptualization, called *OSM-Logic* [12], that more closely matches application demands.

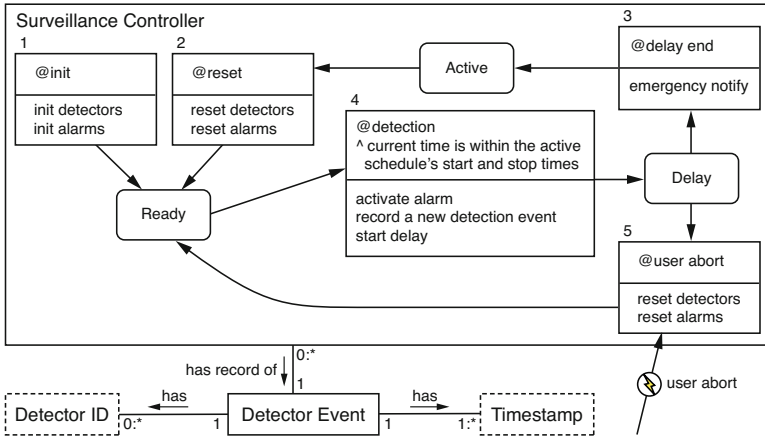


Fig. 4. OSM Diagram Illustrating Object Behavior

OSM-Logic illustrates both the “as simple as possible” and the “no simpler” aspects of principled pragmatism. The formalism is “no simpler” because the complexity of learning-and-prediction applications demands the inclusion of real-world temporality. The inclusion of temporality, however, is “as simple as possible” because it consists only of adding a timestamp argument to each event-fact predicate and a timestamp-argument pair to each existence-fact predicate. Thus, for example, in OSM-Logic, we formalize the fact that surveillance controller s is in the *Active* state of the behavior conceptualization in Figure 4 from time t to time t' as the predicate-calculus fact $inStateActive(s, t, t')$. Also of prime importance for simplicity, the temporal predicate-calculus formalism applies uniformly to all aspects of the modeling—the metamodel instance as well as the model instance, object behavior as well as object existence, and reification for both high-level and low-level specification. And yet, it is “no simpler” than necessary for these applications, because its temporal semantics provides for a temporal history of the existence and behavior of all objects, its time-dependent meta-modeling supports the tracking of model-instance evolution, and its reification properties make high-level abstractions as concrete and formal as low-level particulars, allowing users to either remove detail to survey the broader landscape or to drill down to the finest level of detail to find relevant facts and associations.

3.4 Multilingual Extraction Ontologies

Valuable local information is often available on the web, but encoded in a foreign language that non-local users do not understand. Hence the epistemological and linguistic problem: Can we create a system to allow a user to query in language L_1 for facts in a web page written in language L_2 ? We propose a suite of multilingual extraction ontologies as a solution to this problem [13]. We ground extraction ontologies in each language of interest, and we map both the

data and the metadata via the language-specific extraction ontologies through a central, language-agnostic ontology, reifying our ontological commitment to cross-linguistic information extraction in a particular domain. Our world model can thus be language-neutral and grounded in semantic and lexical equivalencies to a practical, operational degree [14]. This allows new languages to be added by only having to provide one language-specific ontology and its mapping to the language-agnostic ontology.

Mappings at several linguistic levels are required to assure multilingual functionality. Structural mappings (à la mappings for schema integration) associate related concepts across languages. Data-instance mappings mediate content of largely terminological nature: scalar units (i.e. fixed-scale measurements such as weight, volume, speed, and quantity); lexicons (i.e. words, phrases, and other lexical denotations); transliteration (i.e. rendering of proper names across orthographic systems); and currency conversions (i.e. mapping temporally varying indexes like monetary exchange rates). Most of this type of information is primarily referential in nature, with some element of pragmatics since commonly adopted standards and norms are implied, and these vary across cultural and geopolitical realms (e.g. use of the metric vs. imperial measurement systems). One other type of mapping, called commentary mappings, are added to the ontologies: these document cultural mismatches that may not be apparent to the monolingual information seeker, for example tipping practices in restaurants in the target culture.

3.5 Fact Extraction from Historical Documents

Many people are interested in explicit facts, as well as implied facts, found in historical documents. From the document snippet in Figure 5, taken from an online Ely Family History, facts such as *SonOf*(“William Gerard Lathrop”, “Mary Ely”, “Gerard Lathrop”) and *BornInYear*(“Mary Ely”, “1818”) are evident, as are implied facts such as *SurnameOf*(“Maria Jennings”, “Lathrop”) and *GrandmotherOf*(“Mary Ely”, “Maria Jennings Lathrop”). Images of these documents exist, and OCR engines can produce searchable text from these images.

To make explicit facts searchable, we can semantically annotate the OCR'd text of the original document images using conceptual-model-based information extraction. To make implicit facts searchable, we can generate them via logic rules. Unlike the assumption of a short, data-rich text description of a single item like in a classified ad, an obituary, or a Wikipedia article, historical documents have facts about many items (e.g. many people, places, and events in the Ely family history). Therefore, computational solutions require us to delve more deeply into ontology, epistemology, logic, and linguistics.

To obtain ontological commitment, we must address the question of having sufficient evidence to declare the existence of an object. For historical documents, the appearance of names for named entities is usually sufficient, but seeing the name of a make of a car like “Ford” in “The Ford Administration, 1974–1977” does not imply the existence of a car, although seeing “Ford” in “John drove the Ford he bought in 1925 for 20 years” does imply a car’s existence.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840.
3. Donald McKenzie, b. 1840, d. 1843. } Twins.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

Fig. 5. Text Snippet from Page 419 of the Ely Family History

In seeking computational solutions, we apply principled pragmatism by appropriately seeking for linguistic clues denoting the existence of things but do not go so far as to delve into ethereal metaphysical arguments about existence.

Regarding epistemology and logic, Plato, and those who follow his line of thought, demand of knowledge that it be a “justified true belief” [15]. Computationally, for facts in historical documents, we “establish” truth via provenance. Users can ask for fact authentication: the system responds by returning rule chains for inferred facts grounded in fact sources with extracted facts highlighted in images of original documents.

Linguistically, our goal for fact extraction from historical documents is to enable automated reading. We simplistically define reading as being able to extract fact instances with respect to a declared ontology. As a matter of principled pragmatism, we not delve into the depths of reading cognition; however, we can learn from the observation that people typically do not read data-rich text snippets like in Figure 5 left-to-right/top-to-bottom, but rather skip around—implying that fitting facts into an ontological conceptualization likely requires best-fit heuristics for “reading” the maximal number of correct facts.

4 Concluding Remarks

We have argued in this position paper that the relationship among “Ontology as an Artifact, Ontology as a Philosophical Discipline, Conceptual Modeling, and Metamodeling” is synergistic, but should be tempered with principled pragmatism. For conceptual-modeling applications the goal is practical serviceability, not philosophical enrichment. Thus, conceptual-model researchers should draw ideas and seek guidance from these disciplines to enhance conceptual-modeling applications, but should not become distracted from computationally practical solutions by insisting that philosophical tenets should prevail.

To show how principled pragmatism works for building conceptual-modeling applications, we presented several case-study examples. Tempered by Einstein’s sufficiency-with-simplicity adage, each case study illustrates how to directly leverage ideas from philosophical disciplines. Further, these case studies show that the requirements for applications to have computationally tractable solutions, itself, leads to principled pragmatism. Discipline principles can spark

ideas, but decidability, tractability, and usability requirements demand practicality. Lastly, the case studies show that appropriate adaptation of ideas from philosophical disciplines have enhanced the serviceability of conceptual modeling and moreover indicate (as future work) that adaptation will likely further enhance the serviceability of conceptual modeling.

References

1. Smith, B.: Ontology. In: Floridi, L. (ed.) *Blackwell Guide to the Philosophy of Computing and Information*, pp. 155–166. Blackwell, Oxford (2003)
2. Xu, L., Embley, D.W.: A composite approach to automating direct and indirect schema mappings. *Information Systems* 31(8), 697–732 (2006)
3. Embley, D.W., Campbell, D.M., Jiang, Y.S., Liddle, S.W., Lonsdale, D.W., Ng, Y.-K., Smith, R.D.: Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering* 31(3), 227–251 (1999)
4. Embley, D.W., Liddle, S.W., Lonsdale, D.W.: Conceptual modeling foundations for a web of knowledge. In: Embley, D.W., Thalheim, B. (eds.) *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. ch. 15, pp. 477–516. Springer, Heidelberg (2011)
5. Aristotle. *Metaphysics*. Oxford University Press, New York, about 350BC (1993 translation)
6. Embley, D.W., Kurtz, B.D., Woodfield, S.N.: *Object-oriented Systems Analysis: A Model-Driven Approach*. Prentice-Hall, Englewood Cliffs (1992)
7. Embley, D.W., Liddle, S.W., Pastor, O.: *Conceptual-Model Programming: A Manifesto*. ch. 1, pp. 3–16. Springer, Heidelberg (2011)
8. Bunge, M.A.: *Treatise on Basic Philosophy: Ontology II: A World of Systems*, vol. 4. Reidel, Boston (1979)
9. Knowledge discovery and dissemination program, http://www.iarpa.gov/-solicitations_kdd.html/
10. ACM-L-2010 Workshop, <http://www.cs.uta.fi/conferences/acm-l-2010/>
11. Liddle, S.W., Embley, D.W.: A common core for active conceptual modeling for learning from surprises. In: Chen, P.P., Wong, L.Y. (eds.) *ACM-L 2006*. LNCS, vol. 4512, pp. 47–56. Springer, Heidelberg (2007)
12. Clyde, S.W., Embley, D.W., Liddle, S.W., Woodfield, S.N.: *OSM-Logic: A Fact-Oriented, Time-Dependent Formalization of Object-oriented Systems Modeling* (2012) (submitted for publication, manuscript), www.deg.byu.edu/papers/
13. Embley, D.W., Liddle, S.W., Lonsdale, D.W., Tijerino, Y.: Multilingual ontologies for cross-language information extraction and semantic search. In: De Troyer, O., et al. (eds.) *ER 2011 Workshops*. LNCS, vol. 6999, Springer, Heidelberg (2011)
14. Nirenburg, S., Raskin, V., Onyshkevych, B.: *Apologiae ontologiae*. In: *Proceedings of the 1995 AAAI Spring Symposium: Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, Menlo Park, California, pp. 95–107 (1995)
15. Plato: *Theaetetus*. BiblioBazaar, LLC, Charleston, South Carolina, about 360BC (translated by Benjamin Jowett)

Ontological Usage Schemes

A Working Proposal for the Ontological Foundation of Language Use

Frank Loebe

Department of Computer Science (IfI) and Institute of Medical Informatics, Statistics and Epidemiology (IMISE), University of Leipzig, Germany
frank.loebe@informatik.uni-leipzig.de

Abstract. Inspired by contributing to the development of a top-level ontology and its formalization in logical languages, we discuss and defend three interrelated theses concerning the semantics of languages in general. The first is the claim that the usual formal semantics needs to be clearly distinguished from an ontological semantics, where the latter aims at explicating, at least partially, an ontological analysis of representations using a language. The second thesis is to utilize both types of semantics in parallel. Thirdly, it is argued that ontological semantics must be oriented at particular cases of using a language, which may lead to different manifestations of ontological semantics for one and the same language. Based on these views, we outline and illustrate our proposal for establishing usage-specific and ontology-based semantic schemes. Moreover, relations to works regarding conceptual modeling languages are established and potential applications are indicated, including semantics-preserving translations and the re-engineering of representations.

1 Introduction

1.1 Background and Motivation

The proposals in this article and the considerations that lead to them arose and have been pursued since then in the context of a long-term ontology development enterprise, building the top-level ontology General Formal Ontology (GFO) [1, 2]. Top-level ontologies naturally lend themselves to finding applications in numerous areas. This poses a challenge as regards the representation of the ontology, i.e., its provision in appropriate formats / languages (of primarily formal and semi-formal, and to some extent informal kind) for those different areas and for applications regarding different purposes. Due to this we are interested in translations among representations that provably preserve the semantics of the ontology.

Motivated by the ultimate goal of meaning-preserving translations, we are working towards formally defining the semantics of languages on the basis of

¹ <http://www.onto-med.de/ontologies/gfo>

ontologies. The latter is deemed an indispensable tool for meaning-preserving translations. In [11] we argue that a “truly” ontological semantics has not yet been provided with formal rigor. Herein we aim at defending central views in the same regard and extend our (work-in-progress) proposals concerning the semantics of languages. Despite focusing on logical languages in our use cases, the ideas appear generally transferable, including conceptual modeling languages.

1.2 Related Work and Paper Structure

There are two major lines of related work. In conceptual modeling, a general notion of ontological semantics is known, in the sense of providing mappings of language constructs to ontologies. In particular, starting around 1990, a series of works by Y. Wand, R. Weber, A.L. Opdahl, J. Evermann, G. Guizzardi and others provide an ontological semantics for constructs in the Unified Modeling Language (UML) or, more generally, for conceptual modeling languages; of these, [4,5,6,7,8,15] are relevant herein. We subscribe to many of the general ideas in these approaches and advocate similar positions below, at which we arrived in a logical setting. With hindsight and as far as we can see, nevertheless these approaches do not fully satisfy our goals, see the discussion in sect. 4.2.

The second line of research is concerned with the formal establishment of ontological semantics, e.g. [2,13]. However, the drawback of all approaches that we have become aware of is to define ontological semantics by recourse/reduction to accounts of formal semantics. This discounts some of our core presuppositions, as elaborated below, which lead us to seeking a different approach.

The remainder of the paper is structured as follows. Brief terminological remarks conclude the introduction. In the main sections we defend three strongly interrelated theses on language semantics and sketch the working proposal of *ontological usage schemes* to account for / implement the theses in assigning ontological semantics to languages in a formal way. The paper concludes with discussing some (intended) benefits and applications of the proposal (very briefly), as well as a reconsideration of related work and an outline of future work.

1.3 Terminological Preliminaries

Our discussion aims at covering a very broad range of (*representation*) *languages*, for which we expect a precisely defined syntax only. Despite a certain bias towards sentential / word languages the majority of the following ideas applies analogously to graphical / diagrammatic languages. Accordingly, we use *representation* as a general term for particular sets of expressions (R) of sentential languages (L) or particular diagrams (R) of a graphical language (L); formally, we write $R \subseteq L$ for R being stated in L . In semantic considerations of languages, we assume a domain of discourse that comprises entities. By *referential terms* we mean constant or variable symbols or symbolic constructions intended to denote entities in the domain of discourse, whereas *sentences* reflect states of affairs.

The term ‘*ontology*’ is used in a general sense for ‘theory of existential commitments’, which is not limited to highly general notions but may be very

domain-specific, in contrast to the usual understanding in conceptual modeling according to [8, sect. 3.3], [4, p. 235]. Ontology herein is refinable to *conceptualization* / *conceptual model* if the focus is on the theory contents “independently” of any language aspects, or to *formalized ontology* if a particular formalization language is assumed in which the conceptualization is represented.

Eventually, the notions of ontological analysis, ontological translation, and ontological foundation are of relevance herein. We refer to an *ontological analysis* as any process through which an ontology is established, i.e., a conceptualization is developed and possibly represented in informal and / or formal manner. Arbitrary domains as well as pre-established representations are the main target types of ontological analyses, according to our experience. Concerning representations, an *ontological translation* is a translation of a representation $R \subseteq L$ that results in a formalized ontology Ω which accounts for the ontological claims that *that* use of L commits to. Where the analysis does not start from scratch but utilizes available ontologies, reuse and integration become an aspect of ontological analysis. *Ontological foundation* refers to that part of ontological analysis that relates the notions targeted at with more general concepts, which are not among the analysis targets. A very simple case is to declare more general categories for target notions, e.g. that lion (an analysis target) specializes animal (in a core ontology) and continuant (in a top-level ontology).

2 Three Theses on Language Semantics

The number of (representation) languages is vast. They cover a broad range of purposes and domains which they were designed for. Two important landmark clusters in this spectrum are domain-specific languages and general-purpose languages (with representatives like UML and many logical languages, e.g. first-order logic (FOL), description logics (DLs) [1], or Semantic Web languages like the Web Ontology Language (OWL) [14]). All languages under consideration here are equipped with a precise, mostly formal definition of their semantics. By ‘formal’ here we refer to the use of abstract mathematical notions like sets, tuples, etc. in the definition of the semantics. The subsequent analysis focuses on such languages. The proposal of ontological usage schemes should be applicable to arbitrary (representation) languages, however.

2.1 Distinguishing Two Types of Semantics

The first important thesis herein is that the formal semantics of a language must be clearly distinguished from the ontological contents / commitment that a representation using the language may express in particular circumstances. Consider a DL role `hasPurchased`. Conceptually, this may refer to a relationship ‘has-purchased’ (which, in turn, may be derived from ‘purchase event’), connecting a person with a commodity. Yet, the standard formal semantics of a role in a DL interpretation structure is a set of pairs over the interpretation domain.

This basic distinction has already been defended in some detail by others, e.g. [8, sect. 2.1], and ourselves [11]. One supportive argument is that encodings of conceptual notions by mathematical constructs occur and lead to problems where mathematical properties correspond no longer to conceptual interrelations. Further, iterated translations among formalisms are problematic if encodings produce anomalies that aggregate over repeated translations. Nevertheless, the distinction between formal and ontological semantics is worth being stressed. At least, this is our observation regarding many proponents with a formal/logical background. For instance, major, seemingly accepted definitions of notions like ontology-based semantics and equivalence [2][3] and of “semantics-preserving translations” (like those between UML and OWL in the Ontology Definition Metamodel) purely rest on the formal semantics of the languages under study. For conceptual modeling and ontology research related with conceptual modeling the awareness of the difference may be greater, cf. [8, sect. 2.1].

2.2 Formal and Ontological Semantics in Parallel

Given the distinction between formal and ontological semantics, the question arises which type of semantics should be assigned to a language. On the one hand, a language with formal semantics has its benefits, including being very precise and exhibiting a well-understood mathematical basis. This supports clear and precise characterizations of dynamic aspects of the language as well as it allows for the verification of the correctness of algorithms / implementations. Moreover, many languages with formal semantics are tuned to particular purposes and are particularly successful regarding those – one may think of Petri nets [3] and their capabilities of behavioral analysis, for example. Ontological semantics, on the other hand, we deem necessary for translations between languages that preserve conceptual contents and, equivalently, necessary for semantic interoperability (of information systems). Clearly, almost every use of a language involves some conceptual contents. But either ontological semantics is not explicated at all, like frequently in knowledge representation, or it is not rigorously established, i.e., the link to ontologies is kept rather informal and leaves room for vagueness.

Thus far we conclude that an ontological account of semantics should be established for a language L in parallel to a (perhaps purpose-specific) formal semantics of L . At least, this should be done if the represented contents is (possibly) to be transferred to another form of representation. In the specific case where the purpose of a representation is the mere statement of ontological claims, formal and ontological semantics might collapse.

2.3 Establishing Ontological Semantics

The previous section leads us immediately to the question of *How to establish ontological semantics?* In the context of the development of the General Formal Ontology (cf. sect. [1.1]), we have been facing this question for the representation of ontologies themselves. As is nowadays well-established for some communities, the starting point was the use of logical languages, for us first order logic (FOL).

Resulting from detailed analysis and in connection with translation efforts, in [11] we argue that even general purpose languages like FOL cannot be considered to be ontologically neutral. Hence a novel account of semantics for logical syntax is required that assigns to a logical theory a *direct ontological semantics* (as a kind of referential, model-theoretic semantics, parametrized by an ontology). [11] outlines a corresponding approach for FOL syntax and eventually involves some methodological constraints for the use of FOL for formalizing ontologies, respecting the standard formal semantics, as well. However, a detailed elaboration is beyond the scope of this paper and is largely irrelevant in the sequel. Basically, it suffices to acknowledge that ontologies / ontological claims can be adequately captured in terms of logical theories.

Accordingly, a logically formalized ontology Ω_{base} forms the backbone of establishing *indirect ontological semantics*. By performing an ontological analysis of any representation $R \subseteq L$, such semantics can be defined by a translation of R or L “into an ontological theory”, i.e., an ontological translation (sect. 1.3). The latter phrases require some care, however. Notice that, for a substantially rich ontological translation, Ω_{base} must cover the conceptual contents of R (including the “hidden” knowledge) at R ’s level of specificity. In particular, the ontology must then elaborately account for all entities named in the representation. We expect that these circumstances are hard to meet. Therefore, striving for an incomplete / abstracting translation of R appears more realistic. This corresponds more closely to the *ontological foundation* for syntactic constituents of R or L .

Yet there is another aspect to be acknowledged. We argue that the ontological analysis of a representation must take each single referential constant into account, i.e., actually its intended referent. That means, defining ontological semantics usually cannot start at the same syntactic level that formal semantic definitions start from, even if one merely aims at an ontological foundation. Our point here is that *employed* atomic symbols of a representation R – i.e., symbols occurring in a particular use of the language L – form the bottom line for ontological analysis. From the point of view of L ’s grammatical definition, there may be a large set of potentially usable constants. But the latter are opaque to / not usable in typical definitions of formal language semantics. Each use of L , however, involves a much smaller subset of constants, which are equipped / linked with intended referents by the user of the language. We hold that this forms a significant part of the conceptual contents of R , in a sense claiming at least the existence of those entities named by constants. Of course, beyond constants, one may consider syntactic compositions and to what extent they contribute further ontological claims. The central conclusion that we draw in this connection is that ontological semantics should be oriented towards particular uses of a language L rather than finding “the” (single) ontological semantics for L . In particular, this applies to general-purpose formalisms, including logical languages and, to a similar extent, UML. Rephrased again, the grammatical distinctions in the syntax of the language are fairly limited, in contrast to the basically arbitrary choice of entities / notions that one might wish to capture in terms of a language. Consequently, one must expect that much more ontological distinctions exist than

could be mapped uniquely to available syntactic/grammatical distinctions. Thus it is reasonable to orient the assignment of ontological semantics at particular cases of usage of a language, and to have multiple possible ontological semantic definitions for one and the same language. It remains open to what extent the notion of *context* may be employed to account for this diversity.

Further support arises from encoding uses of a language, e.g. for reasons of expressiveness, exploitation of behavioral properties or algorithms provided for that language, etc. For such reasons, users may be lead to “misuse” a language.

3 Ontological Usage Schemes

In this section we present a general method to provide indirect ontological semantics for a language in a rigorous manner. Basically, the notion of ontological usage schemes (OUSes) refers to a specific form of defining an ontological translation. On the basis of our approach as summarized in sect. 2.3 and of the additional considerations therein, the following more precise definition is given. Despite its bias to sentential languages, we hope that the general approach behind becomes visible. Table 1 provides an illustration, which is briefly further discussed below.

Definition 1 (Ontological Usage Scheme). *For a language L , $Tm(L)$ denotes the set of referential terms (sect. 1.3) that can occur in representations using L .*

An ontological usage scheme (OUS) of a language L is captured as a structure $\mathcal{S}(L) = (L, L_\Omega, \Omega_{base}, \varepsilon, \sigma, \tau)$, where L_Ω is a logical language with direct ontological semantics that is established on the basis of the ontology Ω_{base} , where Ω_{base} is formalized in L_Ω . The remaining components are translation mappings: $\varepsilon : Tm(L) \rightarrow \wp(Tm(L_\Omega))$ maps referential terms of L into sets of such of L_Ω , $\sigma : Tm(L) \rightarrow \wp(L_\Omega)$ accounts for the ontological analysis of referential terms, and $\tau : L \rightarrow \wp(L_\Omega)$ explicates the ontological analysis of L expressions.

It should be stressed again that multiple OUS are perfectly reasonable for any single language L . Moreover, L_Ω and Ω_{base} are parameters in Def. 1, allowing for different languages and base ontologies. For our purposes, we have equipped FOL syntax with a direct ontological semantics [11, sect. 3.2] and thus use FOL (as L_Ω) in conjunction with the methodology mentioned in sect. 2.3 for ontology representation, including the ontology of categories and relations (outlined in [11, sect. 4]). The “schematic” aspect of ontological usage schemes becomes manifest in generic definitions for certain sets of identifiers, like N_C , N_R , and N_I in Table 1. This is similar to usual definitions of formal semantics and primarily useful for the ontological *foundation* of language elements.

It is important, on the one hand, that the translations ε , σ , and τ must be specified such that they formally capture the intended meaning of representations in L . On the other hand, they can be defined with great freedom concerning the (semi-)formal semantics of L . This is contrary to existing formal accounts of ontology-based semantics, like [2]. However, due to the encoding of conceptual notions into L ’s syntax and, thereby, semantics (see sect. 2.1), the translation of

Table 1. $\mathcal{S}(\text{ALC}) = (\text{ALC}, \text{FOL}, \mathcal{CR}, \varepsilon, \sigma, \tau)$, an exemplary ontological usage scheme for (a subset of) ALC syntax [11, ch. 2], based on an ontology of categories and relations, cf. [11], closely associated with the General Formal Ontology [9].

ALC SYNTAX	MAPPINGS ($\varepsilon, \sigma, \tau$; AUXILIARY: $\hat{\tau}_x^V$)
Vocabulary	
$C \in N_C$	$\varepsilon(C) = \{\varepsilon_C\}$
	$\sigma(C) = \{\varepsilon_C :: \text{Cat}, \varepsilon_C \supseteq \text{Ind}\}$
$R \in N_R$	$\varepsilon(R) = \{\varepsilon_R, q_1(\varepsilon_R), q_2(\varepsilon_R)\}$
	$\sigma(R) = \{\varepsilon_R :: \text{Reln}, q_1(\varepsilon_R) :: \text{RoleCat}, q_2(\varepsilon_R) :: \text{RoleCat}, \text{RoleBase}^2(\varepsilon_R, q_1(\varepsilon_R), q_2(\varepsilon_R))\}$
$a \in N_I$	$\varepsilon(a) = \{\varepsilon_a\}$
	$\sigma(a) = \{\varepsilon_a :: \text{Ind}\}$
Concepts	
$C \in N_C$	$\hat{\tau}_x^V(C) = x :: \varepsilon_C$
$C \sqcap D$	$\hat{\tau}_x^V(C \sqcap D) = x :: \hat{\tau}_x^V(C) \wedge x :: \hat{\tau}_x^V(D)$
$\neg C$	$\hat{\tau}_x^V(\neg C) = x :: \text{Ind} \wedge \neg x :: \hat{\tau}_x^V(C)$
$\exists R.C$	$\hat{\tau}_x^V(\exists R.C) = \exists y(y :: \text{Ind} \wedge \text{rel}^2(\varepsilon_R, q_1(\varepsilon_R), x, q_2(\varepsilon_R), y) \wedge \hat{\tau}_y^{V \cup \{y\}}(C))$ (for any new variable $y \notin V$)
TBox and ABox axioms (where C, D concepts, $a, b \in N_I$)	
$C \sqsubseteq D$	$\tau(C \sqsubseteq D) = \{\forall x. \hat{\tau}_x^{\{x\}}(C) \rightarrow \hat{\tau}_x^{\{x\}}(D)\}$
$C = D$	$\tau(C = D) = \{\forall x. \hat{\tau}_x^{\{x\}}(C) \leftrightarrow \hat{\tau}_x^{\{x\}}(D)\}$
$C(a)$	$\tau(C(a)) = \{\hat{\tau}_a^{\emptyset}(C)\}$
$R(a, b)$	$\tau(R(a, b)) = \{\text{rel}^2(\varepsilon_R, q_1(\varepsilon_R), \varepsilon_a, q_2(\varepsilon_R), \varepsilon_b)\}$

SYMBOL	N_C – (set of) concept names, N_R – role names, N_I – individual names
LEGEND:	$::$ – instantiation Reln – relation
	\supseteq – (extensional) specialization RoleCat – role category
	Cat – category RoleBase ² – role base (of a binary relation)
	Ind – individual rel ² – relator (binary)

an explicit representation in L should be given priority compared to additionally accounting for anything that is implicitly derivable by means of the formal semantics of L . If both can be achieved without corrupting adequate translation of what is explicitly represented, of course, this is the ideal case. Otherwise, it may be useful to identify mismatches between reasoning over the ontological contents of a representation and formal-semantic derivations in L .

Eventually and as just anticipated, OUSes allow for defining the formally captured, ontological theory that arises from a representation:

Definition 2 (Ontological Image). *Let $\mathcal{S}(L) = (L, L_\Omega, \Omega_{base}, \varepsilon, \sigma, \tau)$ an ontological usage scheme. For a representation $R \subseteq L$, the ontological image of R according to \mathcal{S} is the deductive closure of $\Omega_{base} \cup \bigcup_{t \in T_m(R)} \sigma(t) \cup \bigcup_{e \in R} \tau(e)$.*

Table 1 illustrates a (specification of a section of a) sample usage scheme $\mathcal{S}(\text{ALC}) = (\text{ALC}, \text{FOL}, \mathcal{CR}, \varepsilon, \sigma, \tau)$ for very common expressions in description

logics (DLs) [11, 2]. A detailed explanation of it does not fit here, but for readers aware of DL the scheme is designed to strongly correspond to an ontological reading of the standard translation between DL and FOL [11, sect. 4.2], i.e., DL concepts are understood as categories of (ontological) individuals, DL roles as binary relations among (ontological) individuals. In terms of Def. 1, this proximity between L and L_Ω is *no* prerequisite. For more intuitions on the ontology of categories and relations \mathcal{CR} , the reader is referred to [11, 10]. To complete the illustration with a very simple translation of a DL representation, the ontological image of the DL statement $\text{lion} \sqsubseteq \text{animal}$ is the deductive closure of the theory $\mathcal{CRU}\{\varepsilon_{\text{lion}} :: \text{Cat}, \varepsilon_{\text{lion}} \supseteq \text{Ind}, \varepsilon_{\text{animal}} :: \text{Cat}, \varepsilon_{\text{animal}} \supseteq \text{Ind}, \forall x.x :: \varepsilon_{\text{lion}} \rightarrow x :: \varepsilon_{\text{animal}}\}$.

4 Discussion

4.1 Applications and Benefits

Ontological semantics in general and ontological usage schemes (OUSes) in particular allow for at least the following applications. Firstly, the formal-logical, ontological theory resulting from applying an OUS to a particular representation R allows for *reasoning over the conceptual contents* of R . Such theories also form the basis of *notions of conceptual equivalence*. Notably, logical equivalence within those theories is a first candidate, cf. [2], but we are convinced that further refinements are required. Secondly, with notion(s) of conceptual equivalence (seemingly) *non-standard translations become justifiable*. In particular, these play a role when translating from less expressive to more expressive languages. ‘Expressiveness’ here refers to the distinctions available in the abstract syntax. For instance, consider the case of DLs and UML. (Almost all) DLs do not provide immediate means to express relationships with an arity greater than two, which leads to encoding proposals like [12]. Accordingly, catch-all language-level³ translations like the ones from UML to OWL and vice versa in the Ontology Definition Metamodel must fail in a number of cases, and circular translation chains lead to non-equivalent representations in the same language.

Thirdly, based on an OUS one may derive *constraints / rules for using a language* in accordance with that OUS. For instance, the disjointness of relations and purely non-relational categories suggests that a DL concept encoding an n -ary relation should be declared disjoint with every DL concept capturing a non-relational category. Fourthly, OUSes should be *useful for re-engineering purposes*, because frequently the ontological semantics of a representation is augmented / elaborated in greater detail. Consider the example of modeling ‘purchase’ (processes), initially encoded as a binary relation / association between a buyer and a commodity (in DL and UML), then expanded to include the time of the purchase (requiring a concept in DL and, e.g., an association class in UML). If the connection to the notion of ‘purchase’ is explicated in both OUS (for DL and UML), the relationships between the two versions should be at least easier to grasp in terms of the OUS and might support (semi-)automatic data migration.

² ALC is derived from the phrase ‘attribute logic with complement’.

³ As opposed to “usage-level”.

4.2 Reexamining Ontological Semantics in Conceptual Modeling

Spatial limitations dictate to exclude a reconsideration of the formal approaches [2][13] mentioned in sect. 1.2, but note the assessment of [2] in [11, sect. 5.2].

Respective work in conceptual modeling typically provides mappings of language constructs to ontologies. As far as we can see, those mappings are of informal or semi-formal nature and they are attached to basic modeling constructs / abstract syntax categories, i.e., at the language-level. This corresponds best to our translation mapping σ in an OUS $(L, L_\Omega, \Omega_{base}, \varepsilon, \sigma, \tau)$, although actual formal-logical theories are not established through the given mappings.⁴ The general line of application in those works, besides providing semantics itself, appears to be the evaluation of language constructs and the provision of constraints on their usage. Though valuable, currently this prescriptive approach remains subsidiary for us. Instead, we follow an explanatory approach, originating from a different angle. Our primary aims are reasoning over the translation theories and justifiable translations between different languages based on adequate and provable notions of conceptual equivalence. Especially, we see the need to allow for various translations for the same pair (or set) of languages, because syntactic distinctions are limited. Moreover, if specialized ontologies are available, these should be employed for ontological semantics, even of general purpose languages. The mentioned approaches, e.g. [5][6], differ and seemingly suggest to determine a one-to-one correspondence between language constructs and ontological notions.⁵ Eventually, the intended scope of the present discussion is not restricted to conceptual modeling languages, but aims at covering the ontological contents of “arbitrary” representations.

5 Conclusions and Future Work

The overall aim herein is to unite the formal and conceptual viewpoints on language semantics beneath a common roof, but without reducing one to the other. We advocate the general theses that specific uses of languages should be accompanied by explicating the ontological semantics of the (usage of) language constructs, in addition to (possibly) having a formal or semi-formal semantics assigned to the language already. This presumes that ontological and formal semantics are different. To the best of our knowledge, these positions are not well accepted in formal language communities (if discussed at all), whereas they seem to be acknowledged at least by some researchers in conceptual modeling. After arguing for these views, the approach of ontological usage schemes (OUSes) is outlined and briefly illustrated. It is based on formalization in logical languages and research in top-level ontology, and thus far contributes a translation approach to define precisely ontological semantics for languages.

⁴ Only recently, a translation of a UML representation of Bunge’s ontology into OWL was presented by J. Evermann [4].

⁵ Notably, [6, p. 32] restricts this to only “when using UML for conceptual modeling of a domain”, as distinguished from “software modeling”.

Future work will comprise the definition of further OUSes for languages that are relevant to our ontology development projects. In combination with appropriate notions of (conceptual) equivalence, which we are currently developing, these OUSes should allow us to offer provably conceptually equivalent versions of contents represented in multiple languages. Regarding the rather large collection of works on ontological semantics for UML / in conceptual modeling, it appears worthwhile to elaborate mutual relationships in greater detail. Remembering the distinction of conceptual models and software models in [6], it appears interesting to study the relation and “information flow” between conceptual and software models by means of OUS (or an extension of these). Not at least, this might become one kind of evaluation in practice.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
2. Ciocoiu, M., Nau, D.S.: Ontology-based semantics. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) Proc. of KR 2000, pp. 539–546. Morgan Kaufmann Publishers, San Francisco (2000)
3. Diaz, M. (ed.): Petri Nets: Fundamental Models, Verification and Applications. ISTE, London (2009)
4. Evermann, J.: A UML and OWL description of Bunge’s upper-level ontology model. *Software Syst. Model* 8(2), 235–249 (2009)
5. Evermann, J., Wand, Y.: Ontology based object-oriented domain modelling: Fundamental concepts. *Requir. Eng.* 10(2), 146–160 (2005)
6. Evermann, J., Wand, Y.: Toward formalizing domain modeling semantics in language syntax. *IEEE T. Software Eng.* 31(1), 21–37 (2005)
7. Evermann, J.M.: Using Design Languages for Conceptual Modeling: The UML Case. PhD Thesis, University of British Columbia, Vancouver, Canada (2003)
8. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. CTIT PhD Series No. 05-74, Telematica Instituut, Enschede, The Netherlands (2005)
9. Herre, H.: General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In: Poli, R., Healy, M., Kameas, A. (eds.) *Theory and Applications of Ontology: Computer Applications*. ch. 14, pp. 297–345. Springer, Berlin (2010)
10. Loebe, F.: Abstract vs. social roles: Towards a general theoretical account of roles. *Appl. Ontology* 2(2), 127–158 (2007)
11. Loebe, F., Herre, H.: Formal semantics and ontologies: Towards an ontological account of formal semantics. In: Eschenbach, C., Grüninger, M. (eds.) *Proc. of FOIS 2008*, pp. 49–62. IOS Press, Amsterdam (2008)
12. Noy, N., Rector, A.: Defining N-ary relations on the Semantic Web. W3C Working Group Note, World Wide Web Consortium (W3C) (2006), <http://www.w3.org/TR/swbp-n-aryRelations/>
13. Schorlemmer, M., Kalfoglou, Y.: Institutionalising ontology-based semantic integration. *Appl. Ontology* 3(3), 131–150 (2008)
14. W3C: OWL 2 Web Ontology Language Document Overview. W3C Recommendation, World Wide Web Consortium (W3C) (2009)
15. Wand, Y., Storey, V.C., Weber, R.: An ontological analysis of the relationship construct in conceptual modeling. *ACM T. Database Syst.* 24(4), 494–528 (1999)

Gene Ontology Based Automated Annotation: Why It Isn't Working

Matthijs van der Kroon and Ana M. Levin

Centro de Investigación en Métodos de Producción de Software -PROS-, Universidad
Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Valencia, Spain

Abstract. Genomics has seen a great deal of development since the milestone of the sequencing of the human genome by Craig Venter and Francis Collins in 2000. However, it is broadly accepted now that real challenges are lying ahead in actually understanding the meaning of these raw data. Traditionally this process of assigning meaning to biological crude data is being performed by domain specialists and has been known as annotation. As data chaos becomes larger due to rapid advances in sequencing technologies, the interest for automated annotation has equally increased. Current approaches are often based on the Gene Ontology (GO), but often fail to meet the requirements. Determining why and how they fail will prove crucial in finding methods that perform better, and ultimately might very well deliver the promising feat of turning the Human Genome data chaos into actual knowledge.

1 Introduction

The sequencing of the Human Genome in the year 2000 by Craig Venter and Francis Collins [1] came with tremendous promises. These effects are most probably not yet apparent and science still struggles to process the huge accomplishment into knowledge artifacts. Scientists now broadly agree that reading the sequence of DNA was the relatively easy part of genome analysis; figuring out what the sequence actually means is the real challenge. Following these insights a new scientific line of research was opened as the marriage of informatics and biology: bioinformatics. It is here that bioinformaticians try to combine rigorous yet computationally powerful informatics with the ambiguous and fuzzy biology. And as Venter and Collins efforts start to bear fruits and technology rapidly advances, more and more sequencing experiments are being performed world-wide generating large amounts of data; leading to the following question: how do we manage the data chaos?

Current solutions are often based on ontologies, most notably the Gene Ontology (GO). Literally translated from ancient Greek, "ontos" means "of that which is" and "-logia": science, study. The science of Ontology (uppercase "O") is diverse and dates back to the early Greeks, where it referred to the analytic philosophy of determining what categories of being are fundamental, and in what sense items in those categories can be said to "be". In modern times, an ontology (lowercase "o") is considered many things. Gruber is credited for introducing

the first compact, yet complete description: "[an ontology is a] *specification of a conceptualization*" [2].

2 Gene Expression Data Annotation

As technology advances the amount of data generated by sequencing experiments increases at an equally rapid pace. A DNA microarray is a multiplex technology used in molecular biology. It consists of an arrayed series of thousands of microscopic spots of DNA oligonucleotides, called features, each containing small amounts of a specific DNA sequence, known as probes. These probes can be a short section of a gene or other DNA element. Since an array can contain tens of thousands of probes, a microarray experiment can accomplish many genetic tests in parallel and as such generate abundant amounts of data. Interpreting these data is the main task of the biologist and is often referred to as annotating. It comes as no surprise that research is ongoing to achieve an automated method of annotation. Over the years many automated annotation systems have been proposed, among the most recent ones; [3], [4], [5]. Capturing the meaning of gene expression data by interpretation is an issue that largely depends on, and relates to the theory of knowledge. In some sense, it can be thought of as the quest to identify what in the human genome is thought to "be", which main categories of "being" can be identified and how these interrelate. As these questions correspond largely to the underlying principles of ontologies, it is no coincidence various attempts have been made to combine these two; Tirrell et al. [5] describes RANSUM (Rich Annotation Summarizer) which performs enrichment analysis using any ontology in the National Center for Biomedical Ontologies (NCBO) BioPortal. Jung et al. [6] describes the PoGO (Prediction of Gene Ontology terms) that uses statistical pattern recognition methods to assign Gene Ontology (GO) terms to proteins from fungi.

Khatri et al. [7] provides an overview of current practice on the application of ontologies in the analysis of gene expression data. As a potential pitfall is mentioned that an ontological analysis approach might be biased in favor of certain genes and pathways. Indeed, the number of annotations available is directly proportionate to the number of experiments performed with those genes or pathways. Some biological processes are more intensively studied, thus generating more data. If more data about a specific process are available, this process is more likely to appear in the results. Another potential pitfall is that the results of any analysis are limited by the availability of accurate annotations. It is acknowledged that the existing data are incomplete. Finally, an ontological analysis approach can be criticized because certain genes are more important than others, so the sheer number of genes may not tell the whole story.

3 Gene Ontology Gone Ontology?

Ontologies enable us to build large, maintainable knowledge bases that can codify what we know about specific areas of practice in precise, unambiguous terms.

Adding to this, it allows us to reason over these structured knowledge bases, with the purpose of deducing new knowledge. In this short description we identify two different applications; knowledge management and knowledge deduction. An ontology can be of varying level of rigor, where a lower level, and as such more ambiguous ontology will be fine to deliver the promise of maintaining knowledge bases, but unable to allow for automated reasoning necessary to deduce new knowledge. A higher level of rigor will allow for both accurate knowledge management, and deduction of new knowledge at the cost of increased complexity.

When the Gene Ontology was conceived in 2000 [8], it came with the promise of enabling a conceptual unification of biology by providing a dynamic, controlled vocabulary. Adding to this, it was hoped that the common vocabulary would result "*in the ability to query and retrieve gene and proteins based on their shared biology*", thus deducing new knowledge. Later research has shown that Gene Ontology in reality often lacks rigor to allow for this high level of ontology application. The early discussion started by Smith [9] and Kumar [10], stating that "*It is unclear what kinds of reasoning are permissible on the basis of GOs hierarchies.*" and "*No procedures are offered by which GO can be validated.*". We now proceed to discuss the main points of their work.

3.1 Universals versus Particulars

In metaphysics, a *universal* is a meta-concept. It defines what particular things have in common, namely characteristics or qualities. The idea is that universals can be instantiated by particular things, or *particulars* (also called individuals, exemplars, instances, tokens). For example, the species *E. coli*, with the function to boost insulin production is a universal. Instantiated as the particular *E. coli* bacterium now existing in the Petri dish, its function is to boost insulin production in specific cells in your pancreas. Why is it important to have a distinction between particulars and universals? Consider the following case in which we have modeled a universal, say "*gene*", that corresponds to the biological concept by the same name and has a few properties: it is known for instance that a gene has a promotor and a terminator while being located on a specific chromosome. Once we now establish during a biological experiment that a certain stretch of DNA must be a gene (effectively instantiating the universal to an instance), we are now able to deduce from this knowledge that this particular stretch of DNA must have a promotor, terminator and that it is positioned on a chromosome. Imagine this same case but now we have not modeled the universal, instead storing a list of stretches of DNA (instances) that we consider to be genes. When we try to make the same deduction as earlier in case we find a new stretch of DNA that corresponds to the biological concept of "*gene*", we can not deduce what other properties it has.

3.2 Continuants versus Occurrents

Entities that continue to exist through time are often referred to as *continuants*. In the GO context, organisms, cells and chromosomes are all continuants, even while undergoing changes they do not cease to preserve their identity.

Occurrents on the other hand, are never said to exist in full for a single instant of time. Rather they they unfold during successive phases, like for example a viral infection unfolds itself over time. (Biological) processes usually are characterized by passing through different states: where the nucleus is part of the cell, mitosis is a part of the cellular process.

The continuant/occurrent opposition corresponds in the first place to the distinction between substances (objects, things) and processes. GOs cellular component ontology is in our terms an ontology of substance universals; its molecular function and biological process ontology are ontologies of function and process universals. But functions, too, are from the perspective of philosophical ontology continuants. For if an object has a given function which means a token function for a given interval of time, then this token function is present in full at every instant in this interval. It does not unfold itself in phases in the manner of an occurrent. If, however, the token function gets exercised, then the token process that results does indeed unfold itself in this manner. Each function thus gives rise, when it is exercised, to processes or activities of characteristic types.

3.3 GOs Relations

The GO relation *isa* is referred to as meaning instance of, however in practice it is clearly used in such a way to indicate *is a kind of* or specialization between universals (e.g. "*p53 is a protein*"). Adding to this, sometimes the *isa* relation is used to indicate *part-of*, as in the definition of vacuolar proton-transporting V-type ATPase, V0 domain (GO:0000220), which identifies the concept as *isa* vacuolar part, rather than as a component part thereof.

The part-of relation as defined by GO indicates a transitive relation intended to conceptualize "can be part of, not is always part of". GO uses the part-of relation for representation of parts of both substances and processes, and of functions/activities. The part-of relation is ambiguous in that it does not provide clear means of distinguishing between the following cases:

- A part-of any B
- A part-of some B
- A part-of B, only when a condition holds

However useful as a controlled vocabulary, Gene Ontology all too often fails to deliver on the promise of allowing for deduction of new knowledge due to a lack of necessary conceptual rigor. Egaa Aranguren [11] captures the essence of this issue by stating that "*The computers understanding is determined by the semantics of the language*", thus if the semantics of the language are unclear, so is the computers understanding. It is difficult for humans to adequately reason over situations they dont understand, it is even more so for computers.

3.4 Conceptual Modeling

Conceptual modeling is the practice of creating models for the purpose of either designing an artifact, or achieving a higher understanding of a certain domain.

Often these two overlap in a process referred to as Model Driven Architecture (MDA). In MDA the main objective consists of creating high quality software, made possible by extensive use of conceptual modeling techniques. The idea is to create models of the domain and the to-be created application, after which the software can automatically be generated from these models, in some cases without human interference [12]. Clearly, for this process to succeed the model must be formal and unambiguous, i.e. the semantics of the language must be clear. MDA is most often used in an Information Systems (IS) context, but is it so strange to view the biological mechanism of life as an IS? A very complex and organically based, but an IS nonetheless. Replacing human made, silicon based, chips with organic proteins and processor instructions with DNA transcription and translation to proteins. It is often through making analogies with systems we already comprehend, that we come to understand otherwise difficult to grasp mechanisms. Currently, a very attractive, challenging line of research is the personalized medicine context (have a look for instance at [13], where concrete applications of the IS-based working environment defended in this paper are presented).

The first documented effort to combine the practice of conceptual modeling is [14] which proposes conceptual models for genomic data. Pastor [15][16] then further elaborated on this initial work.

It is important to understand that using either approach; conceptual models (CM) or ontologies is really not that different, as a matter of fact an ontology is always present when creating a conceptual model. It is defined implicitly by the model itself, while the other way around is not always true: not every ontology has a visual representation allowing it to be named a conceptual model.

The universals versus particulars discussion is easily addressed by CMs: that which is reflected in the model must always be a universal, for instance a gene, having the following attributes: HUGO_id, chromosome. While its particulars, for instance a gene that has the HUGO assigned id "BRCA1", and which is located on the human chromosome 17. These particulars are instances of the CMs concepts, and usually exists only at runtime. Runtime being interpreted broadly: both an executing object and a database tuple, stored persistently, are considered runtime.

The GO expressiveness for conceptualizing relations among entities is not rich enough. By being unable to capture certain knowledge adequately, the uncontrolled use of non-standard operators becomes a tempting, ad-hoc, solution. Conceptual modeling offers a rich variety of relations: an aggregation is a weak whole-part relation where the members can exist without the containing or enclosing class, e.g. *Staphylococcus epidermidis* forms part of human skin flora, but can survive without a human host. A composition is also a whole-part relation, but stronger than an aggregation such that the whole does not exist without its parts, e.g. a human individual can not exist without a brain. Further, inheritance allows in an intuitive way to identify hierarchies of concepts; a skin cell is a specified type of cell, thus inherits properties of its super type. The concept of a simple relation merely describes the abstract property of allowing

communication between two entities: for instance the communication between neurons (this would be a reflective association). In case expressiveness is still not rich enough, the Object Constraint Language [17] allows for even finer specification by allowing the application of textual constraints to the models' behavior.

4 Conclusions

Our goal in this work is to start a discussion on the application of Gene Ontology to gene expression annotation, uncovering its flaws and proposing an alternative route. Capturing biological understanding is a huge challenge we face, and the question is not whether we want to do it, but how do we do it. Specifications of conceptualizations are everywhere around us, and thus according to Gruber all these correspond to the concept of an ontology. It is when we try and structure knowledge into a computationally sound structure, where not everything can qualify as an ontology anymore. Rigorous methods are needed, for a semantically sound ontology to emerge. The practice of conceptual modeling has been long around in Information System development, and has proven very suitable to capture domain knowledge in a controlled manner, both syntactically and semantically sound; ultimately even leading to the automated generation and validation of computer software. If we take the liberty of considering the biological system of life as analogous to an Information System it is not easy to miss the grand opportunities such a perspective provides.

References

1. Venter, J., et al.: The Sequence of the Human Genome. *Science* 291(5507), 1304–1351 (2000)
2. Gruber, T.: Principle for the Design of Ontologies Used for Knowledge Sharing. In: Poli, R. (ed.) *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, Dordrecht (1993)
3. Gonzalez-Daz, H., Muo, L., Anadn, A., et al.: MISS-Prot: web server for self/non-self discrimination of protein residue networks in parasites; theory and experiments in Fasciola peptides and Anisakis allergens, *Molecular Biosystems* (2011) [Epub ahead of print]
4. Hsu, C., Chen, C., Liu, B.: WildSpan: mining structured motifs from protein sequences. *Algorithms in Molecular Biology* 6(1), 6 (2011)
5. Tirrell, R., Evani, U., Berman, A., et al.: An ontology-neutral framework for enrichment analysis. In: *American Medical Informatics Association Annual Symposium*, vol. 1(1), pp. 797–801 (2010)
6. Jung, J., Yi, G., Sukno, S., et al.: PoGo: Prediction of Gene Ontology terms for fungal proteins. *BMC Bioinformatics* 11(215) (2010)
7. Khatri, P., Draghici, S.: Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics* 21(18), 3587–3595 (2005)
8. Ashburner, M., Ball, C.A., Blake, J.A.: Gene Ontology: tool for the unification of biology. *Nature Genetics* 25(1), 25–30 (2000)

9. Smith, B., Williams, J., Schulze-Kremer, S.: The Ontology of the Gene Ontology. In: American Medical Informatics Association Annual Symposium Proceedings, vol. 1(1), pp. 609–613 (2003)
10. Kumar, A., Smith, B.: Controlled vocabularies in bioinformatics: a case study in the Gene Ontology. *Drug Discovery Today: BIOSILICO* 2(6), 246–252 (2004)
11. Egea Aranguren, M., Bechhofer, S., Lord, P., et al.: Understanding and using the meaning of statements in a bio-ontology: recasting the Gene Ontology in OWL. *BMC Bioinformatics* 8(57) (2007)
12. Pastor, O., Molina, J.C.: *Model-driven architecture in practice: a software production environment based on conceptual modeling*. Springer, Heidelberg (2007)
13. Collins, F.S.: *The Language of Life: DNA and the Revolution in Personalized Medicine*. Profile Books Ltd. (2010)
14. Paton, N.W., Khan, S.A., Hayes, A., et al.: Conceptual modeling of genomic information. *Bioinformatics* 16(6), 548–557 (2000)
15. Pastor, O., Levin, A.M., Celma, M., et al.: *Model Driven-Based Engineering Applied to the Interpretation of the Human Genome*. In: Kaschek, R., Delcambre, L. (eds.) *The Evolution of Conceptual Modeling*. Springer, Heidelberg (2010)
16. Pastor, O., van der Kroon, M., Levin, A.M., et al.: *A Conceptual Modeling Approach to Improve Human Genome Understanding*. In: Embley, D.W., Thalheim, B. (eds.) *Handbook of Conceptual Modeling*. Springer, Heidelberg (2011)
17. Warmer, J., Kleppe, A.: *Object Constraint Language: Getting Your Models Ready for MDA*, 2nd edn. Addison-Wesley Longman Publishing Co., Boston (2011)

Formal Ontologies, Exemplars, Prototypes

Marcello Frixione and Antonio Lieto

University of Salerno, Italy
{mfrixione, alieto}@unisa.it

Abstract. The problem of concept representation is relevant for knowledge engineering and for ontology-based technologies. However, the notion of concept itself turns out to be highly disputed and problematic in cognitive science. In our opinion, one of the causes of this state of affairs is that the notion of concept is in some sense heterogeneous, and encompasses different cognitive phenomena. This results in a strain between conflicting requirements, such as, for example, compositionality on the one side and the need of representing prototypical information on the other. AI research in some way shows traces of this situation. In this paper we propose an analysis of this state of affairs, and we sketch some proposals for concept representation in formal ontologies which take advantage from suggestions coming from cognitive science and psychological research. In particular we take into account the distinction between prototype and exemplar accounts in explaining prototypical effects.

Keywords: ontologies, knowledge representation, reasoning, knowledge engineering.

1 Introduction

Computational representation of concepts is a central problem for the development of ontologies and for knowledge engineering¹. Concept representation is a multidisciplinary topic of research that involves such different disciplines as Artificial Intelligence, Philosophy, Cognitive Psychology and, more in general, Cognitive Science. However, the notion of concept itself results to be highly disputed and problematic. In our opinion, one of the causes of this state of affairs is that the notion itself of concept is in some sense heterogeneous, and encompasses different cognitive phenomena. This results in a strain between conflicting requirements, such as, for example, compositionality on the one side and the need of representing prototypical

¹ It could be objected that ontologies have to do with the representation *of the world*, and not with the representation *of our concepts*. This is surely true, but, as far as we are (also) interested in our *commonsense* ontologies (i.e., in the representation of the world from the standpoint of our everyday experience, contrasted, for example, with a scientific representation of the world), then, in our opinion, we cannot ignore the problem of how ordinary concepts are structured.

information on the other. This has several consequences for the practice of knowledge engineering and for the technology of formal ontologies.

In this paper we propose an analysis of this situation. The paper is organised as follows. In section 2. we point out some differences between the way concepts are conceived in philosophy and in psychology. In section 3. we argue that AI research in some way shows traces of the contradictions individuated in sect. 2. In particular, the requirement of compositional, logical style semantics conflicts with the need of representing concepts in the terms of typical traits that allow for exceptions. In section 4 we review some attempts to resolve this conflict in the field of knowledge representation, with particular attention to description logics. It is our opinion that a mature methodology to approach knowledge representation and knowledge engineering should take advantage from both the empirical results of cognitive psychology that concern human abilities and from philosophical analyses. In this spirit, in section 5 we individuate some possible suggestions coming from different aspects of cognitive research: the distinction between two different types of reasoning processes, developed within the context of the so-called “dual process” accounts of reasoning; the proposal to keep prototypical effects separate from compositional representation of concepts; the possibility to develop hybrid, prototype and exemplar-based representations of concepts.

2 Compositionality vs. Prototypes

Within the field of cognitive science, the notion of concept is highly disputed and problematic. Artificial intelligence (from now on AI) and, more in general, the computational approach to cognition reflect this state of affairs. Conceptual representation seems to be constrained by conflicting requirements, such as, for example, compositionality on the one side and the need of representing prototypical information on the other.

A first problem (or, better, a first symptom that some problem exists) consists in the fact that the use of the term “concept” in the philosophical tradition is not homogeneous with the use of the same term in empirical psychology [see 1-3]. Briefly, we could say that in cognitive psychology a concept is essentially intended as the mental representations of a category, and the emphasis is on such processes as categorisation, induction and learning. According to philosophers, concepts are above all the components of thoughts. Even if we leave aside the problem of specifying what thoughts exactly are, this requires a more demanding notion of concept. In other words, some phenomena that are classified as “conceptual” by psychologists turn out to be “nonconceptual” for philosophers. There are, thus, mental representations of categories that philosophers would not consider genuine concepts.

The fact that philosophers consider concepts mainly as the components of thoughts brought a great emphasis on *compositionality*, and on related features, such as productivity and systematicity, that are often ignored by psychological treatments of concepts. On the other hand, it is well known that compositionality is at odds with *prototypicality effects*, which are crucial in most psychological characterisations of concepts. Prototypical effects are a well established empirical phenomenon. However, the characterisation of concepts in prototypical terms is difficult to reconcile with the

requirement of compositionality. According to a well known argument by Jerry Fodor [4], prototypes are not compositional (and, since concepts in Fodor's opinion must be compositional, concepts cannot be prototypes). In synthesis, Fodor's argument runs as follows: consider a concept like PET FISH. It results from the composition of the concept PET and of the concept FISH. But the prototype of PET FISH cannot result from the composition of the prototypes of PET and of FISH. For example, a typical PET is furry and warm, a typical FISH is greyish, but a typical PET FISH is not furry and warm neither greyish.

3 Concept Representation in Artificial Intelligence

The situation sketched in the section above is in some sense reflected by the state of the art in AI and, more in general, in the field of computational modelling of cognition. This research area seems often to hesitate between different (and hardly compatible) points of view. In AI the representation of concepts is faced mainly within the field of knowledge representation (KR). Symbolic KR systems (KRs) are formalisms whose structure is, in a wide sense, language-like. This usually involves that KRs are assumed to be compositional.

In a first phase of their development (historically corresponding to the end of the 60s and to the 70s) many KRs oriented to conceptual representations tried to keep into account suggestions coming from psychological research. Examples are early semantic networks and frame systems. Both frames [5] and most semantic networks allowed the possibility to characterise concepts in terms of prototypical information.

However, when AI practitioners tried to provide a stronger formal foundation to concept oriented KRs, it turned out to be difficult to reconcile compositionality and prototypical representations. As a consequence, they often choose to sacrifice the latter. In particular, this is the solution adopted in a class of concept-oriented KRs which had (and still have) wide diffusion within AI, namely the class of formalisms that stem from the KL-ONE system [6-7] and that today are known as *description logics* (DLs) [8]. Networks in this tradition do not admit exceptions to inheritance, and therefore do not allow the representation of prototypical information. Indeed, representations of exceptions can be hardly accommodated with other types of inference defined on these formalisms, concept classification in the first place [9]. Since the representation of prototypical information is not allowed, inferential mechanisms defined on these networks (e.g. inheritance) can be traced back to classical logical inferences. From this point of view, such formalisms can be seen as a revival of the classical theory of concepts, in spite of its empirical inadequacy in dealing with most common-sense concepts.

4 Non-classical Concepts in Computational Ontologies

Of course, within symbolic, logic oriented KR, rigorous approaches exist, that allow to represent exceptions, and that therefore would be, at least in principle, suitable for representing "non-classical" concepts. Examples are fuzzy logics and non-monotonic formalisms. Therefore, the adoption of logic oriented semantics is not necessarily

incompatible with prototypical effects. But such approaches pose various theoretical and practical difficulties, and many unsolved problems remain.

In this section we overview some recent proposal of extending concept-oriented KR, and in particular DLs, in order to represent non-classical concepts.

Recently different methods and techniques have been adopted to represent non-classical concepts within computational ontologies. They are based on extensions of DLs and of standard ontology languages such as OWL. The different proposals that have been advanced can be grouped in three main classes: a) fuzzy approaches, b) probabilistic and Bayesian approaches, c) approaches based on non-monotonic formalisms.

a) Following this direction, for as the integration of *fuzzy logics* in DLs and in ontology oriented formalisms, see for example [10-11], Stoilos et al. [12] propose a fuzzy extension of OWL, f-OWL, able to capture imprecise and vague knowledge, and a fuzzy reasoning engine that lets f-OWL reason about such knowledge. In [13] a fuzzy extension of OWL 2 is proposed, for representing vague information in semantic web languages. However, it is well known [14] that approaches to prototypical effects based on fuzzy logic encounter some difficulty with compositionality.

b) The literature offers also several *probabilistic generalizations* of web ontology languages. Many of these approaches, as pointed out in [15], focus on combining the OWL language with probabilistic formalisms based on Bayesian networks. In particular, in [16] a probabilistic generalization of OWL is proposed, called Bayes-OWL, which is based on standard Bayesian networks. Bayes-OWL provides a set of rules and procedures for the direct translation of an OWL ontology into a Bayesian network. A problem here could be represented by the “translation” from one form of “semantics” (OWL based) to another one.

c) In the field of non-monotonic extensions of DLs, in [17] an extension of ALCF system based on Reiter’s default logic is proposed. The same authors, however, point out both the semantic and computational difficulties of this integration and, for this reason, they propose a restricted semantics for open default theories, in which default rules are only applied to individuals explicitly represented in the knowledge base. In [18] an extension of DLs with circumscription is proposed. One of motivating applications of circumscription is indeed to express prototypical properties with exceptions, and this is done by introducing “abnormality” predicates, whose extension is minimized. A different approach, investigated in [19], is based on the use of the OWL 2 annotation properties (APs) in order to represent vague or prototypical, information. The limit of this approach is that APs are not taken into account by the reasoner, and therefore have no effect on the inferential behaviour of the system [13].

5 Some Suggestions from Cognitive Science

Though the presence of a relevant field of research, there is not, in the scientific community, a common view about the use of non-monotonic and, more in general, non-classical logics in ontologies. For practical applications, systems that are based on classical Tarskian semantics and that do not allow for exceptions (as it is the case of “traditional” DLs), are usually still preferred. Some researchers, as, for example, Pat Hayes [20], argue that the non monotonic logics (and, therefore, the non

monotonic “machine” reasoning for Semantic Web) can be maybe adopted for local uses only or for specific applications because it is “unsafe on the web”. The question about which “logics” must be used in the Semantic Web (or, at least, until which degree, and in which cases, certain logics could be useful) is still open.

The empirical results from cognitive psychology show that most common-sense concepts cannot be characterised in terms of necessary/sufficient conditions. Classical, monotonic DLs seem to capture the compositional aspects of conceptual knowledge, but are inadequate to represent prototypical knowledge. But a “non classical” alternative, a general DL able to represent concepts in prototypical terms does not still emerge.

As a possible way out, we sketch a tentative proposal that is based on some suggestions coming from cognitive science. Some recent trends of psychological research favour the hypothesis that reasoning is not an unitary cognitive phenomenon. At the same time, empirical data on concepts seem to suggest that prototypical effects could stem from different representation mechanisms. In this spirit, we individuate some hints that, in our opinion, could be useful for the development of artificial representation systems, namely: (i) the distinction between two different types of reasoning processes, which has been developed within the context of the so-called “dual process” accounts of reasoning (sect. 5.1 below); (ii) the proposal to keep prototypical effects separate from compositional representation of concepts (sect. 5.2); and (iii) the possibility to develop hybrid, prototype and exemplar-based representations of concepts (sect. 5.3).

5.1 A “Dual Process” Approach

Cognitive research about concepts seems to suggest that concept representation does not constitute an unitary phenomenon from the cognitive point of view. In this perspective, a possible solution should be inspired by the experimental results of empirical psychology, in particular by the so-called dual process theories of reasoning and rationality [21-22]. In such theories, the existence of two different types of cognitive systems is assumed. The systems of the first type (type 1) are phylogenetically older, unconscious, automatic, associative, parallel and fast. The systems of the type 2 are more recent, conscious, sequential and slow, and are based on explicit rule following. In our opinion, there are good *prima facie* reasons to believe that, in human subjects, classification, a monotonic form of reasoning which is defined on semantic networks², and which is typical of DL systems, is a task of the type 2 (it is a difficult, slow, sequential task). On the contrary, exceptions play an important role in processes such as non-monotonic categorisation and inheritance, which are more likely to be tasks of the type 1: they are fast, automatic, usually do not require particular conscious effort, and so on.

Therefore, a reasonable hypothesis is that a concept representation system should include different “modules”: a monotonic module of type 2, involved in classification

² Classification is a (deductive) reasoning process in which superclass/subclass (i.e., ISA) relations are inferred from implicit information encoded in a KB. Categorization is an inferential process through which a specific entity is assigned as an instance to a certain class. In non-monotonic categorization class assignment is a non deductive inferential process, based on typicality.

and in similar “difficult” tasks, and a non-monotonic module involved in the management of exceptions. This last module should be a “weak” non monotonic system, able to perform only some simple forms of non monotonic inferences (mainly related to categorization and to exceptions inheritance). This solution goes in the direction of a “dual” representation of concepts within the ontologies, and the realization of hybrid reasoning systems (monotonic and non monotonic) on semantic network knowledge bases.

5.2 A “Pseudo-Fodorian” Proposal

As seen before (section 2), according to Fodor, concepts cannot be prototypical representations, since concepts must be compositional, and prototypes do not compose³. On the other hand, in virtue of the criticisms to “classical” theory, concepts cannot be definitions. Therefore, Fodor argues that (most) concepts are atoms, i.e., are symbols with no internal structure. Their content is determined by their relation to the world, and not by their internal structure and/or by their relations with other concepts [26-27]. Of course, Fodor acknowledges the existence of prototypical effects. However, he claims that prototypical representations are not part of concepts. Prototypical representations allow to individuate the reference of concepts, but they must not be identified with concepts. Consider for example the concept DOG. Of course, in our minds there is some prototypical representation associated to DOG (e.g., that dogs usually have fur, that they typically bark, and so on). But this representation does not coincide with the concept DOG: DOG is an atomic, unstructured symbol.

We borrow from Fodor the hypothesis that compositional representations and prototypical effects are demanded to different components of the representational architecture. We assume that there is a compositional component of representations, which admits no exceptions and exhibits no prototypical effects, and which can be represented, for example, in the terms of some classical DL knowledge base. In addition, a prototypical representation of categories is responsible for such processes as categorisation, but it does not affect the inferential behaviour of the compositional component.

It must be noted that our present proposal is not entirely “Fodorian”, at least in the following three senses:

- i. We leave aside the problem of the nature of semantic content of conceptual representations. Fodor endorses a causal, informational theory of meaning, according to which the content of concepts is constituted by some nomic mind-world relation.

³ Various attempts to conciliate compositionality and typicality effects have been proposed within the field of psychology ([23-25]). However, when psychologists face the problem of compositionality, they usually take into account a more restricted phenomenon with respect to philosophers. They try to explain how concepts can be combined, in order to form complex conceptual representations. But compositionality is a more general matter: what is needed is an account of how the meaning of *any* complex representation (included *propositional* representations) depends in a systematic way from the meaning of its components and from its syntactic structure. This should allow to account, among other things, for the inferential relationships (typically, of logical consequence) that exist between propositional representations. From this point of view, psychological proposals are much more limited.

We are in no way committed with such an account of semantic content. (In any case, the philosophical problem of the nature of the intentional content of representations is largely irrelevant to our present purposes).

ii. Fodor claims that concepts are compositional, and that prototypical representations, in being not compositional, cannot be concepts. We do not take position on which part of the system we propose must be considered as truly “conceptual”. Rather, in our opinion the notion of concept is spurious from the cognitive point of view. Both the compositional and the prototypical components contribute to the “conceptual behaviour” of the system (i.e., they have some role in those abilities that we usually describe in terms of possession of concepts).

iii. According to Fodor, the majority of concepts are atomic. In particular, he claims that almost all concepts that correspond to lexical entries have no structure. We maintain that many lexical concepts, even though not definable in the terms classical theory, should exhibit some form of structure, and that such structure can be represented, for example, by means of a DL taxonomy.

5.3 Concepts in Cognitive Psychology

Within the field of psychology, different positions and theories on the nature of concepts are available. Usually, they are grouped in three main classes, namely prototype views, exemplar views and theory-theories (see e.g. [23, 28]). All of them are assumed to account for (some aspects of) prototypical effects in conceptualisation, effects which have been firstly individuated by Eleanor Rosch in her seminal works [29].

According to the prototype view, knowledge about categories is stored in terms of prototypes, i.e. in terms of some representation of the “best” instances of the category. For example, the concept CAT should coincide with a representation of a prototypical cat. In the simpler versions of this approach, prototypes are represented as (possibly weighted) lists of features.

According to the exemplar view, a given category is mentally represented as set of specific exemplars explicitly stored within memory: the mental representation of the concept CAT is the set of the representations of (some of) the cats we encountered during our lifetime.

Theory-theories approaches adopt some form of holistic point of view about concepts. According to some versions of the theory-theories, concepts are analogous to theoretical terms in a scientific theory. For example, the concept CAT is individuated by the role it plays in our mental theory of zoology. In other version of the approach, concepts themselves are identified with micro-theories of some sort. For example, the concept CAT should be identified with a mentally represented micro-theory about cats.

These approaches turned out to be not mutually exclusive. Rather, they seem to succeed in explaining different classes of cognitive phenomena, and many researchers hold that all of them are needed to explain psychological data. In this perspective, we propose to integrate some of them in computational representations of concepts. More precisely, we try to combine a prototypical and an exemplar based representation in order to account for category representation and prototypical effects (for a similar, hybrid prototypical and exemplar based proposal, see [30]). We do not take into consideration the theory-theory approach, since it is in some sense more vaguely

defined if compared the other two points of view. As a consequence, its computational treatment seems at present to be less feasible.

5.3.1 Prototypes vs. Exemplars

Prototype and exemplar based approaches present significant differences. Exemplar-based models, for example, assume that the same representations are involved in such different tasks as categorization and identification (i.e., the recognition of a specific instance: “this is the Tower Bridge”) [28]. This contrasts with the prototype models, which assume that these cognitive processes involve different kinds of representations. Furthermore, prototype models capture only on the cognitively central features of a concept, while the exemplar models can represent *in toto* the particular knowledge of a specific entity. For example, the prototypical representation of the concept DOG is based on a subset of cognitively central and relevant traits associated to dogs: e.g. they bark, they wag tail, and so on. Vice versa, the representation of the exemplar Fido could contain such peripheral characteristics as, for example, the information that Fido has got distemper. Another aspect of divergence is represented by the treatment of the categorization process. In order to solve this task, both prototype and exemplar-based models compute the degree of similarity between a target representation on the one side, and the representation of a prototype or of a set of exemplars on the other. The process works as follows: a prototype/exemplar representation of a given category is retrieved from the memory and compared with the target representation. On the basis of this comparison, the process decides whether the object belongs to the category or not. However, despite these similarities, important differences between the two approaches exist. According to the prototype view, the computation of similarity is usually assumed to be linear: if a property is shared by the target and the prototype, then the similarity between the target and the prototype is increased, independently from the fact that other properties are shared or not. According to the exemplar view, the computation of similarity is assumed to be non-linear: a property shared by the target and the exemplar is relevant and gives a contribution to the computation of similarity only if there are also other properties that are shared between the two representations.

The dichotomy between prototype and exemplar based theories of categorization did not emerge only in the field of psychology; rather, it has a counterpart also in such disciplines as machine learning and automatic classification [31]. In particular, within machine learning these two approaches have been adopted for the realization of different classifiers: for example, the Nearest Prototype Classifier (NPC) is based on prototypes, and the Nearest Neighbour Classifier (NNC) is based on exemplars [32]. Recently in this field different proposals of hybrid classifiers have been proposed, in order to overcome the dichotomy between prototypes and exemplars, and to take advantage from both approaches. In this spirit, prototypes and exemplars are not considered as conflicting theories, but as two limit cases of the Instance Based Learning technique used for categorization both by natural and artificial systems. In fact, despite the differences put in evidence by the psychological literature, prototypes and exemplars, taken together, can be considered different aspects of a unique phenomenon: the typicality in categorization processes [23]. In our opinion, such a hybrid approach could be fruitful to face the problem of representing typicality within the field of formal ontologies.

References

1. Dell'Anna, A., Frixione, M.: On the advantage (if any) and disadvantage of the conceptual/nonconceptual distinction for cognitive science. *Minds & Machines* 20, 29–45 (2010)
2. Frixione, M., Lieto, A.: The computational representation of concepts in formal ontologies: Some general considerations. In: *Proc. KEOD 2010, Int. Conf. on Knowledge Engineering and Ontology Development*, Valencia, Spain, October 25–28 (2010)
3. Frixione, M., Lieto, A.: Representing concepts in artificial systems: a clash of requirements. In: *Proc. HCP 2011*, pp. 75–82 (2011)
4. Fodor, J.: The present status of the innateness controversy. *J. Fodor, Representations* (1981)
5. Minsky, M.: A framework for representing knowledge, in Patrick Winston (a cura di), *The Psychology of Computer Vision* (1975); Also in Brachman & Levesque (2005)
6. Brachman, R., Schmolze, J.G.: An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9, 171–216 (1985)
7. Brachman, R., Levesque, H. (eds.): *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos (1985)
8. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: *The Description Logic Handbook: Theory, Implementations and Applications*. Cambridge University Press, Cambridge (2003)
9. Brachman, R.: I lied about the trees. *The AI Magazine* 3(6), 80–95 (1985)
10. Gao, M., Liu, C.: Extending OWL by fuzzy Description Logic. In: *Proc. 17th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 2005)*, pp. 562–567. IEEE Computer Society, Los Alamitos (2005)
11. Calegari, S., Ciucci, D.: Fuzzy Ontology, Fuzzy Description Logics and Fuzzy-OWL. In: Masulli, F., Mitra, S., Pasi, G. (eds.) *WILF 2007. LNCS (LNAI)*, vol. 4578, pp. 118–126. Springer, Heidelberg (2007)
12. Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J.Z., Horrocks, I.: Fuzzy OWL: Uncertainty and the Semantic Web. In: *Proc. Workshop on OWL: Experience and Directions (OWLED 2005)*. *CEUR Workshop Proceedings*, vol. 188 (2005)
13. Bobillo, F., Straccia, U.: An OWL Ontology for Fuzzy OWL 2. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) *ISMIS 2009. LNCS*, vol. 5722, pp. 151–160. Springer, Heidelberg (2009)
14. Osherson, D.N., Smith, E.E.: On the adequacy of prototype theory as a theory of concepts. *Cognition* 11, 237–262 (1981)
15. Lukasiewicz, L., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. *Journal of Web Semantics* 6, 291–308 (2008)
16. Ding, Z., Peng, Y., Pan, R.: BayesOWL: Uncertainty modeling in Semantic Web ontologies. In: Ma, Z. (ed.) *Soft Computing in Ontologies and Semantic Web. Studies in Fuzziness and Soft Computing*, vol. 204. Springer, Heidelberg (2006)
17. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning* 14(1), 149–180 (1995)
18. Bonatti, P.A., Lutz, C., Wolter, F.: Description logics with circumscription. In: *Proc. of KR*, pp. 400–410 (2006)
19. Klinov, P., Parsia, B.: Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008. LNCS*, vol. 5318, pp. 213–228. Springer, Heidelberg (2008)

20. Hayes, P.: Dialogue on rdf-logic. Why must the web be monotonic? (W3C). Link (2001), <http://lists.w3.org/Archives/public/www-rdf-logic/2001Jul/0067.html>
21. Stanovich, K.E., West, R.: Individual Differences in Reasoning: Implications for the Rationality Debate? *The Behavioural and Brain Sciences* 23(5), 645–665 (2000)
22. Evans, J.S.B.T., Frankish, K. (eds.): *In Two Minds: Dual Processes and Beyond*. Oxford UP, New York (2008)
23. Murphy, G.L.: *The Big Book of Concepts*. The MIT Press, Cambridge (2002)
24. Margolis, E., Laurence, S. (eds.): *Concepts: Core Readings*. The MIT Press, Cambridge (1999)
25. Laurence, S., Margolis, E.: Review. *Concepts: where cognitive science went wrong*. *British Journal for the Philosophy of Science* 50(3), 487–491 (1999)
26. Fodor, J.: *Psychosemantics*. The MIT Press/A Bradford Book, Cambridge, MA (1987)
27. Fodor, J.: *Concepts: Where Cognitive Science Went Wrong*. Oxford University Press, Oxford (1998)
28. Machery, E.: *Doing without Concepts*. Oxford University Press, Oxford (2009)
29. Rosch, E.: Principles of categorization. In: Rosch, E., Lloyd, B. (eds.) *Cognition and Categorization*, pp. 27–48. Lawrence Erlbaum, Hillsdale (1978)
30. Gagliardi, F.: A Prototype-Exemplars Hybrid Cognitive Model of “Phenomenon of Typicality” in *Categorization: A Case Study in Biological Classification*. In: *Proc. 30th Annual Conf. of the Cognitive Science Society*, Austin, TX, pp. 1176–1181 (2008)
31. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, 2nd edn., Kaufmann, San Francisco (2005)
32. Gagliardi, F.: The Need of an Interdisciplinary Approach based on Computational Modelling in the Study of Categorization. In: *Proc. of ICCM 2009*, pp. 492–493 (2009)

Unintended Consequences of Class-Based Ontological Commitment

Roman Lukyanenko and Jeffrey Parsons

Faculty of Business Administration, Memorial University, St. John's Canada
{roman.lukyanenko, jeffreyp}@mun.ca

Abstract. Domain ontologies can promote shared understanding of a domain using standardized constructs to represent domain semantics. Building upon this promise, libraries of domain-specific ontologies have been created. However, we argue that adopting any particular domain ontology can in fact hinder domain understanding and reduce the quality of data made available through a shared ontology. This paper examines unintended consequences of class-based ontological commitment and advocates instead an instance-and-property ontological foundation. The proposed approach can better inform the practice of information sharing and can, in participative domains, enable users to contribute higher quality information with fewer constraints.

Keywords: Classification, Domain ontologies, Ontological commitment, Instance-based modeling.

1 Introduction

Specific domain ontologies can offer a number of advantages in integrating and managing data sources. Ontology is a philosophical study aimed at describing what exists in a systematic way [1]. In a particular area of concern, a domain-specific ontology prescribes possible constructs, rules and relationships. Thus, many consider domain-specific ontologies as “surrogates for the semantics of a domain” [2]. Recently, increased attention to domain ontologies is fueled by the need to manage a growing body of heterogeneous data sets, especially on the web [2-4]. Yet, what appears to be a clear advantage of domain-specific ontologies – the explicit representation of domain semantics – may in fact impede domain understanding and distort the originally intended reality. This paper examines unintended consequences of class-based ontological commitment and advocates instead an instance-and-property ontological foundation that avoids the negative effects of class-based ontologies and supports semantic interoperability in a broader sense.

Initially part of philosophy, ontological studies are embraced by the information systems and computer science research based on the pragmatic goal of improving communication between humans and machines. This focus suggests a potential application of ontologies to the Semantic Web, which aims to move beyond syntactic matches and support semantic interoperability [3, 5]. To achieve semantic interoperability, users and machines need to agree on common constructs and rules with

which to understand and reason about domains. In the search for this common blueprint of knowledge, domain-specific ontologies appear to offer an advantage: they contain a structured snapshot of reality usually carefully engineered by domain experts. Thus, ontological commitment implies an agreement between all data producers and users to share common understanding of some domain of interest by adopting a formal set of constructs, relationships and rules [6-7]. For example, when working within the natural history domain, one may use the ETHAN (Evolutionary Trees and Natural History) ontology [8-9]. ETHAN draws upon a combination of Linnaean taxonomy and phylogenetic information to construct a set of classes and properties that form a hierarchical tree of life. This ontology can be used to “discover, integrate, store and analyze” ecological information for a variety of purposes [8].

While ontologies have the potential to facilitate semantically-enhanced information transfer, we argue that the prevailing practice of imposing ontological structure on a domain can impede domain understanding and result in information *loss*. In particular, many ontologies have adopted RDF-compliant (the Resource Description Framework) class-structure [see 4], in which individual information resource instances are recorded as members of *a priori*-defined classes. Classes (which are similar to categories, entity sets, kinds) are typically modelled as sets of attributes/properties that ideally support inferences beyond properties required to establish class membership [see 10]. Thus, ***class-based ontological commitment*** is a requirement to express an ontology in terms of ***classes*** of phenomena and the relationships among them. We further argue that class-based ontological commitment causes loss of potentially valuable properties of the phenomena, thereby impeding domain understanding and semantic interoperability. In contrast, we suggest an instance-based ontological model that does not require classifying individual resources. Instead, each autonomous unit of information (instance) can be described in terms of attributes and the resulting attribute sets can be used to create classes of interest based on ad-hoc utility. This approach is based on the instance-based data model proposed by Parsons and Wand in the context of database design [see 11].

The remainder of the paper is organized as follows. The next section provides a review of relevant ontological literature. Then we discuss theoretical foundation of the new approach and offer a case study to illustrate specific deficiencies of the class-based ontologies. The paper concludes with a discussion on the instance-based ontological foundation and an outlook to the future.

2 Ontological Research in IS

The premise that ontologies can streamline knowledge engineering and improve information transfer is being actively explored in the information systems and computer science literature [12]. Much of the research focuses on domain-specific ontologies, which typically act as content theories and describe classes, properties and relations for a particular domain [see 6, 13]. In that sense, “[o]ntologies are often equated with taxonomic hierarchies of classes” [14, p. 172]. Many class-based libraries of domain-specific ontologies have been created on the Internet (e.g. the Protégé Ontology Library, http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library; DAML, <http://www.daml.org/ontologies/>). The relative success of ontologies in a number of

application areas has led to a further rapid development of domain-specific ontologies, creating an additional challenge of managing ontologies themselves [e.g. 15]. Much of the current research has been dedicated to issues of integration [16], visualization, evaluation [17-18], and improved use of domain-specific ontologies [8, 15, 19]. With the increase in number of ontologies, the question of the impact of ontological commitment to a specific ontology becomes important.

In contrast, general ontologies such as that of Bunge [20] or Sowa [21] have been used to derive fundamental principles of systems analysis and domain modelling [13, 22-23], data quality [24-25], and database design [26].

While ontology development appears to be a prominent research subject, relatively little attention has been paid to the impact of domain-specific ontologies on quality of information in ontology-based information systems. Intuitively, both the choice of ontology and specific constructs within the chosen one should have observable implications. Thus, Orme *et al.* [18] warn: “[o]ntology-specific information may be a limiting factor in the development of future applications unless ontology data are prepared correctly” (p. 50). Burton-Jones *et al.* [17] used semiotic theory (study of signs) to derive a suite of ontological quality metrics, such as lawfulness (correctness of syntax), richness (breadth of syntax), comprehensiveness, accuracy, relevance, semantic and social quality [for a full list see 2]. The comprehensiveness dimension has been related to the *number* of classes and properties described by an ontology: “[l]arger ontologies are more likely to be complete representations of their domains, and provide more knowledge to the agent” [2, p. 92].

Wand and Weber have been concerned with developing fundamental principles of optimal mapping between real-world phenomena and ontology [23, 27-28]. In particular, they have examined the notion of expressive power and identified construct overload, construct redundancy, and construct excess as factors that diminish ontological expressiveness [27]. Here, we attempt to extend the general argument in [23, 27-28] and examine consequences of the class-based ontological models.

2 Consequences of Class-Based Ontological Commitment

Classification is a fundamental activity in which humans engage to manage the perceptual complexity of real-world phenomena. As Raven *et al.* [29] declared: “[m]an is by nature a classifying animal.” As information systems are representations of a perceived real world, information sciences embraced the natural human tendency to organize reality into classes as an important part of domain modelling. Since ontologies are also viewed as semantic surrogates of reality, it appeared natural to model ontological reality in terms of classes. For example, Decker *et al.* [4] define RDF objects “as instances of one or more classes using the type property” (pp. 66-67). Chandrasekaran *et al.* [12] claim identification of “specific classes of objects and relations that exist in some domain” to be the “main contribution” of ontologies (p. 21).

To classify is to abstract from the diversity of the world by focusing on properties that satisfy some utility [10, 30]. Humans need to categorize to manage the large volume of information to which they are exposed on an ongoing basis [see, for example, 31, p.11]. The principle of reducing *cognitive load* suggests that humans

would prefer to use as few categories as possible. From the cognitive load point of view, the “best” categories are the most inclusive ones [32-33]. Yet, little useful information can be inferred by classifying objects into general categories such as *object* or *living being*, except that that they indicate some existence. As a result, humans need to consider categories at finer levels of specificity. As categories become conceptually smaller and include fewer individuals, they gain *inferential power*. This can be explained using probability theory [e.g. 33]. Since there are fewer *birds* than there are *objects*, a certain *bird* property has a greater chance of occurring in a smaller category (e.g. birds) than in the more inclusive category (e.g. objects). Once an individual is identified as an instance of a category, it is easier (with greater probability) to infer additional (unobserved) properties for a smaller category [30]. Yet, since there are more “smaller” categories (e.g. duck) than inclusive ones (e.g. object), maintaining them requires additional resources.

In the case of both humans and machines, classification means that certain properties that are not of immediate interest are ignored. Herein is a fundamental difference between *human* and *computerized* representations of reality. When humans classify, they *focus* on relevant features *but remain aware of other ones* (e.g., they can recall other features of an instance that are not associated with a particular category of interest). Recent research shows that humans can hold a “massive” amount of details for a long time without instructions to maintain any specific information [34]. This information can be stored subconsciously and invoked in response to certain triggers. While humans use a certain classification of the world, they remain aware of other potential alternatives or accept them with new experiences. For example, in William Blake’s poem *The Tyger* a vivid picture of a dangerous predator is depicted. Survival instincts cause humans to categorize large wild animals as *dangerous*. Yet, this does not preclude us from considering *Aunt Jennifer’s Tigers* by Adrienne Rich as symbols of freedom and chivalry and wishing we could be prancing alongside them. Thus, in the domain of poetry, the same *class* of tigers can have different properties of interest in the mind of a poetry reader. As humans engage with their environment, they continuously revise and adjust their class definitions. In contrast, strict adherence to a predefined class structure means only individuals possessing a certain set of properties will be permitted to be classified as members of a particular class. Thus, if an ontology that defines *tiger* as *dangerous* needs to integrate an instance that is characterized as *meeek*, it may either reject classification, or ignore the *meeek* property and assume the *dangerous* one in the process of data integration. It can also lead to a lower probability of class membership in those approaches that treat class boundaries as fuzzy.

Proposition. Domain understanding is necessarily reduced every time a class is used to conceptually represent instances.

The assumption of the class-based approach to data modeling is that individual instances are always members of some (usually one) *a priori* defined class. Yet, since many classes can be used to represent the same phenomena, it is unclear which class is better. Choosing the “wrong” one means that the information stored will be deficient with respect to perceived reality. Parsons and Wand proposed cognitive guidelines for choosing classes that could be used for “reasoning about classification, and their violation indicates something is ‘lost’ from a cognitive point of view”

[30, p. 69; emphasis added]. Extending this idea further, we claim that using classes to represent instances will *always* fail to fully capture reality, no matter how “good” the chosen class is. Simply, no class is good enough to fully represent an individual and whenever an attempt is made to do so, properties not defined by a class are neglected or lost.

Proposition: An instance can never be fully represented by a class.

Any object has a potentially large number of features and no one class can capture them all. The same object can belong to many classes, which means that individual objects exist independent of any given classification. The relationship between individuals and classes is well depicted in Bunge’s ontology [20, 23]. According to Bunge the world is made of distinct *things* and *things* are characterized by their *properties*. Humans perceive properties indirectly as attributes and classes can be formed by grouping properties together.

Each instance is different from all others in some ways. As Bunge puts it, “there are no two identical entities” [20]. Even the atoms and particles of the same element are in some sense *different* because no two things “can occupy the same state at the same time” [20]. In fact, on the basis of individual differences it is possible to construct as many new classes as there are differences.

Typically, however, classes are formed based on the *commonality* of instances, not their *differences*. Nevertheless, this does not preclude humans from considering, if necessary, properties that are not part of the classification schema. For example, when professors think about own *students* each student retains a plethora of individual features. Some students may require more attention than others. The distribution of attention for each student may also change over time. In contrast, a university domain ontology typically defines a *Student* class using the same set of properties. Thus, while humans are capable of both reasoning about similarity and difference, class-based ontologies emphasize commonalities only. An important element of reality – individual differences – is consistently neglected. Rigid classification-based ontologies routinely miss potentially valuable instance-specific information.

Finally, human categorization is a dynamic process, during the course of which class schemas may change. As shown from the discussion of the poetry domain, humans can uncover new features of familiar classes with additional experience [35]. Yet, as emphasized by Uschold and Gruninger semantic interoperability demands standardization [6]. This requires ontologies to have persistent and stable schemas. Thus, Uschold and Gruninger introduce the concept of reusability, defined as “the formal encoding of important entities, attributes, processes and their inter-relationships in the domain of interest” [6, p.3]. Thus, storing instances in terms of predefined classes makes it difficult to dynamically include new properties into class definitions.

The implications of class-based ontologies are illustrated next using a simple case study.

2.1 Case Study: Travel Ontology

To illustrate the implications of class-based ontological commitment, we use an example of fictitious travel ontology (see Fig. 1). A number of travel ontologies have been explored in information systems and computer science literature and the one presented uses typical travel ontology constructs [36-38].

For simplicity, the proposed ontology consists of four classes: traveler, agent, trip and supervisor, each with a set of properties. For example, the class definition of *traveler* mean that any individual members of that class will be expected to have corresponding properties *customer id*, *name*, *passport no*, *nationality*, *date of birth*, *email*, *password*. It is possible that some values of the properties will be missing, but in order to qualify as a *traveler*, a certain number of values known must be present. Similarly, the class *travel agent* contains those properties that best describe agency employees who assist customers in booking and management of trips (*agent id*, *name*, *supervisor id*, *date hired*, *email*).

Each class has its own “unique” relations to other classes. For example, *traveler* can *request information about* or *take a trip*. At the same time, an *agent* assists with *trip booking* and maintains *trips*. Travelers may not have direct relationships with agency’s supervisors, while each agent reports to a particular supervisor. In each case a class defines the set of intuitive and “lawful” relations with other constructs.

The presented ontology can be used to guide distributed application development and automatically integrate data from multiple local agencies into a global one.

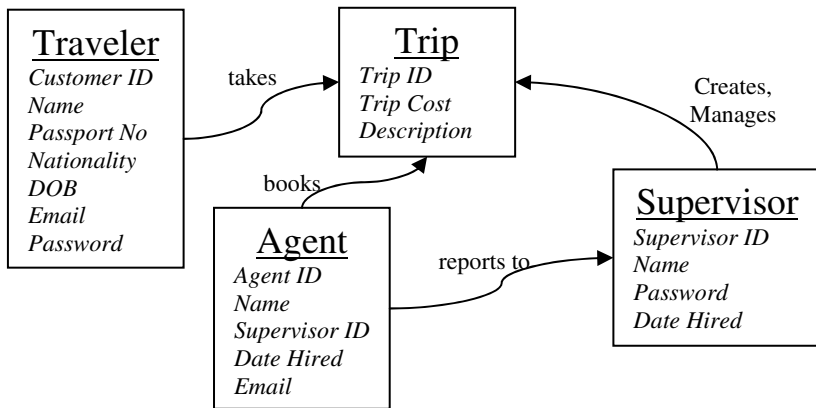


Fig. 1. Hypothetical travel agency ontology

Close examination of the travel ontology presented in Fig. 1 raises a number of issues related to information transfer and domain understanding. In each case where a class is defined, some information is lost. Consider the class *traveler*: the schema that defines the class is focused on the most pertinent characteristics of typical tourists. By the cognitive principles discussed above, it is unreasonable to load *traveler* class with attributes that go beyond its “natural” scope. In other words properties such as *hobbies*, *pet preferences*, *education level*, and *favorite store* may not describe traveler class well. Most data sources dealing with tourism may not have values for such properties. In fact, restricting the scope seems reasonable in order to maintain the domain-specific focus.

Yet each time the scope is restricted, opportunities to collect potentially valuable business data are missed. Some of the additional properties can be valuable to identify

business opportunities and offer a personalized customer experience. For example, while choosing the trips, we may consider that one of the travelers is *afraid of heights* and exclude CN Tower (Toronto, Canada) from the itinerary.

One can argue that some of the individuality can be accommodated using general description in addition to the record, but general descriptions (e.g. comments fields) are difficult to analyze and compare, especially in large datasets. This means that the itinerary of the person who is *afraid of heights* cannot be automatically suggested for a similar traveler at a later time. Thus, in addition to properties, valuable semantic relations between individual records are potentially lost in class-based ontological models.

No individual customer can be described using only the attributes of the *traveler* class. In fact, it is entirely possible for an agent to use corporate discount and book a trip with the agency he/she works for. This will make the same individual a traveler and an agent at the same time. The information about the same individual will have to be maintained in two or more places. This is known as proliferation of operations and has been described by Parsons and Wand in the context of databases [26]. Proliferation of operations is inherent in class-based models of reality because individuals can be described a potentially large number of classes. Yet, this is not merely an issue of multiple classification support, but rather a broader difficulty of accommodating individual differences using a deductive approach of *a priori* classification. Any new person walking into a travel agency will differ from the previous one in some way. Thus, using some predefined filter innately restricts ability to reason about unique properties of an instance.

As seen from this example, a classification-based ontology necessarily leads to loss of valuable information and thereby undermines domain understanding and semantic interoperability.

3 Discussion and Conclusion

In this paper we offered a theoretical discussion on the implications of class-based domain ontologies. We motivated the research by the need to achieve better domain understanding, while maintaining semantic interoperability. From the discussion above, it is clear that information loss is inherent in any class-based ontology. At the same time, an argument can be made that the benefits offered by class-based ontologies may compensate for the data deficiencies. To address this argument, we advocate an instance-and-property ontological foundation, which avoids the negative effects of class-based ontologies while supporting semantic interoperability in a broader sense.

The instance-and-property ontological model draws upon Bunge's ontology and cognitive theories. Instead of creating a predefined set of classes, the focus of domain-specific ontologies should be concerned with capturing individual instances of information and any properties that may describe them. For example, in the travel ontology a particular person is a unique instance. This person can be both a travel agent and a passenger. The instance can also exist without being classified. For example, an anonymous visitor to an online travel agency will not be "qualified" as a customer, but a web log of a site visit indicates an existence of something. We may

already know something about this visitor, which may be of potential business use: we may be interested in his/her IP address, click stream patterns [39], viewing time [40], or comments he/she left on a website. Once several attributes are recorded, the system can match them with pre-existing sets of identifying attributes for a phenomena (such as a class of interest), and either infer a class or seek additional attributes that could also be automatically deduced from those previously supplied. The final attribute set can potentially match to a class (e.g. customer), or integrate instances without classifying them. Doing so avoids inherent data quality deficiencies of the class-based models. By shifting the focus from classification to identification of instances and properties, fuller and semantically richer information about domains can be collected without imposing a particular view and biasing the results.

The instance-based approach can also address the problem of representing temporary changing information. For example, an individual's properties may change overtime. Class membership, thus, becomes more dynamic and evolves with the changing properties.

Instance-and-property ontologies can accommodate the growing semantic diversity of the web. Discretionary data input is growing and increasingly large numbers of websites generate content from direct user input. This is the premise behind the practice of crowdsourcing, the volunteer participation of regular users in purpose-driven projects online [41]. One type of crowdsourcing is citizen science. Online citizen science projects, such as eBird (ebird.org) or iSpot (ispot.org.uk), attempt to capture valuable insights of regular people to be used in academic research. It is clearly difficult to *a priori* anticipate what kind of information non-experts can provide, and creating unnecessary constraints can undermine potential gains from such projects. Moreover, amateur observers are often unable to provide information in compliance with the constructs and relations of a given ontology, especially if it is based on scientific taxonomy (e.g. ETHAN). Being able to give voice to every citizen of the Internet and easily manage that data is an emerging frontier of the web of the future.

References

1. Lacey, A.R.: A dictionary of philosophy. Routledge, New York (1996)
2. Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering* 55, 84–102 (2005)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American Magazine*, 28–37 (2001)
4. Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I.: The Semantic Web: the roles of XML and RDF. *IEEE Internet Computing* 4, 63–73 (2000)
5. Guizzardi, G.: Theoretical foundations and engineering tools for building ontologies as reference conceptual models. In: *Semantic Web*, pp. 3–10 (2010)
6. Uschold, M., Gruninger, M.: Ontologies: principles, methods, and applications. *Knowledge Engineering Review* 11, 93–155 (1996)
7. Gruber, T.R.: A translation approach to portable ontology specifications. *Knowl. Acquis.* 5, 199–220 (1993)

8. Parafiyuk, A., Parr, C., Sachs, J., Finin, T.: Proceedings of the Workshop on Semantic e-Science. In: AAAI 2007 (2007)
9. Parr, C., Sachs, J., Parafiyuk, A., Wang, T., Espinosa, R., Finin, T.: ETHAN: the Evolutionary Trees and Natural History Ontology. Computer Science and Electrical Engineering. University of Maryland, Baltimore County (2006)
10. Parsons, J., Wand, Y.: Using cognitive principles to guide classification in information systems modeling. *Mis Quart.* 32, 839–868 (2008)
11. Parsons, J., Wand, Y.: Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems* 25, 228–268 (2000)
12. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems* 14, 20–26 (1999)
13. Evermann, J., Wand, Y.: Ontology based object-oriented domain modelling: fundamental concepts. *Requir. Eng.* 10, 146–160 (2005)
14. Hansen, P.K., Mabogunje, A., Eris, O., Leifer, L.: The product development process ontology creating a learning research community. In: Culley, S., WDK, W.D.-K. (ed.) *Design Management: Process and Information Issues*. Professional Engineering for The Institution of Mechanical Engineers, pp. 171–186 (2001)
15. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotech.* 25, 1251–1255 (2007)
16. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. *Knowl. Eng. Rev.* 18, 1–31 (2003)
17. Burton-Jones, A., Wand, Y., Weber, R.: Guidelines for Empirical Evaluations of Conceptual Modeling Grammars. *J. Assoc. Inf. Syst.* 10, 495–532 (2009)
18. Orme, A.M., Haining, Y., Eitzkorn, L.H.: Indicating ontology data quality, stability, and completeness throughout ontology evolution. *Journal of Software Maintenance & Evolution: Research & Practice* 19, 49–75 (2007)
19. Beck, T., Morgan, H., Blake, A., Wells, S., Hancock, J.M., Mallon, A.-M.: Practical application of ontologies to annotate and analyse large scale raw mouse phenotype data. *BMC Bioinformatics* 10, 1–9 (2009)
20. Bunge, M.A.: *The furniture of the world*. Reidel, Dordrecht (1977)
21. Sowa, J.F.: *Knowledge representation: logical, philosophical, and computational foundations*. Brooks/Cole, Pacific Grove (2000)
22. Parsons, J., Wand, Y.: Using objects for systems analysis. *Commun. ACM* 40, 104–110 (1997)
23. Wand, Y., Weber, R.: An Ontological Model of an Information-System. *IEEE T. Software Eng.* 16, 1282–1292 (1990)
24. Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. *Commun. ACM* 39, 86–95 (1996)
25. Wand, Y., Monarchi, D.E., Parsons, J., Woo, C.C.: Theoretical Foundations for Conceptual Modeling in Information-Systems Development. *Decis. Support Syst.* 15, 285–304 (1995)
26. Parsons, J., Wand, Y.: Emancipating Instances from the Tyranny of Classes in Information Modeling. *ACM Transactions on Database Systems* 25, 228 (2000)
27. Wand, Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. *Inform. Syst. J.* 3, 217–237 (1993)

28. Wand, Y., Weber, R.: An Ontological Evaluation of Systems-Analysis and Design Methods. *Information System Concepts: An in-Depth Analysis* 357, 79–107 (1989)
29. Raven, P.H., Berlin, B., Breedlove, D.E.: The Origins of Taxonomy. *Science* 174, 1210–1213 (1971)
30. Parsons, J., Wand, Y.: Choosing classes in conceptual modeling. *Commun. ACM* 40, 63–69 (1997)
31. Markman, E.M.: *Categorization and Naming in Children: Problems of Induction*. MIT Press, Cambridge (1991)
32. Murphy, G.L.: Cue validity and levels of categorization. *Psychological Bulletin* 91, 174–177 (1982)
33. Corter, J., Gluck, M.: Explaining basic categories: Feature predictability and information. *Psychological Bulletin* 111, 291–303 (1992)
34. Brady, T.F., Konkle, T., Alvarez, G.A., Oliva, A.: Visual long-term memory has a massive storage capacity for object details. *PNAS Proceedings of the National Academy of Sciences of the United States of America* 105, 14325–14329 (2008)
35. Anderson, J.R.: The Adaptive Nature of Human Categorization. *Psychol. Rev.* 98, 409–429 (1991)
36. Vukmirovic, M., Szymczak, M., Ganzha, M., Paprzycki, M.: Utilizing ontologies in an agent-based airline ticket auctioning system. In: *28th International Conference on Information Technology Interfaces, Croatia*, pp. 385–390 (2006)
37. Chang, C., Miyong, C., Eui-young, K., Pankoo, K.: Travel Ontology for Recommendation System based on Semantic Web. In: *The 8th International Conference on Advanced Communication Technology, ICACT 2006*, vol. 1, pp. 624–627 (2006)
38. Gong, H., Guo, J., Yu, Z., Zhang, Y., Xue, Z.: Research on the Building and Reasoning of Travel Ontology. In: *Proceedings of the 2008 International Symposium on Intelligent Information Technology Application Workshops*, pp. 94–97. IEEE Computer Society, Los Alamitos (2008)
39. Park, J., Chung, H.: Consumers' travel website transferring behaviour: analysis using clickstream data-time, frequency, and spending. *Service Industries Journal* 29, 1451–1463 (2009)
40. Parsons, J., Ralph, P., Gallagher, K.: Using Viewing Time to Infer User Preference in Recommender Systems. In: *Proceedings of the AAAI Workshop on Semantic Web Personalization held in conjunction with the 9th National Conference on Artificial Intelligence (AAAI 2004)*, pp. 52–63 (2004)
41. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the World-Wide Web. *Commun. ACM* 54, 86–96 (2011)

Preface to SeCoGIS 2011

This volume contains the papers presented at SeCoGIS 2011, the Fifth International Workshop on Semantic and Conceptual Issues in GIS, held the 1st of November in Brussels, Belgium.

Current information technologies have increased the production, collection, and diffusion of geographical and temporal data, thus favoring the design and development of geographic information systems (GIS) and more generally speaking spatio-temporal information systems (STIS). Nowadays, GISs are emerging as a common information infrastructure, which penetrate into more and more aspects of our society. This has given rise to new methodological and data engineering challenges in order to accommodate new users' requirements for new applications. Conceptual and semantic modeling are ideal candidates to contribute to the development of the next generation of GIS solutions. They allow eliciting and capturing user requirements as well as the semantics of a wide domain of applications.

The SeCoGIS workshop brings together researchers, developers, users, and practitioners carrying out research and development in geographic information systems. The aim is to stimulate discussions on the integration of conceptual modeling and semantics into current geographic information systems, and how this will benefit the end users. The workshop provides a forum for original research contributions and practical experiences of conceptual modeling and semantic web technologies for GIS, fostering interdisciplinary discussions in all aspects of these two fields, and will highlight future trends in this area. The workshop is organized in a way to highly stimulate interaction amongst the participants.

This edition of the workshop attracted papers from 15 different countries distributed all over the world: Brazil, Argentina, Chile, USA, Canada, France, Italy, Spain, Germany, Belgium, The Netherlands, Austria, Macedonia FYROM, Morocco, and New-Zealand. We received 19 papers from which the Program Committee selected 6 papers, making an acceptance rate of 32 percent. The accepted papers cover a wide range of issues, in particular consistency and integration, refined spatial relationships and conceptual model transformation rules.

We hope that you find the program and presentations beneficial and enjoyable and that during the workshop you had many opportunities to meet colleagues and practitioners. We would like to express our gratitude to the program committee members and the external referees for their hard work in reviewing papers, the authors for submitting their papers, and the ER 2011 organizing committee for all their support.

July, 2011

Esteban Zimányi
Roland Billen
Pierre Hallot

Referring Expressions in Location Based Services: The Case of the ‘Opposite’ Relation

Phil Bartie¹, Femke Reitsma¹, Eliseo Clementini², and Simon Kingham¹

¹ Geography Department, University of Canterbury, Christchurch, New Zealand

² Department of Electrical and Information Engineering, University of L'Aquila, L'Aquila, Italy

Abstract. Mobile devices and location based services enable digital and real worlds to be integrated within our daily lives. The handling of natural language dialogue raises several research challenges, including the ability to direct the user's attention to a particular feature in the field of view through the use of suitable descriptions. To mimic natural language these referring expressions should use attributes which include factors of the building's appearance, and descriptions of its location with reference to the observer or other known buildings in view. This research focuses on one particular positional case used in describing features in a field of view, that of the “opposite” spatial relation, and discusses how this referring expression may be generated by modelling the view from the observer's location to the surrounding features.

Keywords: location based services, visibility modelling, spatial relations.

1 Introduction

Increasingly, digital and real worlds are becoming integrated within our daily lives, with mobile devices and location based services being among the tools that enable this to happen. One of the drawbacks has been that graphical interfaces distract the user from their environment, and alternative interaction experiences are being researched. Augmented Reality [1, 2] is one such innovation whereby digital information is superimposed onto real world views. Speech interfaces are another solution, whereby information may be retrieved using voice commands and speech prompts [3, 4]. The work presented here discusses how speech interfaces may reference items in the current view using natural language terms, with particular focus on the use of the spatial preposition “opposite” as in “we're the house *opposite* the bakery”.

People use language to describe and share experiences about space and the objects which occupy it [5]. These object descriptions are known in natural language research as “referring expressions” and are used, for example, to draw someone's attention to a particular building in a cityscape [6]. Typically, the descriptions include a number of physical attributes relating to the feature, such as its position relative to the observer or other surrounding objects, so that the listener may identify the intended target. The research area has particular relevance for the future of speech based interfaces for Location Based Services (LBS), in both the generation of phrases to direct the user's

attention, and in the parsing of phrases to determine which object is being interrogated.

This paper discusses the positional case of “opposite”, and how a LBS may make use of the term when forming referring expressions. To be able to define the case whereby two buildings are considered opposite, the visibility from the observer to each building needs to be determined, as does the view of a common reference point between the two target buildings, for example a road. A model is proposed which makes use of an urban Digital Surface Model (DSM) to estimate visibility, along with a number of examples of how the term may be calculated and used when generating spatial descriptions. Consideration is also given to the relations with surrounding entities to find the most suitable reference candidates, and minimise the risk of confusion in cluttered environments. Two models are presented which may be used to determine opposite entities with respect to a linear feature and a regional feature. The research shows that linear feature cases may be more easily modelled than regional cases.

2 Background

The range of LBS applications has diversified from basic navigational support into areas of social networking and virtual city guides [4, 7]. User interfaces have tended to have a graphical focus, but mobile speech recognition tools are improving to the point where devices in the near future may incorporate a speech based mode allowing users to operate in a hands-free and eyes-free way discretely, without the need to re-focus attention from their environment [8]. Earlier research has shown that users are able to carry out multiple tasks more easily if they are using a number of sensory modalities [9].

System usability is very important for the success of LBS applications, and great efforts are made to reduce the seam between the application and the user by closely modelling the user’s viewpoint [10]. The ultimate goal would be for an LBS to pass the ‘spatial Turing test’ [11], whereby its instructions are indistinguishable from those generated by a human. Steps towards this goal require that the LBS filter and translate digital information into appropriate forms which match the user’s frame of reference [12, 13].

2.1 Positional Information

The position of any feature in the urban landscape can be described relative to the observer, or a secondary reference object. Relations are rarely described in metric space (e.g. 123.7m at 54 degrees) but instead usually refer to topological space [14, 15] or projective space [16]. For example a paddling pool may be described as being “inside the park” using a topological relations, or “in front of the swings” using a projective relations. Equally a house may be described as “on your left” by referencing the view experienced by an observer at a given location and orientation.

The topological relations between static features are permanent, which in urban areas may include containment within a region (e.g. in a park), topographic feature (e.g. on a hill, a slope, or in a valley), or adjacency to a linear feature (e.g. road, river, rail). In contrast, projective relations are ternary comparisons between the primary

object, a reference object, and the observer [17]. This means that they are dynamic as the user’s viewpoint is considered in each relations, therefore an ability to model which Features of Interest (FOI) are in view is required to ensure only visible items are referenced.

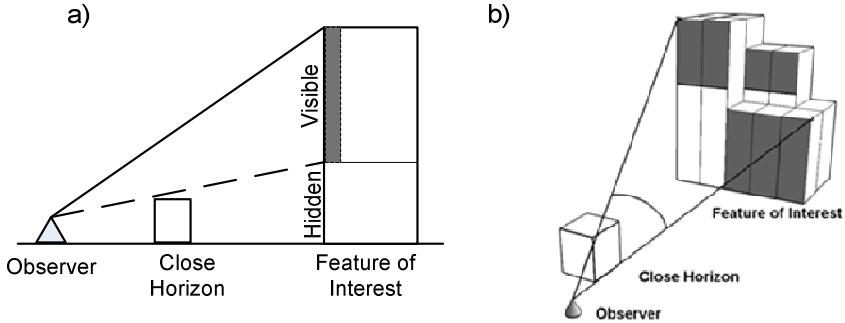


Fig. 1. Visual Exposure in Urban Environments

2.2 Visibility Modelling

Most Geographic Information Systems (GIS) offer the functionality to carry out visibility modelling [18], with a catalogue of research including siting radio masts [19], locating the most scenic or most hidden routes [20], landscape planning [21], as a weapon surrogate in military exercises [22], and in examining spatial openness in built environments [23].

Studies in the urban landscape have tended to be based on isovists [24], using in particular Benedikt’s [25] interpretation and definitions. Essentially isovists describe the space which is visible from a vantage point considering the form of the built environment through the use of architectural plans which denote the building footprint and position. However this model ignores building height, the topography of the land surface, and the continuation of the lines of sight beyond the first intersection with a building footprint. Therefore isovists depict lines which when traversed from the vantage point offer a continuous view of the target, and disregard more distant features.

Recently, 3D isovists [26] and visual exposure models [27, 28] using DSMs built from LiDAR sources have been introduced for urban visibility modelling. These DSMs include building and topographical form and may be used to determine how much of a feature can be viewed from the surrounding space, enabling the creation of surfaces to show in which direction an observer would need to move to view the target more, or less, clearly. These techniques can be used to find visual corridors, or visual ridges, and form a useful basis for considering feature visibility in the context of LBS. The urban visual exposure model calculates the vertical extents visible for each building cell of the DSM, by calculating the lowest visible point on the façade from the intersection of foreground objects, as shown in Fig. 1.

From this the visible façade area, and the percentage of a feature on the skyline may be deduced, along with other metrics. Once the model is able to determine which

features are visible, it is possible to then relate these to construct the positional part of a referring expression. To translate the positional information into the user’s frame of reference requires an egocentric projective spatial model as discussed in the next section.

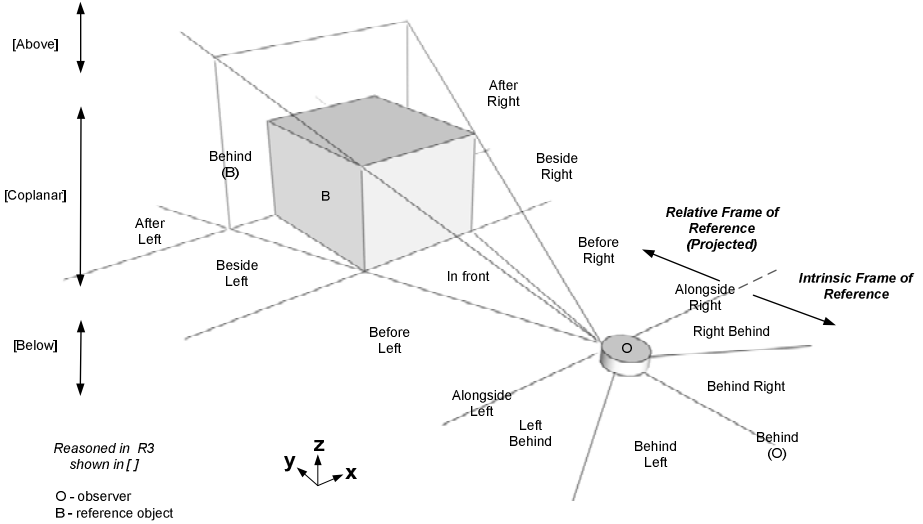


Fig. 2. A combined model of space using relative and intrinsic frames of reference

2.3 Egocentric Spatial Model

Natural language terminology relates space according to the observer’s frame of reference, or that of other features in view. This involves turning the relations from metric space into projective space using terms such as ‘left of’, ‘right of’, ‘before’, ‘after’ and ‘between’ as presented by the 5-intersection model [16]. In addition a quaternary projective relation model is required to include terms for ‘above’, ‘below’, and ‘coplanar’ [29].

Fig. 2 shows a combined model which uses these projective models for the space in front of an observer, and a simplified intrinsic frame of reference for items behind the user [30]. For two items to be considered opposite one another reference is inferred to a common central space, or object. For example, “the library is opposite the park” determines that from some viewing point both the library and park are visible and facing towards each other from a common central region. The following section explores these concepts of the inter-relationship between visibility and projective relations with respect to the “opposite” relation.

3 The ‘Opposite’ Relation

The spatial relation “opposite” is defined in Merriam-Webster’s dictionary as, “set over against something that is at the other end or side of an intervening line or

space” [31]. Hence the interrelated visibility of three related physical entities are required, such as the library, park and street of the previous example. For this reason visual exposure modelling is required to report which objects may be used in the relation. Two cases are examined here, firstly where the entity is represented as a one-dimensional feature, such as a road or river, and secondly where it is represented as a two-dimensional region, such as a park.

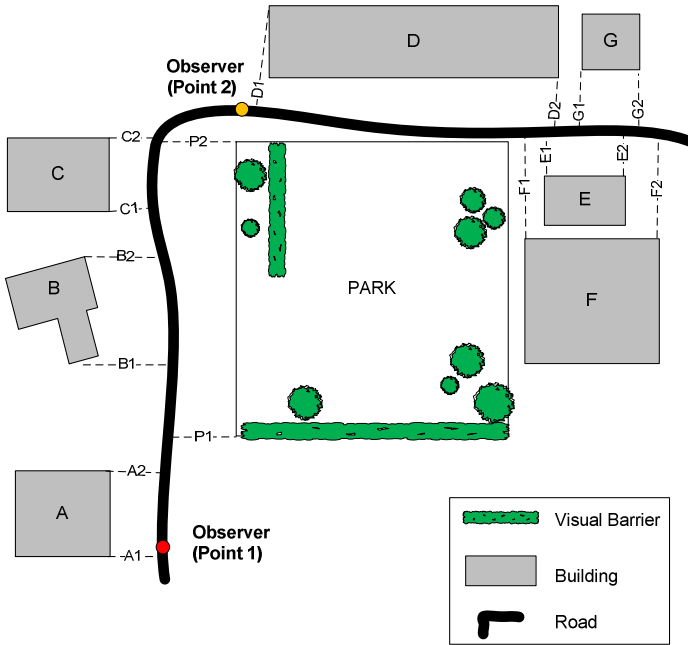


Fig. 3. The ‘Opposite’ Case for a Common Linear Feature

3.1 One-Dimensional Common Feature

The relation conveyed in the phrase “the library is opposite the park”, can be broken down into “the library is *left* of the road” and “the park is *right* of the road” from a viewpoint on the road. However this does not signify that the two objects are “opposite” each other unless both are perceived to occur at similar positions along the road. Consider Fig. 3 which shows a situation where a number of buildings surround a park, and the observer is located on a road at Point 1. From this viewpoint the observer is able to see Buildings A,B,C, and the upper part of D above hedges, but not Buildings E, F and G which are out of view and therefore not included in any referring expressions. When the observer faces Building C, it is valid to report that both Buildings C and B are ‘opposite’ the park. Here the term “with reference to the road” is left out but inferred, and the phrase is equivalent to “Building C is on the other side of the road from the park”. However, it would not be appropriate to define Building A as “opposite the park”, as the two features do not share a similar location along the linear road feature, yet the phrase “opposite side of the road” is still true.

The segments of the road labelled A1-A2, B1-B2 indicate the start and end of the entity along the road’s length, computed at the first and last intersections of a line perpendicular to the direction of travel with the entity. To satisfy the “opposite” condition the following must apply:

- the observer must be able to view both entities from a single point (e.g., C and Park);
- the entities must occur at overlapping sections of the linear entity (e.g., C1-C2 and P1-P2);
- the observer must also be able to view the common linear entity in the overlap region (e.g., road);
- the entities must occupy *beside left/right* space when viewed from the overlapping region.

Table 1. Calculating the Opposite relation for Entities and the Park from Observer Point 1

Entity	Visible	Overlap	View Common Overlap	Beside Left/Right	Result
A	True	False	True	True	False
B	True	True	True	True	True
C	True	True	True	True	True
D	True	True	False	True	False
E	False	False	False	False	False
F	False	False	False	False	False
G	False	False	False	True	False

Following these rules Table 1 may be generated, indicating that B and C are opposite the park when viewed from Point 1. Although part of D is visible above hedges, the section of roadway between the building and the park is out of view behind bushes, rendering the use of “opposite” as less appropriate to assist the user’s visual search from the current location. However, the term could be used if instructions considered the case as the user approaches, such as “when you get to *Point 2* you’ll see D *opposite* the park”.

When multiple features satisfy the ‘opposite’ relation further consideration is necessary to establish which would form the most suitable candidate. So from Point 2 buildings E, F and G come into view and may be considered as candidates for describing the location of building D. A function is required to establish which of these is most suitable, requiring knowledge of the candidates and consideration of the overlap extent. In this case the overlap between D and E is minimal, and although F has a larger overlap it would still make more sense to describe D as “opposite the Park”, as these features share the greatest overlap and the Park is a very recognisable feature.

As a further example, when the observer is at Point 2 looking for building G then the park can no longer be considered opposite, however either building E or F could be used. Factors including the saliency of the building, its visibility, distance from the target, and the number of items between each should be considered. Assuming the visibility of both E and F were high (clear views) then E would form the most logical

choice as it is the first item viewed from the roadside and most readily identifiable. However if F was a visually prominent landmark, such as a church, then it would take precedence despite being further from the target as its saliency allows it to form a more useful descriptor.

Saliency is a measure of the prominence of a feature in the neighbourhood, and there are methods to quantify such distinctiveness [32, 33]. Typically factors including visual appearance and semantic interest are considered by comparing items in the neighbourhood to establish the most easily recognisable and rare features. It is of particular importance in choosing candidates for forming referring expressions, as when targeting a building by describing it as opposite a ‘tree’ it may be logically true, but worthless if the entire street is filled with trees and all houses are opposite a tree. Therefore, when constructing a referring expression, the number of other entities which share a similar relation need to be considered, to minimise the confusion caused by the statement.

Fuzzy classes may be used to establish the most attractive entity in an ‘opposite’ relation, by considering all alternatives and awarding class memberships between 0 and 1 according to a number of factors. The weighting between factors may be adjusted according to the current task, for example car drivers may favour number of items between as scanning opportunities are more limited while driving, whereas pedestrians may favour saliency as they have more freedom to view the surroundings and wish to locate the most prominent landmarks in a wider field of view.

Most suitable entity = $f(V,S,N,D,O)$, where:

- V – visibility (degree of visibility of all items from a single observation point)
- S – saliency (prominent, minimise confusability)
- N – number of items between (measure of separation by entity count)
- D – distance apart (close items preferred)
- O – degree of overlap

A slightly modified set of rules are necessary when considering two-dimensional common features as discussed next.

3.2 Two-Dimensional Common Features

For linear common features the entity overlaps, used to identify whether features are opposite, were determined by considering the first and last intersections of a line perpendicular to the linear feature with each entity (as shown in Fig. 3, e.g., A1-A2). In cases where the common feature is a region, an alternative rule is required to determine overlap, and consequently opposition.

When the observer occupies a space inside the common region, for example standing in a square, then two features may be described as opposite one another by considering the observer as a central point with a feature occupying the *in front* space, and one in *behind* space. As an example, if the observer looks towards feature A as shown in Fig. 4(i), then B can be classed as in the opposite direction, according to the in front/behind relation outlined in Fig. 2. However, if the observer is outside of the common region then the relations may be calculated according to a division of space based on the Orientated Minimum Bounding Box (OMBB), as shown in Fig. 4(ii). In this case the OMBB is drawn around the common region, and the centre point

determined. Lines are extrapolated from the centre point to the corner and edge midway points of the bounding box, creating 8 triangular zones. For any two entities to be considered opposite each other with respect to the square they must occupy zones whose sum adds up to ten, according to the number system shown. Therefore no matter where the observer is located the relation of A, B1 and the square would be classed as ‘opposite’, assuming all entities were visible. Entities occupying a neighbouring zone are also considered to be ‘opposite’, so that A, B2 and A, B3 would be describe as sharing an ‘opposite’ relation with respect to the square. This works for all common region shapes (e.g. lakes), as the algorithm is based on the orientated bounding box.

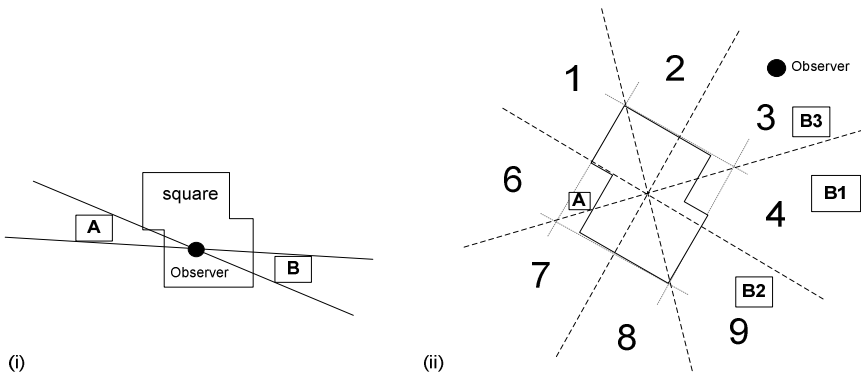


Fig. 4. Using the term “Opposite” across a region

3.3 Additional Considerations for Usage

The concepts outlined so far have failed to include a measure of the appropriateness of the inclusion of the term with respect to the observer’s viewing distance, and object sizes. Instead they have looked to determine the most suitable candidate for the relation. However at greater viewing distances, and for smaller items, it may be harder for the observer to judge when two entities share an opposite relation and it may be necessary to restrict the inclusion of the term in a referring expression according to the percentage of the field of view occupied by the items. This approach accommodates entity scale, such that a house may be described as opposite a bakery from only close range, while a park may be described as opposite the hills from greater distances.

Additionally when a referring expression is used in a more general description of a region, and not to identify a particular target, consideration must be given to the ordering of features in the relation. Jackendorf [5] makes the observation that not all relations are symmetrical, and that “the house is next to the bike” makes less sense than “the bike is next to the house”. When considering the “opposite” relation it makes more sense to use the most salient feature as the reference object, such as “turn right after you see a hut opposite a lake”, whereby the viewer’s attention should be more naturally drawn to the lake, and the decision point confirmed once the hut has been located.

4 Conclusions and Future Work

When constructing descriptions of a feature it is useful to include spatial prepositions to guide the user’s attention. This is particularly relevant when forming referring expressions for use in speech based LBSs while exploring a city. The case of spatially ‘opposite’ an entity has been considered in this paper raising a number of observations about how it may be determined and constructed from GIS datasets. The research has shown that to ensure meaningful descriptions it is necessary to determine whether features are visible to the user by calculating their visual exposure and establish a common reference entity to define their relation. One-dimensional and two-dimensional features have been examined for this purpose. Three-dimensional entities have not yet been explored, but could be included in future work. They may be of particular use when constructing descriptions for features on the side of a building, such as “the window opposite the balcony“, to limit the vertical scan region.

When a number of possible candidates are found it is necessary to select the most useful by determining its saliency and recognisability, to assist in guiding the user’s attention and minimise risk of target confusion. Factors including its visibility, distance, and the number of other similar items in the scene are considered. Future work should examine the weighting of these inputs, to determine the most suitable values for particular tasks. The models presented should be developed further through user trials, and may then be adopted as part of a wider set of defined spatial relations for use in urban LBS, and the geosemantic web.

References

1. Narzt, W., et al.: Augmented reality navigation systems. *Universal Access in the Information Society*, 1–11 (2006)
2. Hollerer, T., et al.: Exploring MARS: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers and Graphics (Pergamon)* 23(6), 779–785 (1999)
3. Goose, S., Sudarsky, S., Zhang, X., Navab, N.: Speech-Enabled Augmented Reality Supporting Mobile Industrial Maintenance. In: *PERVASIVE Computing* (2004)
4. Bartie, P.J., Mackaness, W.A.: Development of a speech-based augmented reality system to support exploration of cityscape. *Transactions in GIS* 10(1), 63–86 (2006)
5. Jackendoff, R.: *Languages of the Mind*. MIT Press, Cambridge (1992)
6. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2), 233–263 (1995)
7. Espinoza, F., et al.: *GeoNotes: Social and Navigational Aspects of Location-Based Information Systems*. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) *UbiComp 2001*. LNCS, vol. 2201, pp. 2–17. Springer, Heidelberg (2001)
8. Francioni, J.M., Jackson, J.A., Albright, L.: The sounds of parallel programs: *IEEE* (2002)
9. Allport, D., Antonis, B., Reynolds, P.: On the Division of Attention: a Disproof of the Single Channel Hypothesis. *Quarterly Journal of Experimental Psychology* 24, 225–235 (1972)
10. Ishii, H., Kobayashi, M., Arita, K.: Iterative design of seamless collaboration media. *Communications of the ACM* 37(8), 83–97 (1994)
11. Winter, S., Wu, Y.: The “spatial Turing test”. In: *Colloquium for Andrew U. Frank’s 60th Birthday*. Geoinfo Series, Department for Geoinformation and Cartography, Technical University Vienna, Vienna (2008)

12. Meng, L.: Ego centres of mobile users and egocentric map design. In: Meng, L., Zipf, A., Reichenbacher, T. (eds.) *Map-based Mobile Services*, pp. 87–105. Springer, Berlin (2005)
13. Reichenbacher, T.: Adaptive egocentric maps for mobile users. In: Meng, L., Zipf, A., Reichenbacher, T. (eds.) *Map-based Mobile Services*, pp. 143–162. Springer, Berlin (2005)
14. Egenhofer, M.J., Herring, J.: *A mathematical framework for the definition of topological relationships* (1990)
15. Clementini, E., Di Felice, P.: A comparison of methods for representing topological relationships. *Information Sciences-Applications* 3(3), 149–178 (1995)
16. Clementini, E., Billen, R.: Modeling and computing ternary projective relations between regions. *IEEE Transactions on Knowledge and Data Engineering* 18, 799–814 (2006)
17. Hernández, D.: Relative representation of spatial knowledge: The 2-D case. In: Mark, D.M., Frank, A.U. (eds.) *Cognitive and linguistic aspects of geographic space*, pp. 373–385. Kluwer Academic Publishers, Netherlands (1991)
18. De Smith, M.J., Goodchild, M.F., Longley, P.: *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*. Troubador Publishing (2007)
19. De Floriani, L., Marzano, P., Puppo, E.: Line-of-sight communication on terrain models. *International Journal of Geographical Information Systems* 8(4), 329–342 (1994)
20. Stucky, J.L.D.: On applying viewshed analysis for determining least-cost paths on Digital Elevation Models. *International Journal of Geographical Information Science* 12(8), 891–905 (1998)
21. Fisher, P.F.: Extending the applicability of viewsheds in landscape planning. *Photogrammetric Engineering and Remote Sensing* 62(11), 1297–1302 (1996)
22. Baer, W., et al.: Advances in Terrain Augmented Geometric Pairing Algorithms for Operational Test. In: *ITEA Modelling and Simulation Workshop*, Las Cruces, NM (2005)
23. Fisher-Gewirtzman, D., Wagner, I.A.: Spatial openness as a practical metric for evaluating built-up environments. *Environment and Planning B: Planning and Design* 30(1), 37–49 (2003)
24. Tandy, C.R.V.: *The isovist method of landscape*. In: *Symposium: Methods of Landscape Analysis*. Landscape Research Group, London (1967)
25. Benedikt, M.L.: To take hold of space: isovists and isovist fields. *Environment and Planning B* 6(1), 47–65 (1979)
26. Morello, E., Ratti, C.: A digital image of the city: 3D isovists in Lynch's urban analysis. *Environment and Planning B* 36, 837–853 (2009)
27. Llobera, M.: Extending GIS-based visual analysis: the concept of visualscapes. *International Journal of Geographical Information Science* 17(1), 25–48 (2003)
28. Bartie, P.J., et al.: *Advancing Visibility Modelling Algorithms for Urban Environments*. *Computers Environment and Urban Systems* (2010); doi:10.1016/j.compenvurbsys.2010.06.002
29. Billen, R., Clementini, E.: Projective relations in a 3D environment. In: Raubal, M., Miller, H.J., Frank, A.U., Goodchild, M.F. (eds.) *GIScience 2006*. LNCS, vol. 4197, pp. 18–32. Springer, Heidelberg (2006)
30. Bartie, P., et al.: *A Model for Egocentric Projective Spatial Reasoning based on Visual Exposure of Features of Interest* (forthcoming)
31. Merriam-Webster: *Opposite* (2010)
32. Raubal, M., Winter, S.: Enriching wayfinding instructions with local landmarks. In: Egenhofer, M.J., Mark, D.M. (eds.) *GIScience 2002*. LNCS, vol. 2478, pp. 243–259. Springer, Heidelberg (2002)
33. Elias, B.: *Determination of landmarks and reliability criteria for landmarks* (2003)

Cognitive Adequacy of Topological Consistency Measures^{*}

Nieves R. Brisaboa¹, Miguel R. Luaces¹, and M. Andrea Rodríguez²

¹ Database Laboratory, University of A Coruña
Campus de Elviña, 15071 A Coruña, Spain
{`brisaboa,luaces`}@udc.es

² Universidad de Concepción, Chile
Edmundo Larenas 215, 4070409 Concepción, Chile
`andrea@udec.cl`

Abstract. Consistency measures provide an indication on how much a dataset satisfies a set of integrity constraints, which is useful for comparing, integrating and cleaning datasets. This work presents the notion of consistency measures and provides an evaluation of the cognitive adequacy of these measures. It evaluates the impact on the consistency measures of different parameters (overlapping size, external distance, internal distance, crossing length, and touching length) and the relative size of geometries involved in a conflict. While a human-subject testing supports our hypotheses with respect to the parameters, it rejects the significance of the relative size of geometries as a component of the consistency measures.

Keywords: Topological similarity measure, inconsistency measures, spatial inconsistency.

1 Introduction

A dataset is consistent if it satisfies a set of integrity constraints. These integrity constraints define valid states of the data and are usually expressed in a language that also defines the data schema (logical representation). Consistency measures provide an indication on how much a dataset satisfies a set of integrity constraints. They are useful to compare datasets and to define strategies for data cleaning and integration. Traditionally, consistency in datasets has been a binary property, the dataset is either consistent or not. At most, consistency measures count the number of elements in a dataset that violate integrity constraints, but the concept of *being partially consistent* does not exist. Spatial information rises new issues regarding the degree of consistency because the comparison of spatial data requires additional operators beyond the classical comparison operators ($=$, $>$, $<$, \leq , \geq , \neq). Geometries are typically related by topological or other spatial relations, upon which different semantic constraints may be defined.

^{*} This work was partially funded by Fondecyt 1080138, Conicyt-Chile and by “Ministerio de Ciencia e Innovación” (PGE and FEDER) refs. TIN2009-14560-C03-02, TIN2010-21246-C02-01 and by “Xunta de Galicia (Fondos FEDER)”, ref. 2010/17.

In a previous work [9], we defined a set of measures to evaluate the violation degree of spatial datasets with respect to integrity constraints that impose topological relations on the semantics of spatial objects. These measures contextualize the relative importance of the difference of the topological relation between two geometries with respect to an expected topological relation by considering the size of geometries within the whole dataset. In this paper we carry out a human-subject testing to evaluate all measures where we analyze not only the degree of violation in itself, but also the impact of the relative size of objects in the dataset as a component of the degree of violation. Three hypotheses were analyzed: (1) The four parameters used by the measures (i.e., *external distance*, *internal distance*, *crossing segment*, and *overlapping size*) are perceived by subjects as factors of the degree violation. (2) The *touching length* of geometries in touch, which is also not considered by the proposed measures, is not considered by subjects as a factor of the degree of violation. (3) The size of geometries involved in a conflict, with respect to other objects in the dataset, is perceived by subjects as a factor of the degree of violation.

The organization of the paper is as follows. Section 2 makes a revision of related work. In particular it analyzes different approaches to comparing topological relations. Section 3 presents preliminary concepts and consistency measures first defined in [9], while Section 4 describes the human-subject testing and its main results. Final conclusions and future research directions are given in Section 5.

2 Related Work

Related work addresses similarity measures of topological relations. Similarity measures are useful to compare the topological relation between geometries stored in a dataset with respect to an expected topological relation as expressed by an integrity constraint. We distinguish qualitative from quantitative approaches to comparing topological relations. A qualitative representation of topological relations uses a symbolic representation of spatial relations, such as the topological relations defined by Egenhofer and Franzosa [3] or by Randell *et al.* [8]. Under this representation, a similarity measure compares topological relations by the semantic distance between relations defined in a conceptual neighborhood graph [7]. The disadvantage of comparing topological relations from a qualitative perspective is that it does not make distinction between particular geometries. For example, it does not distinguish between two pairs of geometries, both disjoint, but where in one case the geometries are very close and in the other case the geometries are far apart. Even more, in most cases when semantic distance is used, all edges in the conceptual graph will usually have the same weight in the determination of the semantic distance.

A quantitative representation of topological relations is given in [1] by the distance and angle between the centroid of the objects. Using this representation, similarity between topological relations is defined as the inverse of the difference between representations. Another study [4] defines ten quantitative measures that characterize topological relations based on metric properties, such as

length, *area*, and *distance*. The combination of these measures gives an indication of the topological relations and their associated terms in natural language (such as *going through* and *goes up to*). The problem of using the previous measures for evaluating the degree of inconsistency is that although datasets handle geometries of objects, constraints are expressed by qualitative topological relations, and therefore, only a symbolic representation of the expected topological relations exists.

In the spatial context, only the work in [9] introduces some measures to compare the consistency of different datasets. In this previous work, given an expected topological relation between any two objects with particular semantics, a violation degree measure quantifies how different is the topological relation between the objects from the expected relation expressed by a topological constraint. While this previous work provides an evaluation with respect to semantic distance, it does not evaluate the cognitive adequacy of the measures neither the impact of the relative size of objects in the quantification of inconsistency.

3 Definition of Consistency Measures

In this work we concentrate on integrity constraints that impose topological relations depending on the semantics of objects. Particularly, we extend the *Topological Dependency* (TD) constraints defined in [2] or the semantic constraints in [5] to consider a wider range of constraints found in practise. The definitions of topological relations are those in the Open Geospatial Consortium Simple Feature Specification [6] and used in the subsequent specification of topological dependency constraints.

Let T be a topological relation, and $P(\bar{x}_1, g_1)$ and $R(\bar{x}_2, g_2)$ be predicates representing spatial entities with non-empty sequences of thematic attributes \bar{x}_1 and \bar{x}_2 , and geometric attributes g_1 and g_2 , respectively. A conditional topological dependency constraint is of the form:

$$\forall \bar{x}_1 \bar{x}_2 \bar{g}_1 \bar{g}_2 (P(\bar{x}_1, g_1) \wedge R(\bar{x}_2, g_2) \wedge \psi \rightarrow T(g'_1, g'_2))$$

where g'_1 is either g_1 , $\Theta_1[g_1]$ or $\Theta_2[g_1, d]$, with d a constant and Θ_1 and Θ_2 geometric operators that return a geometry. Geometry g'_2 is defined in the same way than g'_1 where g_1 is replaced by g_2 . Also ψ is an optional formula in conjunctive normal form (CNF) defined recursively by:

- (a) $y\Delta z$ is an atomic formula, with $y \in \bar{x}_1$, $z \in \bar{y}_2$, and Δ a comparison operator ($=, \neq, >, < \leq, \geq$).
- (b) $T'(g_1, g_2)$ is an atomic formula, with T' a topological relation.
- (c) $\theta_1[g_1]\Delta c$ or $\theta_1[g_1]\Delta\theta_2[g_2]$ are atomic formula, with c a constant, Δ a comparison operator ($=, \neq, >, < \leq, \geq$), and θ_1, θ_2 geometric operators that return real numbers (e.g., area, length, perimeter, and so on).
- (d) An atomic formula is a CNF.
- (e) $(t_1 \vee t_2)$ is a clause with t_1 and t_2 atomic formulas.

- (f) A clause is a CNF.
- (g) $c_1 \wedge c_2$ is a CNF formula with c_1 and c_2 clauses or CNF formulas.

Using the previous definitions and considering predicate $county(idc, ids, g)$ and $state(ids, g)$, a CTD could be “a county must be within the state to which it belongs”:

$$\forall idc, ids, g_1, g_2 (county(idc, ids, g_1) \wedge state(ids, g_1) \rightarrow \text{Within}(g_1, g_2))$$

Let ψ be an integrity constraint of the form [3](#) with topological relation T . A pair of tuples $P(\bar{u}_1, s_1)$ and $P(\bar{u}_2, s_2)$ is inconsistent if the antecedent in ψ instantiated by $P(\bar{u}_1, s_1)$ and $P(\bar{u}_2, s_2)$ is satisfied but the consequent not. We defined in [9](#) a collection of measures to compute the violation degree for all topological relations between surfaces. Two main components define the degree of violation: (1) the magnitude of the conflict and (2) the relevance of the conflict. The magnitude of the conflict measures the difference between the relation held by the two geometries and the expected relation between them. For example, if two geometries must touch but they are disjoint, the magnitude of the conflict is proportional to the separation between the geometries. In the case that two geometries must touch but they overlap, the magnitude of the conflict is proportional to the overlapping area between the geometries. On the other hand, the relevance of the conflict is determined using the relative size of the objects.

We have considered five different parameters to compute the magnitude of the conflict with respect to different topological relations: (1) the *external distance* between disjoint geometries, which has an impact on conflicts risen by the separation of geometries when they must intersect (i.e., equal, touch, overlap, or within). (2) The *internal distance* between geometries when one is within the other geometry, which has an impact on conflicts when geometries must be externally connected (i.e, they must touch or they must be disjoint). (3) The *overlapping size* of geometries that are internally connected, which has an impact on conflicts when geometries must be externally connected (4) The *crossing length* that represents the length of the minimum segment of a curve that crosses another curve or a surface, which has an impact on conflicts when geometries must be externally connected. (5) The *touching length* between geometries represents the length of the common boundary between geometries. In the definition of the violation degree measures, we have used the first four parameters and we have not used the *touching length* in any of the measures.

4 Empirical Validation of Consistency Measures

To validate the cognitive adequacy of our measures, we designed human-subject tests oriented to evaluate both components of our measures: (i) the parameters that measure the magnitude of conflicts (*external distance*, *internal distance*, *overlapping size* and *crossing length*) and (ii) the computation of the conflict relevance using the relative size of the objects. We wanted to evaluate whether

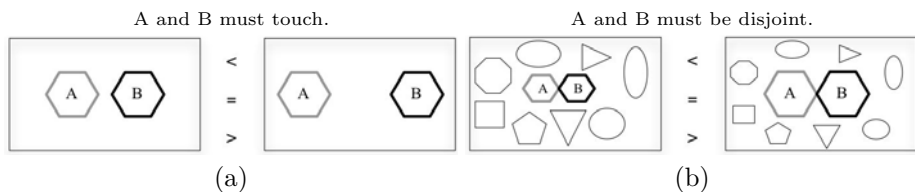


Fig. 1. Comparison of the violation degree: (a) example of section 1 and (b) example of section 2

these parameters have an impact on the violation degree perceived by subjects and, therefore, if they are cognitively adequate to define the violation degree of CTDs. We also wanted to evaluate whether our decision to ignore the parameter *touching length* was correct, that is whether the *touching length* has an impact on the magnitude of conflict (e.g., geometries that should touch must but are disjoint).

The following three hypotheses were studied:

- H₁: *External distance, internal distance, crossing length, and overlapping size* are perceived and used by subjects to evaluate the degree of violation of CTDs.
- H₂: *Touching length* is not considered by subjects to evaluate the degree of violation of CTDs.
- H₃: The relative *size* of the geometries that participate in the violation of CTDs with respect to other objects in the dataset affects the perceived violation degree. More precisely, the larger the geometries the larger the violation degree.

The subjects of the test were 69 second year computer science students. The test was performed at the beginning of a normal class. They were not given any instructions in addition to those written in the test. Nine of the subjects were eliminated because they did not complete the test or their answers showed a clear misunderstanding of the questions. Students did not receive any compensation for answering the test, which explains why some students did not answer all questions.

The test begins with a set of figures describing the topological relations that combine geometries with different dimensions (*surfaces, curves and points*). These figures are the only reference to understand the meaning of each topological relation. Then, a short paragraph explains that the objective of the test is to evaluate the violation degree of the topological relations in the figures that follow in the next pages. The instructions emphasize that the violation of the expected topological relation was referred exclusively to the geometries clearly identified in each figure with colors blue and yellow, explaining that any other geometry coloured in black is correct.

The test consists of 3 sections. Section 1 includes 24 questions to check whether the parameters used by our measures are those perceived by the subjects as a

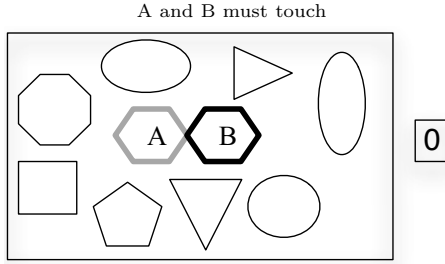


Fig. 2. Example of section 3

factor of the violation degree (H_1 and H_2). Figure 1(a) shows a question of this section. The task is to choose whether or not one of them shows a larger ($>$), equal ($=$) or smaller ($<$) violation degree of an expected topological relation among geometries A and B. Specifically, we check that the larger the value of the parameter that defines the magnitude of conflicts in our measures, the larger the perceived violation.

The questions in this section check the four parameters considered in our measures (i.e., *external distance*, *internal distance*, *crossing length*, and *overlapping size*) as well as the influence of the *touching length* between geometries. The questions also represent a balanced selection of different topological relations and types of geometries (mainly surfaces and curves, but also points).

Section 2 includes 14 questions similar to those in Section 1. The difference is that the figures now include black geometries that represent the context where the violation occurs. Figure 1(b) shows a question of section 2. This section is designed to prove the influence of the context, that is, the influence in the perceived violation of the size of the two geometries in conflict with respect to the other geometries in the dataset (H_3).

Section 3 shows each single figure in the questions in section 3. Fourteen of them represent small blue and yellow geometries and large context geometries and fourteen of them represent large blue and yellow geometries and small context geometries. For each figure, the subjects were asked to provide a value of the degree of violation between 0 and 100. The example given to the subjects (see Figure 2) does not violate the expected topological relation and, consequently, the assigned violation degree value is 0. We decided to use this example with value 0 to avoid any influence on the value given by subjects in case of a violation. This section is designed to validate our measures by evaluating whether or not the violation degrees computed by our measures is in concordance with the violation degrees given by the subjects. If there is a high correlation between the scores provided by the subjects and the values obtained with our measures, we can conclude that our measures are valid and that they reflect the opinions of the subjects.

We decided not to check all combinations of topological relations between geometries versus an expected topological relation because this would require 64 questions in the test. We assume that if a parameter (e.g. the *external distance*

between geometries) was perceived and used to decide a degree of violation between geometries that must *touch* but are *disjoint*, then it will be also perceived and used to decide the degree of violation between two geometries that must *overlap* but are *disjoint*. Similarly, we assume that if a parameter is perceived and used for geometries of a particular dimension (e.g., surfaces), then it will be also for geometries of other dimension (e.g., curves or points). Furthermore, we did not include figures in the test where the expected relation is *Equal*.

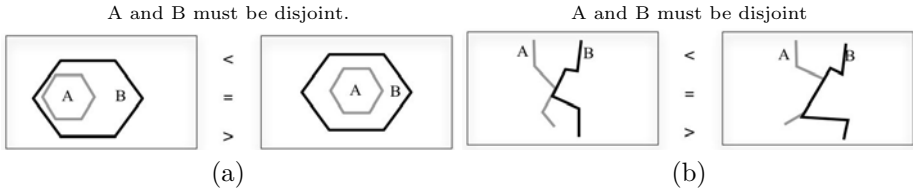


Fig. 3. Comparison of the violation degree: (a) a surface within another surface (b) a point within another point

Table 1 shows the raw data obtained in the different questions of section 1. In this table, *Expected* refers to the expected topological relation as expressed by a CTD, *Actual* refers to the real relation between the geometries in the question, *Geometries* refers to the type of geometries involved in the relation, *Parameter* refers to the parameter evaluated by the test, ‘+’ refers to the percentage of answers that perceive a positive impact of the parameter on the violation degree, ‘=’ refers to the percentage of answers that do not perceive any effect of the parameters on the violation degree, and ‘-’ refers to the percentage of answers that give an inverse influence of the parameter on the violation degree.

One can see that there is always around a 10% of subjects that answered a different option than the one we expected. When some of the subjects were asked about the reasons of their answers, many of them said it was a mistake. We realize now that it was difficult for the subjects to keep the level of concentration to mark correctly < or > on all the questions. The results prove beyond any doubt that the parameters *external distance*, *overlapped size* and *crossing length* are consistently used by the subjects to evaluate the violation degree (hypothesis H₁). We performed the Student’s t-test to evaluate the significance of our results and we found that the average percentage of correct answers for the parameter *external distance* was significant at the 99% level. In the same way the results showed that for the parameters *crossing length* and *overlapping size* the average percentage we obtained is significant at the 95% level.

Results regarding the parameter *internal distance* are more difficult to analyze. For questions 2 and 23 there was a large percentage of subjects (around 40%) that did not answer as we expected. Figure 3(a) shows question number 2. When asked why they answered this way, the subjects said that geometries were more disjoint when the internal distance between them was larger. We believe that this was due to a misunderstanding of the topological relation *Disjoint*.

Table 1. Results section 1

#	Expected	Actual	Geometries	Parameter	+	% =	% -
1	Disjoint	Overlaps	surface × surface	Overlapping size	83	8	8
2	Disjoint	Within	surface × surface	Internal distance	48	28	23
3	Touches	Overlaps	surface × surface	Overlapping size	68	20	12
4	Touches	Within	surface × surface	Internal distance	62	22	17
5	Overlaps	Disjoint	surface × surface	External distance	87	3	10
6	Overlaps	Touches	surface × surface	Touching length	32	48	20
7	Overlaps	Within	surface × surface	Internal distance	53	40	7
8	Within	Overlaps	surface × surface	Overlapping size	68	17	15
9	Within	Touches	surface × surface	Touching length	20	65	15
10	Disjoint	Overlaps	curve × curve	Crossing length	68	23	8
11	Disjoint	Overlaps	curve × curve	Touching length	58	35	7
12	Disjoint	Overlaps	curve × curve	External distance	83	5	12
13	Disjoint	Overlaps	curve × curve	External distance	82	10	8
14	Disjoint	Overlaps	curve × curve	External distance	80	12	8
15	Disjoint	Overlaps	surface × curve	Touching length	40	52	8
16	Disjoint	Overlaps	surface × curve	Crossing length	83	8	8
17	Disjoint	Overlaps	surface × curve	Internal distance	45	45	10
18	Disjoint	Overlaps	surface × curve	Internal distance	50	28	20
19	Disjoint	Overlaps	surface × curve	External distance	87	12	7
20	Disjoint	Overlaps	surface × curve	Crossing length	68	23	8
21	Disjoint	Overlaps	surface × curve	External distance	72	18	10
22	Disjoint	Overlaps	surface × curve	Internal distance	47	43	10
23	Disjoint	Overlaps	surface × point	Internal distance	52	28	20
24	Disjoint	Overlaps	curve × point	External distance	67	27	7

Question 7 was another case where many subjects gave unexpected answers due to the misunderstanding of the topological relation *Overlaps*. When asked why they considered that both figures have the same degree of violation, many subjects answered that in both cases there was no violation because when a geometry is within another geometry they also overlap each other.

After eliminating questions 2, 7, and 23 where there was some misinterpretation, the Student's t-test shows that the average percentage is significant at the 85% level. This means that the *internal distance* parameter affects the perception of consistency by subjects. However, further analysis must be performed in the future to better understand why in some cases the internal distance is considered important and in some cases not.

Finally, as H_2 states, the *touching length* is not a useful parameter to evaluate the degree of violation of a topological constraint. Only question 11 shows a higher percentage of subjects that considered the impact of *touching length* important on the violation degree. However, as it can be seen in Figure B(b), this question was the only one where the geometries in each figure were not exactly the same. Thus, there may be factors other than *touching length* involved in the subjects' answers.

The results for Section 2 indicate that 35%, 35% and 30% of the subjects considered that the size of geometries in conflict had a positive, equal or negative impact on the violation degree, respectively. These results do not support our hypothesis H_3 , but they also do not support the alternative hypothesis that states that the relative size has no impact or a negative impact on the violation degree. Therefore, we cannot extract any conclusion over the influence of the context in the evaluation of the violation degree. These results are in concordance with the results obtained in section 3.

Finally, for each question in Section 3, we computed the average score given by the 60 subjects and the value of our measure. Then, we computed the Pearson correlation between both series of values. The correlation coefficient equals to 0.54. Given that this is a very small value, we excluded the relative weight from the computation of our measures and we obtained a correlation coefficient of 0.84. This result supports the conclusion that we extracted from section 2. We can conclude that the relative size of the geometries is not considered to be important by the subjects. Or at least, that the subjects consider more important the magnitude of the conflicts than the relative size of the geometries with respect to other objects in the dataset.

5 Conclusions

We obtained two types of conclusions from this work: some related to the definition of the measures and other related to the methodology to evaluate these measures. Overall, it is clear that the use of parameters such as *external distance* and *overlapping size* allows us to discriminate situations that a semantic distance approach to comparing topological relations would otherwise overlook. Unless we consider the particularity of the geometries, a pair of geometries holding the same topological relation will always have the same degree of violation with respect to a different expected topological relation.

The results of the empirical evaluation indicate that the parameters that define our measures agree with the human perception of the violation degree. The only one that was not fully confirmed was the *internal distance*, which requires further evaluation. Contrary to our expectation, the results also indicate that the relative size of geometries in conflict with respect to other geometries in the dataset has less impact on the evaluation of the violation degree than what we expected. This is confirmed by the increase in the correlation of the scores given by the users in Section 3 of the test when we eliminated the effect of the relative size of geometries from the measures.

We confirmed that the design of the test is critical. There are two basic problems that need to be solved for future empirical evaluations: the difficulty of the task and the knowledge the subjects need about topological relations.

Regarding the difficulty of the task, the questions in the test require a high level of concentration. This explains the high number of mistakes we found in the questions of section 1. On the other hand, in section 3 the task was easier because only a score was requested. However, the subjects complained about the

difficulty and many of them moved back and forward changing the scores while answering the questions.

The problem of the knowledge about the topological relations is harder to solve. Explaining the meaning of the topological relations before the test does not guarantee that they use these definitions instead of their own interpretations. For instance, some of the subjects considered that two surfaces that *overlap*, also *touch*, or that two surfaces that are one *within* the other also *overlap*. The only way to avoid this problem is to train the subjects in the meaning of the topological relations. However, it may be difficult to do this without instructing them in our view of the parameters that define the measure of the violation degree. Probably, the safest way to tackle this problem is to select the figures and their relations very carefully to avoid that subjects misunderstand topological relations.

References

1. Berreti, S., Bimbo, A.D., Vicario, E.: The computational aspect of retrieval by spatial arrangement. In: Intl. Conference on Pattern Recognition (2000)
2. Bravo, L., Rodríguez, M.A.: Semantic integrity constraints for spatial databases. In: Proc. of the 3rd Alberto Mendelzon Intl. Workshop on Foundations of Data Management, Arequipa, Peru, vol. 450 (2009)
3. Egenhofer, M., Franzosa, R.: Point Set Topological Relations. IJGIS 5, 161–174 (1991)
4. Egenhofer, M., Shariff, A.: Metric details for natural-language spatial relations. ACM Transactions on Information Systems 16(4), 295–321 (1998)
5. Hadzilacos, T., Tryfona, N.: A Model for Expressing Topological Integrity Constraints in Geographic Databases. In: Frank, A.U., Formentini, U., Campari, I. (eds.) GIS 1992. LNCS, vol. 639, pp. 252–268. Springer, Heidelberg (1992)
6. OpenGis: Opengis Simple Features Specification for SQL. Tech. rep., Open GIS Consortium (1999)
7. Papadias, D., Mamoulis, N., Delis, V.: Algorithms for querying spatial structure. In: VLDB Conference, pp. 546–557 (1998)
8. Randell, D., Cui, Z., Cohn, A.: A spatial logic based on regions and connection. In: Nebel, B., Rich, C., Swarthout, W. (eds.) Principles of Knowledge Representation and Reasoning, pp. 165–176. Morgan Kaufmann, San Francisco (1992)
9. Rodríguez, M.A., Brisaboa, N.R., Meza, J., Luaces, M.R.: Measuring consistency with respect to topological dependency constraints. In: 18th ACM SIGSPATIAL Intl. Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, San Jose, CA, USA, pp. 182–191 (2010)

The Neighborhood Configuration Model: A Framework to Distinguish Topological Relationships between Complex Volumes

Tao Chen and Markus Schneider*

Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611, USA
{tachen,mschneid}@cise.ufl.edu

Abstract. Topological relationships between spatial objects are considered to be important for spatial databases. They lead to topological predicates, which can be embedded into spatial queries as join or selection conditions. Before rushing into the implementation of topological predicates, topological relationships between spatial objects must be first understood and clarified. This requires a detailed study of a vast number of possible spatial configurations at the abstract level, and as a result, methods that are able to classify and identify as many as possible different spatial configurations are needed. While a lot of research has already been carried out for topological relationships in the 2D space, the investigation in the 3D space is rather neglected. Developed modeling strategies are mostly extensions from the popular 9-intersection model which has been originally designed for simple 2D spatial objects. We observe that a large number of topological relationships, especially the ones between two complex 3D objects are still not distinguished in these models. Thus, we propose a new modeling strategy that is based on point set topology. We explore all possible neighborhood configurations of an arbitrary point in the Euclidean space where two volume objects are embedded, and define corresponding *neighborhood configuration flags*. Then, by composing the Boolean values of all flags, we uniquely identify a topological relationship between two complex volume objects.

1 Introduction

Topological relationships like *overlap*, *inside*, or *meet* describe purely qualitative properties that characterize the relative positions of spatial objects and are preserved under affine transformations such as translation, scaling, and rotation. The exploration of topological relationships between spatial objects is an important topic in fields like artificial intelligence, cognitive science, geographical information systems (GIS), linguistics, psychology, robotics, spatial database

* This work was partially supported by the National Science Foundation under grant number NSF-IIS-0915914.

systems, and qualitative spatial reasoning. From a database and GIS perspective, their development has been motivated by the need of formally defined topological predicates as filter conditions for spatial selections and spatial joins in spatial query languages and as a support for spatial data retrieval and analysis tasks.

The central conceptual approach, upon which almost all publications in this field have been based, is the 9-intersection model (9IM) [1]. This model checks the nine intersections of the *boundary*, *interior*, and *exterior* of a spatial object with the respective components of another spatial object for the topologically invariant criterion of non-emptiness. Extensions have been proposed to obtain more fine-grained topological predicates. However, the main focus has been on spatial objects in the 2D space; the study of topological relationships between spatial objects in the 3D space has been rare. An available strategy is to apply 9IM based models and to investigate the total number of relationships that can occur in reality between spatial objects in the 3D space. However, the third dimension introduces more complicated topological situations between 3D spatial objects. When directly applying the 9IM based models to 3D complex spatial objects like volumes, they suffer from a major problem, which we call the *high granularity* problem. That is, the 9IM considers the interior, exterior, and boundary point sets as the basic elements for empty and non-empty intersection tests, which ignores the fact that the interior, exterior, and the boundary of a spatial object are also complex spatial object parts, and may have multiple components. Thus, the interaction between any pair of the basic elements from two spatial objects can be complex, and empty or non-empty intersection results may not be enough to describe such interactions. For example, the boundary of a volume object is a closed surface object, which may have multiple components. Thus, the interaction between the boundaries of two volume objects is equivalent to the interaction between two complex surface objects, which can *touch* at a point, *meet* at a face, *cross* each other, or have *touch*, *meet*, and *cross* interactions coexist on one or more components. Since 9IM based models do not have the capability to handle these interaction details for their basic elements, a large number of topological relationships between complex volumes are not distinguished.

In this paper, we propose a new framework based on point set theory and point set topology to model topological relationships between two complex volume objects. We overcome the problems raised on the 9IM by investigating the interactions between two volumes at a much lower granularity. Instead of checking the intersections of the interior, boundary, and exterior of two volumes, we investigate the interaction of the two volumes within the *neighborhood* of any point in the Euclidean space where the two volumes rest, which we call the *neighborhood configuration*. We explore all possible *neighborhood configurations*, and define corresponding Boolean *neighborhood configuration flags*. By evaluating and composing the values of the neighborhood configuration flags for a given scenario with two complex volumes, we can obtain a binary encoding of the topological relationship between the two volumes. We show that our model yields more and thus a more fine-grained characterization of topological relationships between two complex volumes compared to 9IM.

$$\begin{pmatrix} A^\circ \cap B^\circ \neq \emptyset & A^\circ \cap \partial B \neq \emptyset & A^\circ \cap B^- \neq \emptyset \\ \partial A \cap B^\circ \neq \emptyset & \partial A \cap \partial B \neq \emptyset & \partial A \cap B^- \neq \emptyset \\ A^- \cap B^\circ \neq \emptyset & A^- \cap \partial B \neq \emptyset & A^- \cap B^- \neq \emptyset \end{pmatrix}$$

Fig. 1. The 9-intersection matrix for topological relationships

Section 2 discusses related work about topological relationships. We propose our *neighborhood configuration model (NCM)* in Section 3, where we introduce in detail the definition of neighborhood configurations, the exploration of all possible neighborhood configurations, and the encoding of a topological relationship between two complex volumes. In Section 4, we compare our approach to the 9IM based models. Finally, Section 5 draws some conclusions.

2 Related Work

A special emphasis of spatial research has been put on the exploration of topological relationships (for example, overlap, inside, disjoint, meet) between spatial objects. An important approach for characterizing them rests on the so-called *9-intersection model*, which employs point set theory and point set topology [1]. The model is based on the nine possible intersections of the boundary (∂A), interior (A°), and exterior (A^-) of a spatial object A with the corresponding components of another object B . Each intersection is tested with regard to the topologically invariant criteria of emptiness and non-emptiness. The topological relationship between two spatial objects A and B can be expressed by evaluating the matrix in Figure 1. A total of $2^9 = 512$ different configurations are possible from which only a certain subset makes sense depending on the combination of spatial data types just considered. Several extensions based on the 9IM exist. Examples are the dimensionality extensions in [2,3], the Voronoi-based extensions in [4], and the extensions to complex spatial objects in [5].

A topic that has been partially formally explored at the abstract level deals with topological relationships between simple 3D spatial objects. In [6], the author applies the 9-intersection model to simply-connected 3D spatial objects, that is, simple 3D lines, simple surfaces (no holes), and simple volumes (no cavities), in order to determine their topological relationships. A total of 8 topological relationships are distinguished between two simple volumes. Zlatanova has also investigated the possible 3D topological relationships in [7] by developing a complete set of negative conditions. The 9-intersection model for 3D can be extended with the dimensionality being considered. In [8], the values of the matrix elements are extended. Besides the \emptyset and $\neg\emptyset$ symbols, the $*$ is used to indicate the omitted specification for the resulting set at this position, and the numbers (0, 1, 2, 3) refer to the dimensionality of the resulting set. Therefore, unlike the 1:1 mapping between the matrix with the topological relationships in the 9-intersection model, a matrix that contains a $*$ value represents a class of topological relationships. As a result, the topological relationships are clustered and manageable to the user. However, these 9IM based models suffer from the

mentioned high granularity problem due to the use of interior, exterior and boundary sets as basic elements.

The above models in common apply set theory to identify topological relationships. Thus, they can be categorized as *point-set based* topological relationships models. There are also other approaches that do not employ the point-set theory. The approach in [9] investigates topological predicates between cell complexes, which are structures from algebraic topology. It turns out that, due to limitations of cell complexes, the topological relationships between them are only a subset of the derived point-set based topological relationships. The topological relationships between 3D spatial objects that consist of a series of cell complexes can be described by the combination of relationships between those cells [10]. The *Dimensional Model (DM)* [11] is a model that is independent of the 9-intersection model. It defines *dimensional elements* on a spatial object, and all the dimensional elements contribute to the final result. Three levels of details for the topological relationships are developed for different application purposes. The dimension model can distinguish some cases, especially meet cases that the 9-intersection model cannot identify. However, since it leaves the abstract topological space where only point sets are used, it is not clear how the dimensional elements can be constructed.

3 The Neighborhood Configuration Model (NCM)

In the following sections, we introduce in detail our new modeling strategy for topological relationships between 3D complex volumes. Section 3.1 gives an overview of our modeling strategy. We explore all possible neighborhood configurations for a given scenario of two spatial volume objects in Section 3.2. Finally, in Section 3.3, we show how the topological relationships can be identified and encoded with the neighborhood configuration flags.

3.1 Overview

In this paper, we are interested in complex volumes that may contain cavities or multiple components. A formal definition of complex volume objects can be found in [12], which models volumes as special infinite point sets in the three-dimensional Euclidean space. Our approach is also based on point set theory and point set topology. The basic idea is to evaluate the values of a set of Boolean *neighborhood configuration flags* to determine the topological relationships between two volumes. Each *neighborhood configuration flag* indicates the existence or non-existence of a characteristic neighborhood configuration of the points in a given scenario. The neighborhood configuration of a point describes the ownerships of the points that are “near” the reference point. If the existence of a neighborhood configuration is detected, then the corresponding neighborhood configuration flag is set to true. For example, for a scenario that involves two volumes A and B , if there exists a point p whose neighboring points all belong to both A and B , then the corresponding neighborhood configuration flag *exist_nei_in_overlap* (see Definition 1(1)) is set to true. Later, this neighborhood

configuration flag contributes to the determination of the topological relationships between A and B .

3.2 Exploring Neighborhood Configuration Information

In this section, we explore all *neighborhood configurations*. We begin with some needed topological and metric concepts. Let \mathbb{R} be the set of real numbers, $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x > 0\}$, and \mathbb{R}^3 be the three-dimensional Euclidean space. We assume the existence of a Euclidean distance function $d : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ with $d(p, q) = d((x_1, y_1, z_1), (x_2, y_2, z_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$. Then for any point $p \in \mathbb{R}^3$ and $r \in \mathbb{R}^+$, the set $N_r(p) = \{q \in \mathbb{R}^3 \mid d(p, q) \leq r\}$ is called the *closed ball* with *radius* r and *center* p . Let ϵ denote a small positive value ($\epsilon \rightarrow 0$); we call the closed ball $N_\epsilon(p)$ the (closed) *neighborhood* of point p . We can think of the neighborhood around p as a tiny closed ball containing all points in \mathbb{R}^3 that are “nearest” to p .

With the necessary topological concepts introduced, we now explore the possible neighborhood configurations of points in a scenario that involves two volume objects. Let us assume two volumes A and B that are point sets in \mathbb{R}^3 ($A \subset \mathbb{R}^3, B \subset \mathbb{R}^3$). Then given a point $p \in \mathbb{R}^3$, any point q in its neighborhood $N_\epsilon(p)$ ($q \in N_\epsilon(p)$) has one of the following four possible *ownerships*: (i) $q \in A \wedge q \notin B$, (ii) $q \notin A \wedge q \in B$, (iii) $q \in A \wedge q \in B$, and (iv) $q \notin A \wedge q \notin B$. As a result, we can describe the ownership of a neighborhood by combining the ownerships of all points in it. We call this ownership description of a neighborhood the *neighborhood configuration*. With the four possible ownerships of a single point, we can obtain a total of $2^4 - 1 = 15$ possible ownership combinations for a neighborhood (the case where none of the four ownerships exists in a neighborhood is not possible). In other words, 15 *neighborhood configurations* are possible for any point in \mathbb{R}^3 where A and B are embedded. As a result, based on Definition 1 we can define 15 corresponding *neighborhood configuration flags* for a scenario that involves A and B .

Definition 1. Let $A, B \subset \mathbb{R}^3, p \in \mathbb{R}^3$, and $N_\epsilon(p)$ be a neighborhood of p with a tiny radius ϵ . We first define four ownership flags for p :

$$\begin{aligned} \alpha(A, B, p) &\Leftrightarrow \exists x \in N_\epsilon(p) : x \in A \wedge x \in B \\ \beta(A, B, p) &\Leftrightarrow \exists x \in N_\epsilon(p) : x \in A \wedge x \notin B \\ \gamma(A, B, p) &\Leftrightarrow \exists x \in N_\epsilon(p) : x \notin A \wedge x \in B \\ \lambda(A, B, p) &\Leftrightarrow \exists x \in N_\epsilon(p) : x \notin A \wedge x \notin B \end{aligned}$$

Then we define the following 15 neighborhood configuration flags for a scenario involves two objects A and B :

- (1) $exist_nei_in_overlap(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (2) $exist_nei_in_op1(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \neg\lambda(A, B, p)$

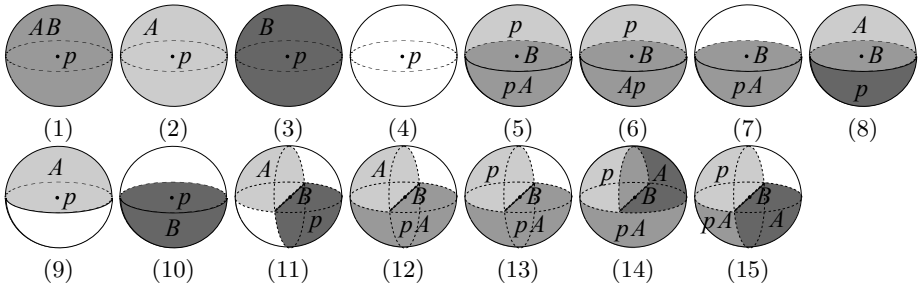


Fig. 2. The drawing of the 15 neighborhood configurations for point p

- (3) $exist_nei_in_op2(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (4) $exist_nei_in_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \lambda(A, B, p)$
- (5) $exist_nei_contain_overlap_op1(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (6) $exist_nei_contain_overlap_op2(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (7) $exist_nei_contain_overlap_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \lambda(A, B, p)$
- (8) $exist_nei_contain_op1_op2(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \beta(A, B, p) \wedge \gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (9) $exist_nei_contain_op1_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \lambda(A, B, p)$
- (10) $exist_nei_contain_op2_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \gamma(A, B, p) \wedge \lambda(A, B, p)$
- (11) $exist_nei_contain_op1_op2_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \neg\alpha(A, B, p) \wedge \beta(A, B, p) \wedge \gamma(A, B, p) \wedge \lambda(A, B, p)$
- (12) $exist_nei_contain_op2_overlap_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \neg\beta(A, B, p) \wedge \gamma(A, B, p) \wedge \lambda(A, B, p)$
- (13) $exist_nei_contain_op1_overlap_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \beta(A, B, p) \wedge \neg\gamma(A, B, p) \wedge \lambda(A, B, p)$
- (14) $exist_nei_contain_op1_op2_overlap(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \beta(A, B, p) \wedge \gamma(A, B, p) \wedge \neg\lambda(A, B, p)$
- (15) $exist_nei_contain_op1_op2_overlap_ext(A, B)$
 $\Leftrightarrow \exists p \in \mathbb{R}^3 : \alpha(A, B, p) \wedge \beta(A, B, p) \wedge \gamma(A, B, p) \wedge \lambda(A, B, p)$

The above 15 neighborhood configuration flags identify all possible interactions between two spatial volumes A and B in \mathbb{R}^3 at any single point. We demonstrate the validity of these neighborhood configuration flags by creating drawings for the corresponding neighborhood configurations in Figure 2. For example, if flag (8) yields true (Figure 2(8)), then it means that there exists a point whose neighborhood consists of points that are only from the first operand object A and points that are only from the second operand object B . The value *true* of

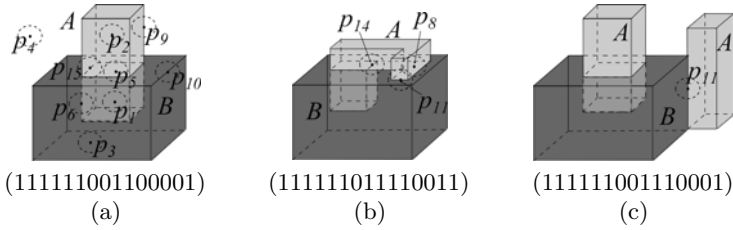


Fig. 3. Examples of topological relationship scenarios and their corresponding topological relationship encodings

this flag (*exist_nei_contain_op1-op2*) further indicates the existence of a *meeting on face* topological relationship between two volume objects *A* and *B*.

3.3 Topological Relationship Encoding with Neighborhood Configuration Flags

In the previous section, we have introduced 15 neighborhood configuration flags for two complex volume objects embedded in the Euclidean space \mathbb{R}^3 . These flags capture all topological situations for two volumes at all points in the Euclidean space \mathbb{R}^3 . In other words, we have identified the topological relationships between two volumes at a very low granularity level, which involves a small neighborhood of a point. Thus, to determine the topological relationships between two volumes, we just need to collect the values of all 15 neighborhood configuration flags. First, let *F* denote an array that stores the values of all 15 neighborhood configuration flags that are defined in the previous section. Further, we assume the ordering of the flags stored in *F* is the same as in Definition 1. Definition 2 defines the *topological relationship encoding* for two volume objects.

Definition 2. Let *A, B* be two volume objects in \mathbb{R}^3 ($A, B \subset \mathbb{R}^3$), and let *FV* denote the function that encodes the topological interaction between two volumes with respect to the *i*th neighborhood configuration flag’s value ($1 \leq i \leq 15$). Then we have:

$$FV(A, B, i) = \begin{cases} 0 & \text{if } F[i] \text{ yields false for } A \text{ and } B \\ 1 & \text{if } F[i] \text{ yields true for } A \text{ and } B \end{cases}$$

Thus, we can use a 15 bit binary array to encode the topological relationship between *A* and *B*. The definition for the topological relationship encoding *TRE* is given as:

$$TRE(A, B) = (FV(A, B, 0) FV(A, B, 1) \dots FV(A, B, 14))$$

Definition 2 introduces a 15 bits binary representation for the topological relationships between two volume objects. Figure 3 presents three examples of the topological relationship encodings. The encoding in Figure 3a indicates that 9 topological relationship flags yield true for the scenario involving *A* and *B*,

which are *exist_nei_in_overlap*, *exist_nei_in_op1*, *exist_nei_in_op2*, *exist_nei_in_ext*, *exist_nei_contain_overlap_op1*, *exist_nei_contain_overlap_op2*, *exist_nei_contain_op1_ext*, *exist_nei_contain_op2_ext*, and *exist_nei_contain_op1_op2_overlap_ext*. To demonstrate the validity of the encoding, we have marked the corresponding points p_i that validates the true value of flag $F[i]$. In Figure 3b, three additional flags, which are *exist_nei_contain_op1_op2*, *exist_nei_contain_op1_op2_ext*, and *exist_nei_contain_op1_op2_overlap*, become true due to the points p_8 , p_{11} , and p_{14} respectively. Therefore, Figure 3 presents three different topological relationships that can be distinguished by our encoding, which are *overlap* encoded by 111111001100001, *overlap with meet on face* encoded by 111111011110011, and *overlap with meet on edge* encoded by 111111001110001.

As a result, we obtain a total of $2^{15} = 32768$ possible topological relationship encoding values, which implies a total of 32768 possible topological relationships between two volume objects. However, not all encoding values represent valid topological relationships. We call a topological relationship encoding *valid* for two volumes if, and only if, it can be derived from a real world scenario that involves two volume objects. For example, there does not exist a real world scenario with the topological relationship encoding 000000000000000. Thus, the open questions are now (i) which are the valid topological relationship encodings for two complex volumes, (ii) which are the valid topological relationship encodings for two simple volumes, and (iii) what their numbers are. However, due to space limitation, our goal is only to propose the basic modeling strategy in this paper; thus, we leave these questions open.

4 The Comparison of NCM with 9IM Based Models

Our neighborhood configuration model and the 9-intersection based models share an important basis, which is point set theory and point set topology. This basis is important because it enables the modeling strategies to stay in the abstract topological space where only point sets are used and the discrete representation of a spatial object is not relevant. As a result, models based on point sets are more general than other models that leave the abstract topological space. No matter what data representation of a spatial object is used, the models based on the point sets will always work. On the other hand, models based on one particular representation of spatial objects may not work or may yields different results on another representation of the same spatial objects. For example, the *dimensional model* [11] (DM) is based on the dimensional elements of spatial objects, which determines the corner points of a region object as 0D elements. It highly depends on the representation of a spatial object, and it yields different results for a circle with a smooth boundary and a circle with its boundary approximated with a set of connected segments. Therefore, in this section, we only compare our model with the 9IM model, and show that we can distinguish more topological relationships between two volume objects.

According to [6], 8 topological relationships are distinguished between two simple 3D volume objects. Figure 4 shows the 8 configurations and their corresponding 9IM matrices. To demonstrate that our model can also distinguish

9IM	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$
NCM	(011100001100000)	(011100011110000)	(111111001100001)	(101101100101000)
9IM	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
NCM	(101101000100000)	(110110101000100)	(110110001000000)	(100100100000000)

Fig. 4. The 8 topological relationships between two volumes that can be distinguished with both 9IM and NCM

9IM	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
NCM	(111111011110011)	(111111011110001)	(111111101101001)	(111111111111001)

Fig. 5. The 4 topological relationships that can be distinguished with NCM but not 9IM

these 8 topological relationships, we also list our topological relationship encoding for each configuration in Figure 4. We observe that for all 8 topological relationships the topological relationship encodings are also different.

Moreover, the 9IM model only picks up the dominating topological relationships such as overlap, and ignores other topological relationship factors that may also exist at the same time, e.g. meet at a face. However, since we start with a very small neighborhood of each point, we are able to capture more fine-grained topological relationships. In Figure 5, we have listed 4 topological relationships that can be distinguished with our NCM model but are all identified simply as *overlap* in the 9IM model. Apart from the overlap relationship between *A* and *B*, the topological relationships like *A* meets *B* on a face from the outside (first and second scenario in Figure 5), *A* meets *B* on a face from the inside of *B* (third scenario in Figure 5), and *A* has a component inside *B* and touches *B* on a face (fourth scenario in Figure 5) are all distinguished in our model.

Therefore, we can conclude that our NCM model is more powerful than 9IM based models in terms of differentiating topological relationships between two volume objects, especially complex volumes.

5 Conclusion and Future Work

In this paper, we have introduced a novel framework for distinguishing topological relationships between two volume objects. Our approach is based on point set theory and point set topology, and by identifying the neighborhood configurations of all points, we can encode and distinguish the topological relationships. We have also made a comparison between our model and the popular 9IM model in an informal manner, and have shown that we can distinguish more topological relationships than the 9IM can. However, we were not able to explore all topological relationships that our model can distinguish in a systematical way due to space limitation. Thus, in the future, our goal is to find the answer to the open questions that we mentioned at the end of Section 3.3, and further extend our model to other complex 3D spatial objects like surfaces and lines.

References

1. Egenhofer, M.J., Herring, J.: A Mathematical Framework for the Definition of Topological Relationships.. In: *Int. Symp. on Spatial Data Handling*, pp. 803–813 (1990)
2. Clementini, E., Felice, P.D., Oosterom, P.: A Small Set of Formal Topological Relationships Suitable for End-user Interaction. In: *3rd Int. Symp. on Advances in Spatial Databases*, pp. 277–295 (1993)
3. McKenney, M., Pauly, A., Praing, R., Schneider, M.: Dimension-refined Topological Predicates. In: *13th ACM Symp. on Geographic Information Systems (ACM GIS)*, pp. 240–249 (2005)
4. Chen, J., Li, C., Li, Z., Gold, C.: A Voronoi-based 9-intersection Model for Spatial Relations. *International Journal of Geographical Information Science* 15(3), 201–220 (2001)
5. Schneider, M., Behr, T.: Topological Relationships between Complex Spatial Objects. *ACM Trans. on Database Systems (TODS)* 31(1), 39–81 (2006)
6. Egenhofer, M.J.: Topological Relations in 3D. Technical report (1995)
7. Zlatanova, S.: On 3D Topological Relationships. In: *11th Int. Conf. on Database and Expert Systems Applications (DEXA)*, p. 913 (2000)
8. Borrmann, A., van Treeck, C., Rank, E.: Towards a 3D Spatial Query Language for Building Information Models. In: *Proceedings of the Joint International Conference for Computing and Decision Making in Civil and Building Engineering* (2006)
9. Pigot, S.: Topological Models for 3D Spatial Information Systems. In: *International Conference on Computer Assisted Cartography (Auto-Carto)*, pp. 368–392 (1991)
10. Guo, W., Zhan, P., Chen, J.: Topological Data Modeling for 3D GIS. *Int. Archives of Photogrammetry and Remote Sensing* 32(4), 657–661 (1998)
11. Billen, R., Zlatanova, S., Mathonet, P., Boniver, F.: The Dimensional Model: a Framework To Distinguish Spatial Relationships. In: *Int. Symp. on Advances in Spatial Databases*, pp. 285–298 (2002)
12. Schneider, M., Weinrich, B.E.: An Abstract Model of Three-Dimensional Spatial Data Types. In: *12th ACM Symp. on Geographic Information Systems (ACM GIS)*, pp. 67–72 (2004)

Reasoning with Complements

Max J. Egenhofer

National Center for Geographic Information and Analysis
School of Computing and Information Science
University of Maine
Boardman Hall, Orono, ME 04469-5711, USA
max@spatial.maine.edu

Abstract. This paper develops a method to determine consistently the integration of spatial integrity specifications, particularly for the case when two constraints, c_1 and c_2 , are applied between three classes A, B, C (i.e., $A \subset B$ and $B \subset C$), as such scenarios give rise to an implied consistency constraint c_3 that holds between A and C. To determine c_3 , the mere composition reasoning with c_1 and c_2 is often insufficient, as it lacks consideration of the interferences that come through c_1 's and c_2 's complements. Two augmentation methods are introduced, one with a constraint's complement to yield the specified constraint between all pairs of members of A and C, the other as the least upper bound of c_1 and c_2 with the constraints' poset of possible transitions between constraints.

Keywords: Spatial reasoning, conceptual modeling, integrity constraints, topological relations, consistency.

1 Introduction

Associations between and aggregations of spatial objects are often specified by integrity constraints to ensure consistent semantics. The consideration of spatial relations as part of such integrity constraints offers a focus that is innate to the modeling and analysis of geospatial semantics. Such spatial relations may be applied at the instance level (e.g., Orono is inside the State of Maine) or in an ontology or a conceptual schema at the class level (e.g., islands are surrounded by waterbodies). The spatial relations used are typically *qualitative* spatial relations, which abstract away the myriad of quantitative detail, while they capture comprehensively some of the most critical traits of how individuals are related to each other. This paper focuses on the consistency of such spatial integrity specifications, particularly for the case when two constraints, c_1 and c_2 , are applied between three classes A, B, C (i.e., $A \subset B$ and $B \subset C$), as such scenarios give rise to an implied consistency constraint c_3 that holds between A and C.

The mere specification of a spatial relation at the class level is typically insufficient, as it does not quantify the relation. Therefore, associations are often supplemented with their cardinalities, specifying the relations as $n:m$ (with n and m typically taking on values of \mathbb{Z}^2). For relation specified between classes A and B, such cardinality specifications yield multiple ("each instance of class A may be

related to 0, 1, or many instances of class B”), singular (“each instance of class A has exactly one instance of class B”), or optional (“each instance of class A may be related to one instance of class B”) associations and aggregations. The use of such annotations has become standard practice in conceptual schema design for spatial applications [1,2,9,10] to model spatial properties at the class level, but without extensive quantification it often still lacks the ability to capture the intended semantics fully. A comprehensive classification of constraints on relations has identified a set of 17 abstract class relations [7,8], which apply to associations, specifying concisely how a specific binary relation (e.g., a topological relation) must be distributed among the participating objects in order to yield a valid representation. For instance, an island cannot exist without a surrounding water body, which requires that each island has a *surroundedBy* relation with respect to a waterbody (captured by a left-total specification). Such spatial integrity constraints may be nested. For instance, the additional constraint that any lake on an island must be *inside* a single island—a constraint that is left total and injective—implies that there is also the topological relation *surroundsWithoutContact* between the lake on the island and the waterbody in which the island is located. This paper addresses how to determine correctly not only the implied spatial relation, but also the implied abstract class relation.

The remainder of the paper is structured as follows: Section 2 briefly summarizes the 17 abstract class relations. Section 3 introduces the specification of the complement constraint and Section 4 includes complement constraint into the derivation of compositions. The paper closes with conclusions in Section 5.

2 Integrity Constraints and Topological Integrity Constraints

This paper bases the modeling of spatial integrity constraints on the results of two complementary models: (1) the abstract class relations to capture how instances of an association must be related in order to form a valid implementation of a conceptual data model, and (2) the binary topological relations between regions. The integrity constraint model builds on the work by Donnelly and Bittner [3], and Mäs [7,8]. Donnelly and Bittner introduced three base constraints to capture totality of a relation (Eqs. 1a-c) and a fourth constraint for non-total relations (Eqn. 1d).

$$R_{\text{some}}^{\text{D\&B}}(A,B) := \exists x \exists y (\text{Inst}(x,A) \wedge \text{Inst}(y,B) \wedge r(x,y)) \quad (1a)$$

$$R_{\text{all-1}}^{\text{D\&B}}(A,B) := \forall x (\text{Inst}(x,A) \rightarrow \exists y (\text{Inst}(y,B) \wedge r(x,y))) \quad (1b)$$

$$R_{\text{all-2}}^{\text{D\&B}}(A,B) := \forall y (\text{Inst}(y,B) \rightarrow \exists x (\text{Inst}(x,A) \wedge r(x,y))) \quad (1c)$$

$$R_{\text{all-all}}^{\text{D\&B}}(A,B) := \forall x \forall y (\text{Inst}(x,A) \wedge \text{Inst}(y,B) \rightarrow r(x,y)) \quad (1d)$$

Mäs [7] augmented this set with another two constraints, refining the non-total specification to address definiteness, such as the injective relation left-definite (Eqs. 2a and 2b).

$$R_{\text{Left-D}}(A,B) := \forall x,y,z (\text{Inst}(x,A) \wedge \text{Inst}(y,B) \wedge \text{Inst}(z,A) \wedge r(x,y) \wedge r(z,y) \rightarrow x=z) \wedge R_{\text{some}}^{\text{D\&B}}(A,B) \quad (2a)$$

$$R_{\text{Right-D}}(A,B) := \forall x,y,z (\text{Inst}(x,A) \wedge \text{Inst}(y,B) \wedge \text{Inst}(z,B) \wedge r(x,y) \wedge r(x,z) \rightarrow y=z) \wedge R_{\text{some}}^{\text{D\&B}}(A,B) \quad (2b)$$

In combination these six base constraints yield 17 abstract class relations as combinations of left-definite (LD), right-definite (RD), left-total (LT), and right-total (RT) (Figure 1). Although the naming conventions assigned may suggest otherwise, no two different constraints apply to a single configuration. For instance, if a constraint is classified as LD, then none of the other constraints that carry LD in their name (i.e., LD.RD, LD.LT, LD.RT, LD.RD.LT, LD.RD.RT, LD.LT.RT, and LD.RD.LT.RT) is possible. The universal consistency constraint—that is the exclusive disjunction of all 17 abstract class relations—is denoted by \mathcal{X} .

	$R_{\text{Left-D}}$	$R_{\text{Right-D}}$	$R_{\text{all-1}}^{\text{D\&B}}$	$R_{\text{all-2}}^{\text{D\&B}}$	$R_{\text{all-all}}^{\text{D\&B}}$	$R_{\text{some}}^{\text{D\&B}}$
LD	x	–	–	–		
RD	–	x	–	–		
LT	–	–	x	–		
RT	–	–	–	x		
LD.RD	x	x	–	–		
LD.LT	x	–	x	–		
LD.RT	x	–	–	x		
RD.LT	–	x	x	–		
RD.RT	–	x	–	x		
LT.RT	–	–	x	x	–	
LT.RT-all					x	
LD.RD.LT	x	x	x	–		
LD.RD.RT	x	x	–	x		
LD.LT.RT	x	–	x	x	–	
RD.LT.RT	–	x	x	x	–	
LD.RD.LT.RT	x	x	x	x	–	
some	–	–	–	–		x

Fig. 1. Specification of the 17 abstract class relations [7] in terms of six base constraints: “x” means a required base constraint, while “–” stands for a negative constraint. For instance, LD fully develops to $R_{\text{LD}}(A,B) := R_{\text{Left-D}}(A,B) \wedge \neg R_{\text{Right-D}}(A,B) \wedge \neg R_{\text{all-1}}^{\text{D\&B}}(A,B) \wedge \neg R_{\text{all-2}}^{\text{D\&B}}(A,B)$

The relations to which the abstract class relations are applied is the set of eight topological relations between two regions (Figure 2), which derives from the 4-intersection [6]. The universal topological relation as the disjunction of all eight relations is referred to by τ .

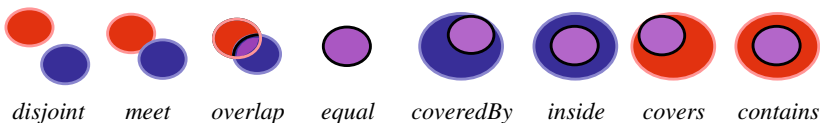


Fig. 2. The eight topological relations between two regions in \mathbb{R}^2 with the relations’ labels

Both relation models share at the meta-level a number of properties.

- The 17 abstract class relations and the eight region-region relations in \mathbb{R}^2 are *jointly exhaustive and pairwise disjoint*. This property implies that for any two instances—of a compatible type—exactly one of the 17 constraints and exactly one of the eight region-region relations applies. Therefore, any disjunction of integrity constraints or of topological relations is exclusive (denoted by \oplus).
- Both sets feature an *identity relation* (LD.RD.LT.RT and equal).
- In both sets, each abstract class relation and each topological relation has a *converse relation*. If converses map onto themselves the relations are symmetric.
- Both sets feature a complete *composition table* [4,8], which captures the result of combining two relations over the same object. None of the 17x17 and 8x8 compositions results in an empty relation.
- Both composition tables follow the axioms of a *relation algebra*, in particular the composition with the respective identity relation is idempotent, and the converses of compositions are compositions of converse relations, taken in reverse order.

3 Complements of Topological Integrity Constraints

If an association holds between two different classes A and B and the relation that links the two classes is from a set of relations that are jointly exhaustive and pairwise disjoint, then all instances of the two classes form a *complete bipartite graph*. In a bigraph the vertices can be divided into two disjoint sets A and B such that every edge connects a vertex in A with a vertex in B . A bigraph is complete if every vertex in A has an edge to every edge in B . Such a complete bigraph is specified by the integrity constraint LT.RT-all (Figure 3a). For instance, two simply-connected, hole-free regions embedded in \mathbb{R}^2 are always related by exactly one of the eight topological region-region relations [6]. Therefore, the universal topological relations applies by default as the constraint LT.RT-all (Figure 3b).

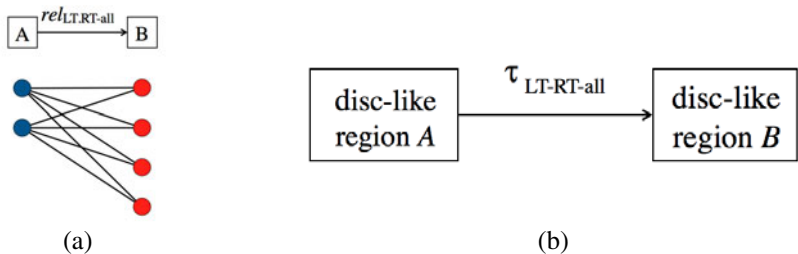


Fig. 3. (a) The complete bigraph formed by the LT-RT-all relation between all instances of class A and class B and (b) the constraint that any disc-like regions embedded in \mathbb{R}^2 must be related by exactly one of the eight topological region-region relations to another disc-like region B

Associations typically specify the entities’ relation only with respect to its host . For example, “building (*inside* \oplus *coveredBy*)_{RD,LT} landParcel” captures each

building's relation with the one land parcel that it is built on (excluding such special cases a building being partially built on a neighboring lot or a building's footprint extending over an entire land parcel). The integrity constraint RD.LT also captures that each building must have the relation with respect to a land parcel (because the constraint is left-total), that there are also land parcels that have no building on it (because the constraint is not right-total), and that a land parcel may have multiple buildings on it (because the constraint is not left-definite). It does not include, however, a specification of the relation and the integrity constraint with all other land parcels than the building's host. Such a relation exists, however between each building and each land parcel other than the building's host. For instance, if building $a1$ is *coveredBy* landParcel $b1$, and $b1$ *meets* land parcel $b2$, then building $a1$ and landParcel $b2$ would either be *disjoint* or they would *meet*. On the other hand, if building $a2$ is *inside* landParcel $b1$, then $a2$ would be *disjoint* from $b2$ without an option that $a2$ and $b2$ *meet*.

The *complement* of a topological integrity constraint captures the topological relations that must hold between all instances of the related classes other than the hosts. As a topological integrity constraint, the complement has a topological component and an integrity constraint. Its topological component follows from the relations' compositions [4]. For "building ($inside \oplus coveredBy$)_{RD,LT} landParcel" the topological complement is "building ($disjoint \oplus meet$)_{landParcel}." It applies to all buildings, therefore, the complement constraint must be left-total. Only if all buildings were *inside* their respective host parcel the topological complement would be unambiguously *disjoint*.

With respect to the land parcels, the integrity constraint is ambiguous as well, as three different scenarios may occur: (1) all buildings are on one land parcel, and there exists a second land parcel (without a building on it); (2) all buildings are on one land parcel, and there exist at least another two land parcels (each without a building); and (3) buildings are on more than one land parcel. In the first case the complement of RD.LT is RD.LT (Figure 4a), while in the second case RD.LT's complement is the mere left-total constraint LT (Figure 4b). For the third case the complement of RD.LT is left-total and right-total, yielding the constraint LT.RT. So the constraint "building ($disjoint \oplus meet$)_{LT,RT \oplus RD,LT \oplus LT} landParcel" is the complement of the constraint ($inside \oplus coveredBy$)_{RD,LT} (Figure 4c).

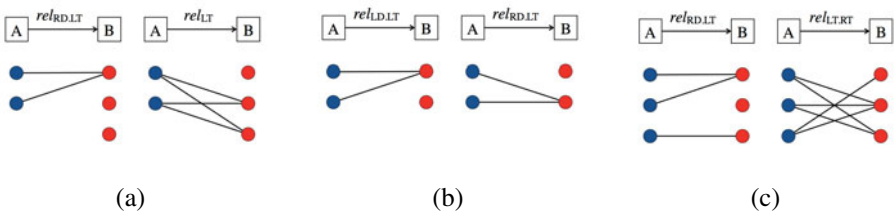


Fig. 4. Complement constraints for RD.LT: (a) LT, (b) RD.LT, and (c) LT.RT

In order to derive the complement for each of the 17 abstract class relations, the bigraph of each constraint was represented by an ordered pair of strings of the vertices' degrees [string1; string2]—string1 stands for the source, and string2 for the target. The sequence of the degrees within a string is immaterial. For instance, the two graphs in Figure 4b are captured by the two strings [1,1; 2,0] and [1,1; 0,2]. These two graphs are isomorphic, as captured by the same source strings and same target strings. For any of the 17 constraints, its complete bigraph serves as the base for determining its complement. The number of vertices in the source determines the degree of each vertex in the complete bigraph's target, while the number of vertices in the target determines the degree of each vertex in the complete bigraph's source. The complement of a constraint c , denoted by \bar{c} , is then the difference between the degree strings of its complete bigraph and the constraint's degree string. For example, the degree string of the constraint depicted in Figure 4a is [1,1; 2,0,0], and the degree string of its complete bigraph is [3,3; 2,2,2]. The complement of [1,1; 2,0,0] is then [3-1,3-1; 2-2,2-0,2-0], that is [2,2; 0,2,2].

The complements of all 17 abstract class relations (Figure 5) were derived from all respective bigraphs up to cardinality 5. No additional complements were found beyond cardinality 4. Completeness will be proven in the future.

c	\bar{c}	c	\bar{c}
LD.RD	LT.RT	RD.RT	LD.RD.RT, RD.RT, LT.RT
LD	LT.RT	LT	RD.LT,LT, LT.RT
RD	LT.RT	RT	LD.RT,RT, LT.RT
some	LT.RT	LD.RD.LT.RT	LD.RD.LT.RT, LT.RT
LD.RD.LT	LD.RD.LT, LD.LT, LT.RT	LD.LT.RT	LD.LT.RT, LT.RT
LD.RD.RT	LD.RD.RT, RD.RT, LT.RT	RD.LT.RT	RD.LT.RT, LT.RT
LD.RT	LD.RT, RT, LT.RT	LT.RT	$\mathcal{X} \setminus \text{LT.TR-all}$
RD.LT	RD.LT, LT, LT.RT	LT.RT-all	\emptyset
LD.LT	LD.RD.LT, LD.LT, LT.RT	\emptyset	LT.RT-all

Fig. 5. The complements \bar{c} to the 17 abstract class relations c (plus, for completeness, the empty constraint)

Four constraints (LD.RD, LD, RD, and some) have a unique, non-trivial complement. LT.RT-all (which forms the complete bigraph) has the empty graph as its complement. On the other end of the spectrum is complement to LT.RT, which is the universal constraint relation \mathcal{X} , except for LT-RT-all. The remaining eleven constraints' complements are each of cardinality 3, always including LT.RT.

4 Reasoning with Complements of Topological Integrity Constraints

The composition inferences over topological integrity constraints appear to be straight forward, given the compositions of abstract class relations [8] and the compositions of the binary topological relations [4]. Given two topological integrity constraints $t1_{c1}$ and $t2_{c2}$, where $t1$ and $t2$ are a topological region-region relation and $c1$ and $c2$ are

an abstract class relation, Mäs showed how to produce their composition as the Cartesian product of the class relation composition and the topological relation composition $(t1; t2)_{(c1; c2)} \leftarrow t1_{c1}; t2_{c2}$. This inference, however provides only a lower bound for the inferred topological integrity constraint as it unaccounts for any additional inferences that may result from the complements. For instance, calculating the composition $\text{meet}_{\text{some}}; \text{contains}_{\text{LD.RD.LT.RT}}$ as the mere Cartesian product of the the class relation composition and the topological relation composition yields $A \text{ disjoint}_{\text{some}} C$. There may be, however, additional instances of A and C that are disjoint without being related to another region through a meet-contains chain. For instance, if there is an region d such that $a_{\in A} \text{ disjoint } d$ and $d \text{ covers } c_{\in C}$, then there exists an additional relation between A and C , which may turn the constraint $A \text{ disjoint}_{\text{some}} C$ into the stronger constraint $A \text{ disjoint}_{\text{LT}} C$ or $A \text{ disjoint}_{\text{RT}} C$. The additional relation results from considering one of the complement relations. If additional disjoint relations are found, then the constraint could become even stronger, either $A \text{ disjoint}_{\text{LT.RT}} C$ or $A \text{ disjoint}_{\text{LT.RT-all}} C$.

4.1 Compositions with Complements

Accounting fully for the complement relations turns the calculation of the composition of two topological integrity constraints into a multi-step process. Two topological integrity constraints $t1_{c1}$ and $t2_{c2}$ yield the composition of the actual relations (Eqn. 3a) as well as the three compositions involving their complements $t1_{\bar{c1}}$ and $t2_{\bar{c2}}$ (Eqs. 3b-d).

$$t3_{c3} \leftarrow (t1; t2)_{(c1; c2)} \leftarrow t1_{c1}; t2_{c2} \quad (3a)$$

$$t4_{c4} \leftarrow (\bar{t1}; \bar{t2})_{(\bar{c1}; \bar{c2})} \leftarrow \bar{t1}_{\bar{c1}}; \bar{t2}_{\bar{c2}} \quad (3b)$$

$$t5_{c5} \leftarrow (t1; \bar{t2})_{(c1; \bar{c2})} \leftarrow t1_{c1}; \bar{t2}_{\bar{c2}} \quad (3c)$$

$$t6_{c6} \leftarrow (\bar{t1}; t2)_{(\bar{c1}; c2)} \leftarrow \bar{t1}_{\bar{c1}}; t2_{c2} \quad (3d)$$

There is no guarantee, however, the complement compositions always enhance the base inference. For example, the composition of $\text{meet}_{\text{some}}$ and $\text{contains}_{\text{LD.RD.LT.RT}}$, which yields $\text{disjoint}_{\text{some}}$, is complemented by three compositions with complements (Eqs. 4a-c). Since each topological complement inference results in this case in the universal topological relation τ , no conclusive inferences can be made.

$$\tau_{\text{LT.RT} \oplus \text{LT.RT-all}} \leftarrow ((\tau \setminus \text{meet}); (\tau \setminus \text{contains}))_{\text{LT.RT}; (\text{LD.RD.LT.RT} \oplus \text{LT.RT})} \leftarrow \overline{\text{meet}_{\text{some}}; \text{contains}_{\text{LD.RD.LT.RT}}} \quad (4a)$$

$$\tau_{\text{RD} \oplus \text{some} \oplus \text{RD.RT} \oplus \text{RT}} \leftarrow (\text{meet}; (\tau \setminus \text{contains}))_{\text{some}; (\text{LD.RD.LT.RT} \oplus \text{LT.RT})} \leftarrow \overline{\text{meet}_{\text{some}}; \text{contains}_{\text{LD.RD.LT.RT}}} \quad (4b)$$

$$\tau_{\text{LT.RT}} \leftarrow ((\tau \setminus \text{meet}); \text{contains})_{\text{LT.RT}; \text{LD.RD.LT.RT}} \leftarrow \overline{\text{meet}_{\text{some}}; \text{contains}_{\text{LD.RD.LT.RT}}} \quad (4c)$$

In case that all $a \in A$ form a partition of space, the topological complement of meet can only be disjoint (rather than $\tau \setminus \text{meet}$). This restriction yields one more constrained complement compositions (Eqs. 5c), while the other three complement

compositions remain inconclusive (Eqs. 5a and 5b). For the time being only conclusive complement composition inferences will be used. The role of inconclusive inferences will be explored in the future.

$$\tau_{LT,RT \oplus LT,RT\text{-all}} \leftarrow (\text{disjoint} ; (\tau \setminus \text{contains}))_{LT,RT ; (LD,RD,LT,RT \oplus LT,RT)} \quad (5a)$$

$$\tau_{RD \oplus \text{some} \oplus RD,RT \oplus RT} \leftarrow (\text{disjoint} ; (\tau \setminus \text{contains}))_{\text{some} ; (LD,RD,LT,RT \oplus LT,RT)} \quad (5b)$$

$$\text{disjoint}_{LT,RT} \leftarrow (\text{disjoint} ; \text{contains})_{LT,RT ; LD,RD,LT,RT} \quad (5c)$$

4.2 Constraint Augmentation with Complements

The next step is the attempt to augment a base inference (e.g., Eqn. 3a) with the result of a conclusive complement composition (e.g., Eqn. 5c). The topological integrity constraint $t3_{c3}$ is the base, and class relation $c3$ may be augmented if any of the three compositions with the complements results in the topological relation $t3$ as well (i.e., $t3=t4$ or $t3=t5$ or $t3=t6$). In such cases, the augmented class relation is then the result of combining the corresponding class relation with $c3$ (Eqn. 6).

$$c3 \text{ e } c_i \leftarrow t3 = t_{i,4..6} \quad (6)$$

For the composition of $\text{meet}_{\text{some}}$ with $\text{contains}_{LD,RD,LT,RT}$ the composition of the complement of $\text{meet}_{\text{some}}$ with $\text{contains}_{LD,RD,LT,RT}$ (Eqn. 5c) has the same resulting topological relation disjoint ; therefore, $\text{disjoint}_{\text{some}}$ may be augmented by $\text{disjoint}_{LT,RT}$. Since LT,RT is the complement of the class relation some (Figure 5), the augmentation of these two class relations ($\text{some} \text{ e } LT,RT$) yields $LT,RT\text{-all}$ for the topological relation disjoint (Eqn. 7).

$$\text{disjoint}_{LT,RT\text{-all}} \leftarrow \text{meet}_{\text{some}} ; \text{contains}_{LD,RD,LT,RT} \quad (7)$$

4.3 Constraint Augmentation from Transitions

The augmentation to $LT,RT\text{-all}$ is the strongest result of an integration as the resulting composition forms a complete bigraph. Other augmentations are also possible whenever the class relations over the same topological relation are not complements. Since an augmentation of the base relation essentially means the addition of at least one edge to the class relation’s graph one can derive all possible transitions that are due to the addition of one or more edges. Not all of the 17×17 possible transitions are feasible, however. For instance, one cannot add another edge to the graph of $LT,RT\text{-all}$, because it is already a complete bigraph. On the other hand, if one finds a transition from A to B, then the reverse transition from B to A will be infeasible with the addition of an edge.

Five conflicts govern which of the 289 transitions are impossible. While these conflicts are not mutually exclusive—several potential transitions respond to multiple conflicts—all five are necessary in order to derive the set of feasible transitions (i.e., those transitions that do not respond to any of the five conflicts).

- *Cardinality conflict:* The binary class relation $A2 \text{ c}2 \text{ B}2$ cannot result from the addition of a relation to the binary class relation $A1 \text{ c}1 \text{ B}1$ if $c1$ and $c2$ require different counts of instances. For example, $R_{LD,RD,LT,RT}$ requires that that the

cardinalities ($\#$) of A and B are the same, while $R_{RD.LT.RT}$ requires that $\#(A) > \#(B)$. Mäs [7] determined cardinality constraints for seven class relations (5, 6, 9, 10, 13, 14, 15), whose 49 combinations create 24 cardinality conflicts.

- *Definiteness-totality conflict*: Transitions from a relation that is a-definite.b-total ($a, b \in \{\text{left}, \text{right}\}$) cannot yield a relation that contains a-definite.b-total, because the definite property cannot be retained when adding an edge to it).
- *Definiteness increase*: The addition of a relation to a class relation that is not a-definite ($a \in \{\text{left}, \text{right}\}$) cannot turn that relation into an a-definite class relation.
- *Totality decrease*: The addition of a relation to an a-total class relation ($a \in \{\text{left}, \text{right}\}$) cannot turn it into a class relation that lacks the a-total property.
- *Complete-graph conflict*: LT.RT-all is a complete bigraph, so that the addition of another edge is impossible.

Out of the 289 transitions, only 80 are feasible, among them nine transitions between a class relation an itself (LD.RD, LD, RD, some, RD.LT, LD.LT, LT, RT, LT.RT). Such idempotent transitions have no effect on the augmentation of a derived base relation. Among the remaining 71 transitions are 44 atomic transitions (i.e., they cannot be obtained from successive combinations of other transitions) and 27 that can be obtained by two or more successive application of an atomic transition. The feasibility of all 44 transitions was confirmed by constructing for each transition two corresponding bigraphs that differ only by the addition of one edge. All feasible transitions are comprehensively captured by the transition graph (Figure 6), a directed conceptual neighborhood graph [5] in the form of a partially-ordered set, with LD.RD as the bottom element and LT.RT-all as the top element. The augmentation of a class relation with another one is then the *least upper bound* of these two relations within the transition poset (i.e., the least element that is common to the two class relations that participate in the augmentation). This property also shows that augmentation is commutative as $c \odot c_i$ equals $c_i \odot c$.

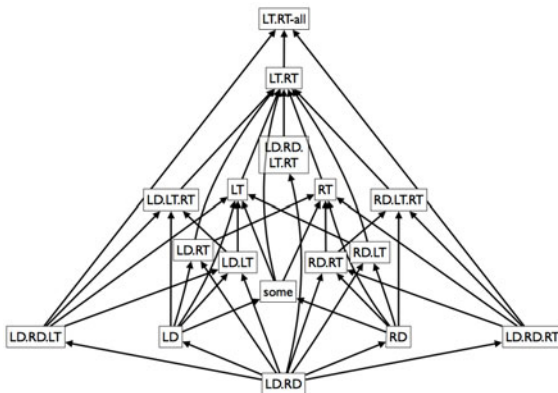


Fig. 6. The transition graph of the 17 class relations when adding an edge to a relation

5 Conclusions

In order to guarantee consistency in the specification with spatial integrity constraints, not only the soundness of compositions of must be considered, but also the

compositions with the constraints' complements. For Mäs's 17 abstract class relations [7] we derived their complements, and developed two methods of augmenting spatial integrity constraints. The first method, which applies when composition and complement composition are complementary, leads to the complete graph. For the second method that applies to any pair of the 17 abstract class relations, we developed the transition poset, a type of conceptual neighborhood graph that captures transitions between class relations when adding an additional edge to the bigraph of a constraint.

Acknowledgments. This work was partially supported by partially supported by the National Science Foundation under NSF grant IIS-1016740. Discussions with Stefan Mäs are gratefully acknowledged.

References

1. Belussi, A., Negri, M., Pelagatti, G.: An ISO TC 211 Conformant Approach to Model Spatial Integrity Constraints in the Conceptual Design of Geographical Databases. In: Roddick, J., Benjamins, V.R., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R.A., Grandi, F., Han, H., Hepp, M., Lytras, M.D., Mišić, V.B., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (eds.) ER Workshops 2006. LNCS, vol. 4231, pp. 100–109. Springer, Heidelberg (2006)
2. Borges, K., Laender, A., Davis, C.: Spatial Data Integrity Constraints in Object Oriented Geographic Data Modeling. In: Bauzer Medeiros, C. (ed.) 7th International Symposium on Advances in Geographic Information Systems, pp. 1–6. ACM, New York (1999)
3. Donnelly, M., Bittner, T.: Spatial Relations Between Classes of Individuals. In: Cohn, A.G., Mark, D.M. (eds.) COSIT 2005. LNCS, vol. 3693, pp. 182–199. Springer, Heidelberg (2005)
4. Egenhofer, M.: Deriving the Composition of Binary Topological Relations. *Journal of Visual Languages and Computing* 5(2), 133–149 (1994)
5. Egenhofer, M.: The Family of Conceptual Neighborhood Graphs for Region-Region Relations. In: Fabrikant, S.I., Reichenbacher, T., van Kreveld, M., Schlieder, C. (eds.) GIScience 2010. LNCS, vol. 6292, pp. 42–55. Springer, Heidelberg (2010)
6. Egenhofer, M., Franzosa, R.: Point-Set Topological Relations. *International Journal of Geographical Information Systems* 5(2), 161–174 (1991)
7. Mäs, S.: Reasoning on Spatial Semantic Integrity Constraints. In: Winter, S., Duckham, M., Kulik, L., Kuipers, B. (eds.) COSIT 2007. LNCS, vol. 4736, pp. 285–302. Springer, Heidelberg (2007)
8. Mäs, S.: Reasoning on Spatial Relations between Entity Classes. In: Cova, T.J., Miller, H.J., Beard, K., Frank, A.U., Goodchild, M.F. (eds.) GIScience 2008. LNCS, vol. 5266, pp. 234–248. Springer, Heidelberg (2008)
9. Pelagatti, G., Neri, M., Belussi, A., Migliorini, S.: From the Conceptual Design of Spatial Constraints to their Implementation in Real Systems. In: Agrawal, D., Aref, W., Lu, C.-T., Mokbel, M., Scheuermann, P., Shahabi, C. (eds.) 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, pp. 448–451. ACM, New York (2009)
10. Tryfona, N., Hadzilacos, T.: Logical Data Modeling of SpatioTemporal Applications: Definitions and a Model. In: Eaglestone, B., Desai, B., Shao, J. (eds.) International Database Engineering and Applications Symposium, pp. 14–23. IEEE Computer Society, Los Alamitos (1998)

Towards Modeling Dynamic Behavior with Integrated Qualitative Spatial Relations*

Stefan Mitsch, Werner Retschitzegger, and Wieland Schwinger

Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria
firstname.lastname@jku.at

Abstract. Situation awareness and geographic information systems in dynamic spatial systems such as road traffic management (RTM) aim to detect and predict critical situations on the basis of relations between entities. Such relations are described by qualitative calculi, each of them focusing on a certain aspect (e. g., topology). Since these calculi are defined isolated from each other, dependencies between them are not explicitly modeled. We argue, that a taxonomy—containing a plethora of special cases of inter-calculi dependencies—can only be defined in a consistent manner, if evolution of entities and the relations of calculi are grounded in a unified model. In this paper, we define such a unified model, which is used to derive a taxonomy of inter-calculi dependency constraints contained in an ontology utilizing various spatial calculi. The applicability of this approach is demonstrated with a case study in RTM, and concluded with lessons learned from a prototypical implementation.

1 Introduction

Situation Awareness in Dynamic Spatial Systems. Situation awareness and geographic information systems (GIS) are gaining increasing importance in dynamic spatial systems such as road traffic management (RTM). The main goal is to support human operators in assessing current situations and, particularly, in predicting possible future ones in order to take appropriate actions pro-actively. The underlying data describing real-world entities (e. g., tunnel) and their spatial relations (e. g., inside, near), which together define relevant situations (e. g., a traffic jam inside and near the boundary of a tunnel), are often highly dynamic and vague. As a consequence reliable numerical values are hard to obtain, which makes *qualitative modeling* approaches better suited than quantitative ones [17].

Dynamic Behavior in Qualitative Spatial Calculi. Recently, ontology-driven situation awareness techniques [1], [6] and qualitative approaches to modeling the dynamic behavior of spatial systems [3] have emerged as a basis for predicting critical situations from relations between objects. Such relations are expressed by employing multiple *relation calculi*, each of them focusing on a certain aspect, such as topology [8], [20], size [13], or distance [15]. These calculi

* This work has been funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under grant FIT-IT 829598.

are often temporalized by means of *Conceptual Neighborhood Graphs* (CNGs, [9]) and dominance spaces [10], [11], imposing constraints on the existence of direct transitions between relations. The domain-independent nature of calculi and their focus on a particular aspect of relationship (e.g., topology), however, results in dependencies between calculi being not explicitly modeled (e.g., topological transitions imply transitions in the distance between object boundaries). In a pioneering work on qualitative distance, Clementini stresses the importance of interdependency between calculi as follows: “the meaning of close depends not only on the actual relative position of both objects, but also [on] their relative sizes and other scale-dependent factors” [5].

A Unified Model for Inter-calculi Dependencies. Ontology-based approaches to dynamic spatial systems utilizing multiple calculi tackle the integration of these calculi by providing dedicated modeling primitives, for instance in terms of so-called *axioms of interaction* [3] and *relation interdependencies* [2]. Since these approaches, however, completely abstract from the underlying continuous space, a taxonomy exhaustively describing inter-calculi dependencies is still missing. We argue, that such a taxonomy—containing a plethora of special cases of inter-calculi dependencies—can only be defined in a consistent manner, if evolution of spatial primitives (e.g., regions) and the relations of calculi are grounded in a unified model. In order to define such a unified model for motion and scaling of spatial primitives and their effects in terms of transitions in topology, size, and distance, we base on Galton’s approach to constructing a “homomorphic image of the full space of possible region-pairs” [11].

Structure of the Paper. In the next section, the focus of this work is detailed as the basis for discussing relevant related work. In Sect. 3, a unified model for spatial calculi is presented along with a case study in the domain of RTM. The model is the basis for an ontology briefly sketched in Sect. 4. Finally, Sect. 5 concludes the paper with lessons learned from a prototypical implementation.

2 Related Work

In this section, we discuss related work on modeling the dynamic behavior of spatial systems with qualitative spatial reasoning approaches, focusing on those approaches in the domain of GIS. In this discussion, we follow the common ontological distinction (cf. the SNAP/SPAN approach [14]) often applied in GIS [12], [23] between the states of a system describing relations between entities from a snapshot point-of-view, and the evolution between these states in terms of occurments, such as events and actions. Causal relations between states and occurments [12] comprise (i) qualification constraints defining preconditions for states (i.e., states enable or disable other states, e.g., being smaller enables being a part), and for occurments (i.e., states allow or prevent occurments, e.g., having very close boundaries enables becoming externally connected), whereas

(ii) frame constraints define effects of occurments¹ (i. e., occurments cause other occurments, e. g., motion causes two objects becoming disrelated). In this paper, we focus on qualification constraints for states and occurments, since these are the primary source of inter-calculi dependencies.

Many qualitative spatial reasoning approaches (e. g., [7], [10], [11], [21]) provide or utilize a single qualitative calculus modeling a particular aspect, and naturally, encode qualification constraints in CNGs (i. e., each relation is a qualification constraint for its neighboring relations). A slightly broader view is applied in GIS [8], informally discussing states, in particular the size of objects, as qualification constraints for relations. The same constraint is used in a modeling framework for dynamic spatial systems [4] as qualification constraint on the transitions between relations. Arbitrary qualification constraints spanning multiple qualitative spatial calculi are explicitly supported in Bhatt's approach to modeling the dynamic behavior of spatial systems [3] in the form of so-called *axioms of interaction*. However, this modeling approach lacks a taxonomy of states and constraints. As a consequence, both must be provided by users of this modeling framework, instead of being integrated within its ontology.

Focusing on the integration of multiple calculi, Gerevini and Renz [13] discuss interdependencies between the Region Connection Calculus (RCC) and their Point Algebra for describing size relations. These interdependencies describe qualification constraints for states (i. e., relations) of one calculus in terms of states of the other. For example, a relation TPP (tangential proper part) of RCC entails a size relation $<$ (i. e., the contained entity must be smaller than the containing one). Using the same calculi (RCC and size), Klippel et al. [18] investigated the impact of different size relationships on the relation transitions in RCC induced by motion events, and the cognitive adequacy of these changes. Since the interdependencies between topological and size relations are rather obvious, providing a formal integration model, however, has not been the focus.

Clementini et al. [5] present several algorithms for combining distance and orientation relations from a compositional point-of-view (e. g., these algorithms compute the composition of distance relations, given a known orientation relation). In contrast, we focus on interpreting relations of a particular calculus as qualification constraints for relations and/or transitions in other calculi.

In summary, existing works lack a model of space and of spatial primitive pairs, preventing consistent integration of multiple calculi with major evolution causes (motion, scaling, orientation, shape, cf. [8]). In the next section, we discuss such a model along three spatial calculi modeling important aspects like topology (RCC, [20]), distance of boundaries [15], and size [13].

3 Inter-calculi Dependencies in Spatial Calculi

In qualitative spatial reasoning, as introduced above, a multitude of different spatial calculi has been proposed. Although each of these calculi focuses on a

¹ Occurrences initiating and terminating states (e. g., becoming very close initiates being very close) are not considered here, since they are modeled in CNGs.

particular aspect of the real world, some of their relations implicitly model other aspects as well (i. e., these relations restrict the relations that can hold and the transitions that can occur in another calculus). For instance, a topological non-tangential proper part relation (NTPP) between two objects does not only define that a particular object is contained in another one, but also implicitly defines that the contained object must be smaller than the containing one [13]. Additionally, real-world evolution abstracted to transitions in one calculus might be modeled in more detail in another calculus. For example, a topological transition from being disconnected (DC) to being externally connected (EC) in RCC is modeled from a distance viewpoint [15] with a sequence of relations and transitions, comprising transitions from being very far (VF) over far (F) and close (C) to being very close (VC). We make such assumptions explicit by combining existing calculi with qualification constraints modeling inter-calculi dependencies.

In order to define such qualification constraints in a consistent manner and account for a plethora of different special cases, a mapping between relations and the underlying spatial primitives including their numerical representation is needed. For example, let us consider relations describing the spatial distance between object boundaries. Since the boundary of an object implicitly defines its size and center, the options concerning the distance between the boundaries of two objects can only be narrowed by taking into account information about their topological relationship, relative size, and distance of their centers: If one object is known to be a proper part of the other one, a rather small object being located at the center of a large object is regarded to be very far from the large object’s boundaries, whereas the same object with a large distance to the center would result in the boundaries being considered to be very close. The boundaries of two nearly equally-sized objects would be considered very close as well.

As the basis for determining the above sketched variety of special cases making up inter-calculi dependencies, we base upon Galton’s approach [11] to deriving a two-dimensional image of relations from the CNG of RCC, since this approach covers the full space of possible region-pairs. In such a two-dimensional image, the topological relations between two spheres are encoded, using the radii r_1 and r_2 of the spheres along the x-axis ($x = r_1/(r_1 + r_2)$) and the distance d between their centers on the y-axis ($d/2(r_1 + r_2)$). The relations DC (disconnected), EC (externally connected), PO (partly overlapping), TPP (tangential proper part) and its inverse TPPi, NTPP (non-tangential proper part) and its inverse NTPPi, as well as EQ (equals) are defined in terms of these two measures in [1].

DC	$0.5 < y < 1$	EC	$y = 0.5$	(1)
PO	$ 0.5 - x < y < 0.5$	EQ	$x = 0.5 \wedge y = 0$	
TPP	$0 < y = 0.5 - x$	TPPi	$0 < y = x - 0.5$	
NTPP	$y < 0.5 - x$	NTPPi	$y < x - 0.5$	

The resulting image of possible relations in RCC between intervals in \mathbb{R} , circular regions in \mathbb{R}^2 , and spheres in \mathbb{R}^3 is depicted in Fig. 17. Besides reflecting

² This is different from Galton [11], since we normalize both the x- and y-axis metric with the sum of the radii to obtain a symmetric image.

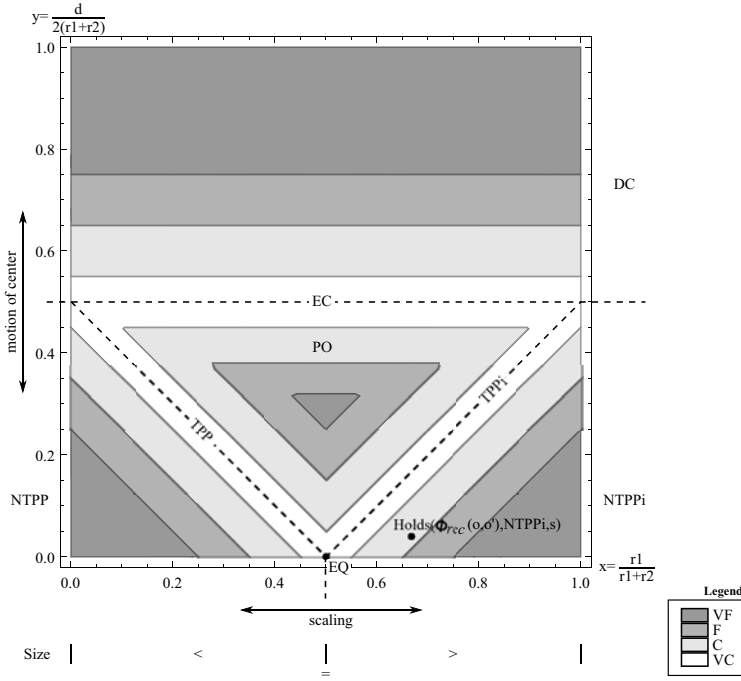


Fig. 1. Combined image of topology, distance of boundaries, and size (cf. [11])

the CNG of RCC (neighboring relations in the CNG are neighboring regions, lines, or points), this image encodes two interesting aspects of evolution: (i) the implications of motion and scaling, and (ii) the dominance relationship between relations (e.g., EQ being a point indicates that it can hold for a time instant, whereas those relations being denoted by regions must hold for a time interval).

Considering the impact of evolution, a point in this figure denoting a particular relation of RCC moves along the x-axis when one of the spheres changes its size with respect to the size of the other sphere (i.e., due to scaling), whereas its movement along the y-axis is caused by motion of the centers (which in turn is either due to motion of an entire sphere, or scaling). For example, consider the black dot labelled $Holds(\phi_{rcc}(o,o'), NTPPi,s)$ in the sector NTPPi, denoting that o contains o' : in terms of size and distance, this dot means that o is approximately twice the size of o' (cf. 0.67 on the x-axis), and that their centers are near each other. If o shrinks, the black dot moves along the x-axis to the left, until o can no longer fully contain o' , leading to an overlap relation (represented by the dot moving from NTPPi into PO). During this scaling, at a single time instant the boundary of o touches the boundary of o' , represented by the dot passing the line labeled TPPi (i.e., o' is a tangential proper part of o). If o shrinks even further, it will eventually be contained in o' (i.e., the dot will move into TPP). Now consider that the centers of o and o' coincide (i.e., their distance is zero): the same scaling event will then traverse EQ instead of TPPi, PO, and TPP.

We now define such a space representing relations as points for each of the employed positional relation calculi (distance of boundaries and size). To begin with, we discuss the integration of size, since the x-axis in Fig. 1 already expresses size relationships in terms of the ratio of interval radii $r_1/(r_1+r_2)$. The mapping to a qualitative size calculus is straightforward: a ratio below 0.5 corresponds to smaller ($<$), above 0.5 to larger ($>$) and one of exactly 0.5 to equal size ($=$).

Less obvious is the integration of the distance between boundaries. As a starting point, we informally define that two objects are very close whenever the boundaries meet, which is the case along the lines labeled EC, TPP, and TPPi, as well as at the point EQ. To both sides of these lines and around the point of topological equality, we define a region where the boundaries are still very close to each other (e.g., 10% off in distance and size as used in Fig. 1). Since we must consistently encode the CNG (represented by the sequence VF-F-C-VC), to each side of VC a region C must follow, which itself neighbors to regions F. Finally, regions VF are positioned at the outermost and innermost sectors of the image, neighboring only to regions F. Considering PO in conjunction with VC, it becomes obvious why our metrics are normalized. Let o be much larger than o' ($r_1 \gg r_2$) and o overlap with o' : their boundaries certainly should be regarded to be very close to each other, since in comparison to the size of o the distance between their boundaries is quite small (analogous assumptions hold for $r_1 \ll r_2$). This means, that our image should be symmetric with respect to size equality ($x = 0.5$), which cannot be achieved using an unnormalized metric. In (2) below, we define the distance relation VC with respect to $x = r_1/(r_1 + r_2)$ and $y = d/2(r_1 + r_2)$. With analogous formalizations, C, F, and VF can be defined.

$$VC \quad 0.45 < y \leq 0.55 \vee 0.45 - x \leq y \leq 0.55 - x \vee x - 0.55 \leq y \leq x - 0.45 \quad (2)$$

Case Study in the Domain of Road Traffic Management. We demonstrate the applicability of the integrated model by means of a hypothetical case study in the domain of road traffic management, which is made up of a situation evolution along various traffic entities, cf. Table 1. The entities are represented by traffic signs, and their spatial extent along the highway (direction from right to left) is indicated by surrounding boxes. The situation evolution comprises a traffic jam τ_j that starts growing due to capacity overload at a highway on-ramp onr in the middle of road works rwk . Shortly after, an accident acc occurs at the end of the traffic jam, which soon is contained within the further growing traffic jam. In order to reach the accident acc , an ambulance amb later passes through the traffic jam τ_j . In Table 1, the overall evolution of this situation is depicted as arrows representing evolution of relations in terms of their transitions in icons of the two-dimensional model introduced above. In order to represent traffic objects in our model, their occupied regions on a highway are modeled as intervals in \mathbb{R} . Next to each icon, Table 1 provides an informal description of the relation evolution between the entities. Summing up the case study, we have illustrated our approach by applying it to a scenario involving various different aspects

³ This measure has simply been chosen due to ease of presentation. It has neither been determined nor tested using cognitive studies.

Table 1. Case study of situation evolution in terms of relation transitions

Informal evolution description	Icon
<p>1. Traffic jam grows. In the beginning, the area of road works (1: $tj \leftrightarrow rwk$) is much larger than the traffic jam. Since the traffic jam grows, it thereafter extends beyond the area of road works, so causing transitions in topology, distance, and size. At the same time it remains externally connected to the on-ramp (2: $tj \leftrightarrow onr$).</p> <p>2. Accident occurs. Next, an accident occurs at the end of the traffic jam, further reducing traffic flow. Since the traffic jam (3: $tj \leftrightarrow acc$) is still growing, it soon completely contains the accident. In contrast, the accident and the area of road works are both stationary, resulting in no evolution between them (4: $rwk \leftrightarrow acc$).</p> <p>3. Ambulance drives towards accident. Finally, an ambulance drives towards the nearly-equally sized accident (5: $amb \leftrightarrow acc$), indicated by the arrow pointing downwards along the horizontal center and ending at EC). On its way to the accident, the ambulance enters the much larger traffic jam (6: $amb \leftrightarrow tj$). Thus, their boundaries are considered to become very far from each other even though the ambulance is within the traffic jam.</p>	

of evolution: (i) scaling in comparison to stationary objects with and without leading to relation transitions, (ii) non-evolution between two stationary, non-scaling entities, and (iii) motion of a non-scaling entity with respect to a scaling, and to a stationary, non-scaling one. The inter-calculi dependencies of Fig. 1 are extracted as qualification constraints into an ontology in the next section.

4 An Ontology of Inter-calculi Dependencies

Since we focus on the dynamic behavior of spatial systems, we express the inter-calculi dependencies summarized above in Fig. 1 in an ontology on the basis of the Situation Calculus [22] providing explicit support for modeling change between states in the form of occurrents. Change in the Situation Calculus is manifested in the properties of entities and the relations between them (e.g., a traffic jam's position can change, or its distance relation to an accident). In the terminology of the Situation Calculus, entities are *continuants* $O = \{o_1, o_2, \dots, o_n\}$, whereas their properties and relations to other entities in a particular situation are referred to as *fluents* $\Phi = \{\phi_1(o_1), \phi_2(o_2), \dots, \phi_n(o_n)\}$. We use in accordance with [3] *relational fluents* ϕ_r relating two continuants to each other using denotation sets $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$ and a ternary predicate *Holds* denoting that a fluent holds a particular value in a particular situation: for instance, $Holds(phi_{rccs}(o, o'), EQ, s)$ describes that the objects o and o' are topologically equal in situation s . Changed fluents are the result of *occurrents* $\Theta = \{\theta_1(o_1, \theta_2(o_2), \dots, \theta_n(o_n))$ [3]. Qualification constraints for occurrents can be defined using axioms of the form $Poss(\theta(o), s) \equiv Holds(\phi(o_1), \gamma, s)$, denoting that θ is possible in situation s , when a fluent ϕ of entities o_1 holds a particular value γ in s . For example, being of small size might be a precondition for growing, as stated by $(\forall o \in O)(\forall s \in S)Poss(grow(o), s) \equiv Holds(size(o), small, s)$.

Utilizing the Situation Calculus, we define qualification constraints for relation transitions (occurrents) on the basis of relational fluents, cf. Def. 1.

Definition 1 (Inter-calculi qualification constraint). *In accordance with [3] let transitions between relations be occursents $\text{tran}(\gamma, o, o')$, meaning that o and o' transition to the relation γ . An inter-calculi qualification constraint can then be formulated in the Situation Calculus as an action precondition axiom [22] of the syntactic form given in (3), meaning that a transition to γ_1 is possible, if a relation γ_2 of another spatial calculus currently holds between o and o' .*

$$(\forall o, o' \in O)(\forall s \in S) \text{Poss}(\text{tran}(\gamma_1, o, o'), s) \equiv \text{Holds}(\phi_{\text{spatial}}(o, o'), \gamma_2, s) \quad (3)$$

In Ex. 1 we provide a sample inter-calculi transition qualification constraint that formalizes the preconditions of the transition between DC and EC in terms of the states of relational fluents defining qualitative size and distance relationships.

Example 1. A transition from DC to EC in RCC is possible, if (trivially) DC from RCC holds, from a distance point-of-view VC holds, and from a size point-of-view any relation holds (summarized by the light-gray region VC that borders EC and spans all size relations in Fig. 1).

$$\begin{aligned} (\forall o, o' \in O)(\forall s \in S) \text{Poss}(\text{tran}(\text{EC}, o, o'), s) \equiv & \text{Holds}(\phi_{\text{rccs}}(o, o'), \text{DC}, s) \\ & \wedge \text{Holds}(\phi_{\text{dist}}(o, o'), \text{VC}, s) \wedge \text{Holds}(\phi_{\text{size}}, \gamma_1, s) \text{ where } \gamma_1 \in \{<, =, >\} \end{aligned} \quad (4)$$

As a proof-of-concept, we implemented the conceptual neighborhood structure of RCC, spatial distance of boundaries, and size, as well as the above-defined constraints in SWI-Prolog and used the FSA planner [16] implementing GOLOG (Reiter’s Situation Calculus programming language [22]) to synthesize sequential plans comprising the necessary relation transitions in order to reach a future goal situation from a current one. The lessons learned from this prototypical implementation and directions for further work are summarized below.

5 Critical Discussion and Further Work

Synthesized Plans Reflect Commonsense Understanding of Evolution.

The synthesized plans, without inter-calculi qualification constraints, reflect some implementation-dependent choice of the planner between independent transitions being possible at the same time (e.g., in our test runs, the order of transitions in the plan corresponded with the order of relations in the initial situation definition). Considering the additional inter-calculi qualification constraints, these transitions are no longer independent and, hence, the synthesized plans are consistent with commonsense understanding of the evolution of entities.

Generalization in Terms of Calculi and Spatial Primitives. Existing topological and positional calculi (e.g., Egenhofer’s approach [7]) can be integrated into the model by defining for each relation of the calculus a mapping to the x- and y-coordinate measures of our model. For example, the relation **inside** modeled as 4-intersection $\begin{pmatrix} -\emptyset & \emptyset \\ -\emptyset & \emptyset \end{pmatrix}$ describes a relation between two objects o and o' , where the intersection of the interiors of o and o' is not empty,

the intersection of the boundary of o and the interior of o' is not empty, whereas the intersection of the interior of o and the boundary of o' , as well as the intersection of their boundaries are empty. In terms of our model, for such a relation the distance between the centroids of o and o' must be smaller than the difference between their radii ($d < r_2 - r_1$), hence the following must hold true: $d/2(r_1 + r_2) < 0.5 - r_1/(r_1 + r_2)$ (i.e., **inside** is NTPP of RCC). In order to integrate additional spatial aspects not being representable with the spatial primitives employed above (e.g., orientation of entities towards each other), a generalization (e.g., in terms of higher-dimensional images) of the presented abstraction in terms of radii and center distance of spatial primitives is still necessary (e.g., considering orientation vectors). Likewise, in order to support the multitude of different spatial primitives found especially in GIS (e.g., regions, lines, points, as well as fuzzy approaches with broad boundaries) going beyond the intervals, regions, and spheres utilized above, metrics for comparing spatial primitives of different sorts must be defined (e.g., a line passing a region [7]).

Encoding of the Ontology with Semantic Web Standards. Since current Semantic Web standards, in particular OWL 2, formalize ontologies using a decidable fragment of first-order logic, an interesting further direction is to define a mapping of the ontology excerpt expressed in terms of the Situation Calculus into the concepts of OWL 2. For this, it can be based on prior work in terms of description logic rules [19] integrating rules and OWL. As a result, an integration with Semantic-Web-based GIS would be an interesting option.

References

1. Baumgartner, N., Gottesheim, W., Mitsch, S., Retschitzegger, W., Schwinger, W.: BeAware!—situation awareness, the ontology-driven way. *International Journal of Data and Knowledge Engineering* 69(11), 1181–1193 (2010)
2. Baumgartner, N., Gottesheim, W., Mitsch, S., Retschitzegger, W., Schwinger, W.: Situation Prediction Nets—Playing the Token Game for Ontology-Driven Situation Awareness. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) *ER 2010*. LNCS, vol. 6412, pp. 202–218. Springer, Heidelberg (2010)
3. Bhatt, M., Loke, S.: Modelling Dynamic Spatial Systems in the Situation Calculus. *Spatial Cognition and Computation* 8, 86–130 (2008)
4. Bhatt, M., Rahayu, W., Sterling, G.: Qualitative Simulation: Towards a Situation Calculus based Unifying Semantics for Space, Time and Actions. In: *Proc. of the Conf. on Spatial Information Theory*, Ellicottville, NY, USA (2005)
5. Clementini, E., Felice, P.D., Hernández, D.: Qualitative Representation of Positional Information. *Artificial Intelligence* 95(2), 317–356 (1997)
6. Cohn, A.G., Renz, J.: Qualitative Spatial Representation and Reasoning. In: *Handbook of Knowledge Representation*, pp. 551–596. Elsevier, Amsterdam (2008)
7. Egenhofer, M.: A Reference System for Topological Relations between Compound Spatial Objects. In: *Proc. of the 3rd Intl. Workshop on Semantic and Conceptual Issues in GIS*, Gramado, Brazil, pp. 307–316. Springer, Heidelberg (2009)
8. Egenhofer, M.: The Family of Conceptual Neighborhood Graphs for Region-Region Relations. In: *Proc. of the 6th Intl. Conf. on Geographic Information Science*, Zurich, Switzerland, pp. 42–55. Springer, Heidelberg (2010)

9. Freksa, C.: Conceptual neighborhood and its role in temporal and spatial reasoning. In: Proc. of the Imacs International Workshop on Decision Support Systems and Qualitative Reasoning, pp. 181–187 (1991)
10. Galton, A.: Towards a Qualitative Theory of Movement. In: Proc. of the Intl. Conf. on Spatial Information Theory: A Theoretical Basis for GIS. Springer, Heidelberg (1995)
11. Galton, A.: Continuous Motion in Discrete Space. In: Proc. of the 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning, Breckenridge, CO, USA, pp. 26–37. Morgan Kaufmann, San Francisco (2000)
12. Galton, A., Worboys, M.: Processes and Events in Dynamic Geo-Networks. In: Rodríguez, M.A., Cruz, I., Levashkin, S., Egenhofer, M.J. (eds.) GeoS 2005. LNCS, vol. 3799, pp. 45–59. Springer, Heidelberg (2005)
13. Gerevini, A., Nebel, B.: Qualitative spatio-temporal reasoning with RCC-8 and allen’s interval calculus: Computational complexity. In: Proc. of the 15th European Conf. on Artificial Intelligence, Lyon, France, pp. 312–316. IOS Press, Amsterdam (2002)
14. Grenon, P., Smith, B.: SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition & Computation: An Interdisciplinary Journal* 4(1), 69–104 (2004)
15. Hernández, D., Clementini, E., Felice, P.D.: Qualitative Distances. In: Kuhn, W., Frank, A.U. (eds.) COSIT 1995. LNCS, vol. 988, pp. 45–57. Springer, Heidelberg (1995)
16. Hu, Y., Levesque, H.J.: Planning with Loops: Some New Results. In: Proc. of the ICAPS Workshop on Generalized Planning: Macros, Loops, Domain Control, Thessaloniki, Greece (2009)
17. Ibrahim, Z.M., Tawfik, A.Y.: An Abstract Theory and Ontology of Motion Based on the Regions Connection Calculus. In: Proc. of the 7th Intl. Symp. on Abstraction, Reformulation, and Approximation, Whistler, Canada, pp. 230–242. Springer, Heidelberg (2007)
18. Klippel, A., Worboys, M., Duckham, M.: Conceptual Neighborhood Blindness—On the Cognitive Adequacy of Gradual Topological Changes. In: Proc. of the Workshop on Talking about and Perceiving Moving Objects: Exploring the Bridge between Natural Language, Perception and Formal Ontologies of Space, Bremen, Germany, Springer, Heidelberg (2006)
19. Krötzsch, M., Rudolph, S., Hitzler, P.: Description Logic Rules. In: Proc. of the 18th European Conf. on Artificial Intelligence, pp. 80–84. IOS Press, Amsterdam (2008)
20. Randell, D.A., Cui, Z., Cohn, A.G.: A Spatial Logic based on Regions and Connection. In: Proc. of the 3rd Intl. Conf. on Knowledge Representation and Reasoning. Morgan Kaufmann, San Francisco (1992)
21. Reis, R., Egenhofer, M., Matos, J.: Conceptual Neighborhoods of Topological Relations between Lines. In: Ruas, A., Gold, C. (eds.) Proc. of the 13th Intl. Symp. on Spatial Data Handling (2008)
22. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, Cambridge (2001)
23. Worboys, M., Hornsby, K.: From Objects to Events: GEM, the Geospatial Event Model. In: Egenhofer, M.J., Freksa, C., Miller, H.J. (eds.) GIScience 2004. LNCS, vol. 3234, pp. 327–343. Springer, Heidelberg (2004)

Transforming Conceptual Spatiotemporal Model into Object Model with Semantic Keeping

Chamseddine Zaki, Myriam Servières, and Guillaume Moreau

LUNAM Université, École Centrale Nantes, CERMA UMR CNRS 1563
{chamseddine.zaki,myriam.servieres,
guillaume.moreau}@cerma.archi.fr

Abstract. Our work is developed in the context of spatiotemporal data modeling and more particularly of urban data modeling. Our goal is to propose a methodology describing the overall process of urban data modeling from the design phase to the implementation in an information system. For that, we propose, within a GIS, an approach based on a conceptual model specific to spatiotemporal data modeling MADS, and on an ODBMS for storing and manipulating data. MADS uses spatial and temporal types that are more precise than those of programming languages on which are based ODBMS. MADS is semantically richer than the object model, and the purpose of this paper is to present the transformation rules of this conceptual model into an object model that keeps (as close as possible) its semantics.

Keywords: Spatiotemporal Modeling, Conceptual Models, ODBMS, Transformation of models, programming language, GIS.

1 Introduction

Our work takes part of a generic approach proposal to efficiently deal with urban data within a GIS. Currently, the major shortcomings of a GIS are in the spatiotemporal data design and in the incomplete coverage of the temporal dimension and multi-representation of data. For that, we propose a GIS based on the MADS conceptual model (Modeling of Application Data with Spatio-temporal features) [1], [2] and on an ODBMS (Object Database Management System) db4o (DataBase For Objects) [3] for storing and manipulating data.

MADS is a conceptual model dedicated to spatiotemporal data modeling [1]. It is based on the Entity-Relation (EA) formalism. However to take advantage of the semantics offered by the MADS model and to achieve its coupling with a GIS, we perform its transformation into an object model. For that, a key step in creating structures (generic classes) simulating the MADS concepts in the object language is necessary. This step is followed by the definition of mapping rules (algorithms) that allow the translation of specific conceptual schemas defined by MADS into the corresponding object language.

We emphasize that, in our approach, we are specifically interested in the use of ODBMS. ODBMS store the objects in Object Databases that are more flexible than

relational databases. In these ODBMS, data objects are treated the same way as in the programming language and access to data is by reference. With these models, we can store and access any type of data without using (or improving) specific manipulation languages like SQL. We believe that this gives advantages in the formulation and understanding of requests (indeed, the way in which the data will be handled is, in our case, planned by the conceptual schemas). Among the ODBMS we used db4o [3]. It is a free ODBMS based on the Java programming language that provides facilities for backing up any data type (even complex) in a uniform, automatic and transparent way. Db4o uses simple mechanisms for manipulating data based on methods of Java language. But on the other side the basic concepts offered by the programming language (on which the db4o is based) are less expressive than those of MADS. So semantic preservation verification methods are automatically implemented, on different levels, during the transformation from conceptual to physical model (programming language) to ensure the preservation of the semantics imposed by the designer when creating conceptual MADS schemas.

In other words, we propose the design of spatiotemporal applications to be made according to MADS schemas whose formal specifications, undergo later a series of systematic transformations towards operative object programs. The scripts generation, and the Object Database creation are viewed as sequences of the transformations.

In the rest of this paper, we will present in section 2 a background of conceptual spatiotemporal models. Then in section 3 we will present the general procedure for implementing our conception based on MADS into an object paradigm. We will detail the transition rules to express the thematic, spatial and temporal characteristics of MADS schemas into an object language. We will conclude this paper and give some perspectives of our work in the last section.

2 Background

Our state of art focuses on spatiotemporal data design models since they are, for our approach, the most important part that will monitor the processing of information. For this reason, we seek a semantically rich model that allows a good modeling of the spatiotemporal data and of applications. The implementation of this model and the adjustment of semantics during the implementation is the part of our approach that we will detail later in this article. A conceptual data model (especially for urban spatiotemporal data) must be able to offer users a wealth of expression to meet their diverse needs and also enable them to implement readable and easy to understand data schemes. In order to properly model spatiotemporal data, some requirements, besides those related to traditional data modeling, must be taken into account during the design phase. In fact, the conceptual model should allow modeling of spatial and temporal dimensions (modeling the objects and events that occur in a field of study, along with their interactions) as well as the temporal evolution of objects (history tracking) [2], [4].

Several spatiotemporal modeling approaches that take into account these requirements have already been proposed [5]. Most of them are based on modeling techniques like EA and Object-Oriented (OO) used for traditional data modeling.

Among OO techniques we can cite STUML "Spatio-Temporal UML" [6] and Perceptory [7], which are based on UML (Unified Modeling Language) [8]. Perceptory, for example, provides support (as icons) for spatiotemporal properties. Spatial and temporal properties can be used simultaneously, and can be assigned to classes and attributes. Its main weak point is that it does not assign spatial nor temporal properties to associations between the objects of a conceptual model.

Among the models of the EA approach we have MADS [1]. MADS allows the representation of real world entities as objects and associations. It supports orthogonality between the structural, spatial and temporal dimensions. MADS has interesting features like inheritance, effective temporal and historical data manipulation, and uniform handling of spatiotemporal data. It has intuitive and very readable visual notations to represent spatial and temporal concepts clearly, as well as a specific language to query the data model (For a more detailed description of MADS, the reader may refer to [2]). This model is considered as one of the best spatiotemporal design models in various studies ([3], [5], [9]). We decided to use MADS as the basic conceptual model for our approach.

However, MADS has been implemented with relational and object relational database [10] but has not been properly coupled to a GIS yet. Implementation attempts [11] consist of joint use of GIS (respectively ArcView and MapInfo) and a relational database (ACCESS), but these authors did not formally propose solutions to the problem of semantic loss in this implementation (e.g. ACCESS cannot handle intervals and temporal relations that are part of MADS concepts) [2]. We also precise that natively, MADS does not distinguish clearly between objects and events. For this purpose, we have proposed in [12], some improvements to this model in order to explicitly take into consideration the representation and the modeling of events as well as their interactions and their implementations in the object model.

For the implementation of MADS we used an ODBMS. ODBMS provide object persistence and take advantage of all the benefits of the object paradigm (frequently used in MADS). The addition of new data types to this category of DBMS is both easy and beneficial to our approach that is based on the semantically rich MADS model (whose specifications are not considered in any actual DBMS). Indeed, by using an ODBMS (from the second generation) as db4o for example, adding, saving and manipulating any type of data is done with a single language, without having to use nor extend any other data manipulation languages like SQL for instance.

3 Transformation Rules: Implementation of MADS in an Object Environment

To take advantage of the semantics of the MADS model we implement it into an object model using Java as a programming language. This is also justified by the fact that Java is the programming language for db4o. The transformation methodology of MADS in an OO framework is summarized as:

- Transformation of the generic conceptual model: This first phase consists on translating the spatial and temporal types of MADS into generic Java classes. For this reason, we exploit existing Java libraries and we create new classes when they

do not already exist. This first phase is to be done once and for all. The java classes created are called basic classes, and they will be the same no matter what is the considered case study.

- **Transformation of the conceptual schema:** While the first phase deals with the creation of basic classes, the second deals with the application schemas and consists in transforming each MADS conceptual schema into an equivalent Java program. This is achieved through the exploitation of the basic Java classes created in the first phase. In this second phase, for each given conceptual schema modeling a specific application, a corresponding structure (java program) is automatically created. The program resulting from this transformation is created using the rules of model transformation (described in the next paragraphs).

The orthogonality of the concepts and of the spatial and temporal dimensions of the MADS model allows us to conduct separate studies of transformation for these concepts and dimensions. Thus we begin by transforming the general concepts of MADS and then we present the transformation of spatial and temporal dimensions.

3.1 Transformation of General Concepts

MADS offers the usual concepts of extended entity-relationship models: classes, associations and generalization / specialization links. Classes and associations can have methods, attributes and primary keys. Attributes can be simple or complex (composed of other attributes) and they are characterized by a double cardinality (minimum, maximum) to distinguish the fact that they can be mono-valued or multi-valued (multiple values), optional or mandatory.

Transformation of Classes: The class is the fundamental concept of any object technology. This concept both exists in Java and in MADS. Our first transformation rule is to replace each class in the MADS conceptual schema MADS by a Java class.

Transformation of Attributes: The domain of an attribute is the collection of possible values for this attribute. It can be simple, object or complex type. MADS defines a set of simple data types (Float, Integer, Boolean, String, etc.) which can be assigned to attributes. These types will be directly converted into equivalent classes provided by the Java platform. Thus, a mono-valued attribute of simple type will be directly transformed into an attribute in java with the same name and with the type that matches his type declared in MADS.

Table 1 shows the transformation into Java of a “Building” MADS class containing several attributes of different types. In this example the attribute “number” is a mono-valued attribute of the simple type “Integer” that is transformed into a java attribute of the same name and type. In addition to simple data types, attributes in MADS can be objects of user-defined classes. The transformation of a mono-valued attribute of type “object” is similar to the transformation of an object of simple type. (See the transformation of the attribute “fireplace” in Table 1).

MADS allows the definition of complex attributes formed of a group of attributes. Sub-attributes, that are elements of the complex attribute, have the same characteristics of the direct attributes of the class and can be mono-valued or multi-valued, simple or complex, etc. In our proposal, any complex mono-valued attribute

becomes an instance of a new class declared inside the main class in order to simulate this attribute. Java allows creating classes within bounding classes. These inner classes will have a reference to the bounding class, and access to all its members (fields, methods, other inner classes). In addition, they cannot be declared outside the bounding class. (See the transformation of “roofing” in an inner-class in Table 1).

MADS allows an attribute to take more than one value and provides several types of collections to support this. MADS modeling of this multi-valued attribute is achieved by assigning a maximum cardinality equal to n . MADS’ defined collections are list, set and bag [2]. However, Java also offers the use of collections, and defines the Collection, Set, List and TreeList classes. Thus, we propose to establish a correspondence between the concepts: collection, set, bag and list of MADS and java basic classes: Collection, HashSet, ArrayList and TreeList, respectively. Subsequently, a MADS multi-valued attribute is translated into Java by an attribute having an equivalent collection type and same name (see the transformation of the attribute “owner” in Table 1). Although the maximum cardinality of an attribute indicates whether the attribute is multi-valued or mono-valued, the minimum cardinality specifies whether it is optional or mandatory. When the minimum cardinality is equal to zero then the attribute is optional, otherwise it is mandatory. In our proposal, the optional (mono or multi-valued) attributes of MADS are treated the same way as multi-valued attributes and are mapped to java collections. The only difference is that in this second case, additional constraints must be enforced. In fact, we must verify their existence right while accessing the data. If the attribute is optional and mono-valued, then another check is required (method created and called automatically) to insure before accessing the value of the attribute that the attribute is instantiated (see the transformation of the attribute “otherName” in Table 1).

Table 1. Examples of transformation of MADS attributes to Java

MADS conceptual schema	Java code (parts)																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: left; padding: 2px;">Building</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">number</td> <td style="padding: 2px;">Integer</td> <td style="padding: 2px;">1:1</td> </tr> <tr> <td style="padding: 2px;">fireplace</td> <td style="padding: 2px;">Fireplace</td> <td style="padding: 2px;">1:1</td> </tr> <tr> <td style="padding: 2px;">roofing</td> <td style="padding: 2px;">1:1</td> <td></td> </tr> <tr> <td style="padding: 2px;"> form</td> <td style="padding: 2px;">String</td> <td style="padding: 2px;">1:1</td> </tr> <tr> <td style="padding: 2px;"> surface</td> <td style="padding: 2px;">Integer</td> <td style="padding: 2px;">1:1</td> </tr> <tr> <td style="padding: 2px;"> tent</td> <td style="padding: 2px;">Integer</td> <td style="padding: 2px;">1:1</td> </tr> <tr> <td style="padding: 2px;">owner</td> <td style="padding: 2px;">String</td> <td style="padding: 2px;">1:n</td> </tr> <tr> <td style="padding: 2px;">otherName</td> <td style="padding: 2px;">String</td> <td style="padding: 2px;">0:1</td> </tr> </tbody> </table>	Building			number	Integer	1:1	fireplace	Fireplace	1:1	roofing	1:1		form	String	1:1	surface	Integer	1:1	tent	Integer	1:1	owner	String	1:n	otherName	String	0:1	<pre> public class Building { // constructor... Integer number; Fireplace fireplace; // method set (), get ()... Roofing roofing; class Roofing { String form; Integer tent; ... } List<String> owner = new ArrayList <String>(); List<String> otherName = new ArrayList <String>(); ... } </pre>
Building																												
number	Integer	1:1																										
fireplace	Fireplace	1:1																										
roofing	1:1																											
form	String	1:1																										
surface	Integer	1:1																										
tent	Integer	1:1																										
owner	String	1:n																										
otherName	String	0:1																										

Transformation of Relations between Classes. Different types of relationships may exist between MADS classes. In this article we only address the transformation of the most used relationships: association and generalization.

The association is a relationship linking two or many classes. Each class participating in an association plays a role. A line linking this class and the association models this role. It is characterized by a double cardinality that specifies the minimum and maximum number of relationships that an object of this class can have. An association relationship in MADS is transformed into a Java class (we have

chosen to make this correspondence since our target database is an object database and hence access to objects is done "by reference"). If the association contains attributes, then the transformation rules are the same as those of the "Class" concept. In addition to the attributes defined in the association, we add new attributes (or lists of attributes) to reference the object of classes involved in this association.

For the classes participating in the association, we also add references to the class that simulates the association. The minimum and maximum cardinalities of each role linking a class to the association specify the data structure we are going to use to store the references (to the instances of class denoting the association). This structure will be chosen in accordance with the rules used to transform an attribute with a minimum, and a maximum cardinality. In other words, for a class participating in an association, this association can be simulated by an attribute with a cardinality equal to the cardinality of the role that links the class to the association.

Table 2 shows the transformation of the association "Contains" linking "Plot" and "Building". The cardinalities of rules indicate that a plot may contain many buildings and a building is included in one plot. That will give us the java classes: Plot, Building and Contains. Building contains an attribute of type "Contains". Plot contains *n* records (grouped in collection of type ArrayList) to objects of class "Contains". "Contains" contains two references to "Plot" and "Building".

The "generalization relationship" is one of the fundamental concepts of object technology. It allows creation of specific classes that inherit attributes and methods of other classes. The concept of "generalization" between classes and associations of MADS is directly translated into the mechanism of inheritance between classes in Java. Table 2 presents an example of the transformation of this relation.

Table 2. Example of transformation of MADS relations to Java

MADS conceptual schema	Java code (parts)
<pre> classDiagram class Plot class Building class Contains class Public_building class Private_building Plot "1" --> "n" Contains Building "1" --> "1" Contains Building < -- Public_building Building < -- Private_building </pre>	<pre> public class Plot { List <Contains> contient = new ArrayList < Contains >(); ...} public class Building { Contains estContenu ; ...} public class Contains { Plot plot ; Building Building; ...} public class Public_builging extends Building{...} public class Private_builging extends Building{...} </pre>

3.2 Transformation of MADS Spatial Dimension

MADS offers spatial data types (Point, Line, etc.) [2] to denote the shape of objects. These spatial types can be assigned to classes, attributes and associations.

In Java, we took advantage of the existence of the spatial library "JTS" [13] which takes into account and enforce all the suggestions defined by the OGC (Open Geospatial Consortium) [14]. Nevertheless, the spatial types of MADS are more detailed than those of the JTS, but, for the moment, we still decided to make a direct

correspondence between the spatial types of MADS and the JTS classes. Indeed there are in MADS some types that have no equivalent in JTS, as for example "SimpleGeo" and "OrientedLine". In our transformation (Fig.1) we have not created classes for these types but we did match them with the closest JTS classes. (The creation of an exact equivalent of all MADS spatial types is feasible. It will be done by adding some new classes on JTS and it is one of our perspectives).

Type MADS		Classe JTS		Type MADS		Classe JTS
Geo	↔	Geometry		ComplexeGeo	↔	GeometryCollection
SimpleGeo	↔	Geometry		PointBag	↔	MultiPoint
Point	↔	Point		LineBag	↔	MultiLineString
Line	↔	LineString		OrientedLineSet	↔	MultiLineString
OrientedLine	↔	LineString		SurfaceBag	↔	MultiPolygon
Surface	↔	Polygon		SimpleSurfaceBag	↔	MultiPolygon
SimpleSurface	↔	Polygon				

Fig. 1. Correspondences between the spatial types of MADS and JTS classes

Nevertheless, the transformation of the spatial dimension in MADS schemas depends on whether this spatiality is assigned to the class, association or attributes.

Transformation of Spatiality for Classes and Associations: The spatiality is attributed to a class (or association) by adding it one of the MADS spatial types. To transform a spatial class in Java, the solution consists of transforming the spatiality of the class into a spatial mono-valued attribute called 'SpatialFeature'. This attribute will have as type the JTS class that corresponds to the spatiality of the MADS class.

Table 3 gives an example of transformation of the spatiality of class "Building". We note that the associations between two spatial classes can have topological constraints (contains, is contained, touches, overlaps, etc). A topological constraint is transformed into a static method that validates the constraint in the Java class simulating the association.

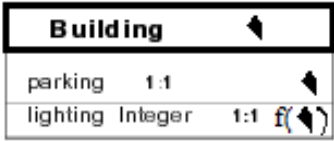
Transformation of the Spatiality for the Attributes: It is treated the same way as a thematic attribute of a not-simple type. The 'not-simple' type in this case, is the JTS class watching the spatial attribute type defined in the MADS conceptual schema.

In addition, MADS allows to describe continuous fields. A continuous field describes the values of an attribute over the geometry of an object type. Thus, the concept of "Space-varying attributes" are introduced in MADS and represented using the function $iconf()$ [2], [10]. Indeed, a space-varying attribute is an attribute whose value depends on a spatial location no matter its type (int, string, etc.) or its characteristics (mono-valued, multi-valued).

A space-varying attribute will be simulated as a multi-valued and complex object that contains two sub-attributes. The first is a spatial sub-attribute that can be of type point (when the domain of values to be calculated is continuous) or of type surface (in the other case). The second sub-attribute (of the same type as the space variable attribute) is used to retrieve the calculated value corresponding to the spatial location (first sub-attribute). For its transformation (see the transformation of attribute

“lighting” in Table 3), a space-varying attribute becomes an instance of a new class declared inside the main class in order to simulate this attribute. (This inner class contains a “Map” java structure that has the function domain as key and the type of the space-varying attribute as the value of this key. In addition the inner class contains some methods for adding and retrieving information).

Table 3. Example of transformation of MADS spatial concepts to Java

MADS conceptual schema	Java code (parts)
	<pre>import com. Vividsolutions .jts .geom.Polygon; public class Building { Polygon spatialFeature; Polygon parking; Lighting lighting ; class Lighting { private Map <Polygon, Integer> value ; // methods set() et get () ... } ... }</pre>

3.3 Transformation of Temporal MADS Dimension

MADS temporal types are used to design precise dates (instants, intervals, etc.), the lifecycle of objects or the temporal evolution of spatial and thematic attributes.

These temporal types can be assigned to classes, attributes and associations. They are organized in a temporal hierarchy (see [2]) that has no direct equivalent in the core classes of Java. Indeed, no existing Java time libraries can take into account the general complex and life cycle types of MADS. This is why we have created this structure and we have developed generic classes to simulate all the MADS temporal types. However, if powerful enough datatypes were introduced in the java language, it would be easy to use them.

The classes we have created are semantically equivalent and have the same names as MADS temporal types ("Instant", "Interval", "SimpleTime", etc.). Once these classes are created, the second step is to transform into Java the MADS conceptual schema using these temporal types. This transformation depends on the fact that temporality is assigned to the class, to the association or to the attributes.

Temporality for Classes and Associations: By assigning a temporal type to a class or an association, we keep track of the lifecycle of instances of this class or association.

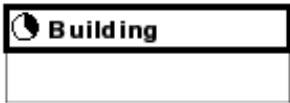
Tracking the lifecycle of instance is to indicate, its scheduling, creation, deletion or deactivation date. Each object can take at a given date, one of the following four statuses: planned, active, suspended or destroyed. The active period of an object is the temporal element associated to the active status in its life cycle [10].

The transformation of the temporality of a class or association (Table 4) is similar to the transformation of a complex multi-valued attribute named "LifeCycle" and having as sub-attributes:

- Attribute "status" which takes one of values: planned, active, suspended or destroyed
- A temporal attribute of the same type as the class (or association).

The associations between two temporal classes can have synchronized constraints (the active period of the first class is for example before, after, during, starts with, etc. the second). A synchronized constraint is transformed into a static method that validates the constraint in the Java class simulating the association.

Table 4. Example of transformation of MADS temporal concept to Java

MADS conceptual schema	Java code (parts)
	<pre>public class Building { List <lifecycle> lifecycle = new ArrayList <Lifecycle> ; class Lifecycle{ String status; Interval time; // verification methods... } ... }</pre>

Temporality at the Attributes Level: An attribute defined in a class (or association) of a MADS schema with a temporal type is translated into Java with an attribute of the generic java type corresponding to its temporal type.

In addition, the attributes in MADS, independently of their types (spatial, thematic, etc.) and their characteristics (mono-valued, multi-valued, optional, required) can be dynamic. In other words, the values of these attributes may change (possibly following a function) over time, and these changes will be saved. Changes in attribute values can be continuous, discrete or stepwise.

From a structural point of view, an attribute that varies over time is seen as a complex multi-valued attribute. Sub-attributes of this complex attribute are a "value" of the same type as the complex attribute, and a time element "time" of type "Interval" if the temporal variability is "stepwise" and of type "Instant" in the other cases ("continuous" and "discrete"). Thus, the transformation of a time-varying attribute is similar to the transformation of a space-varying attribute, but moreover, we add new methods to check constraints relative to this type (methods used to verify, for example, that the values of the attribute "time" are different if the type of time is "Instant", or that they are disjoint if the type of time is "Interval").

In order to test the capacity of our proposed modeling methodology in a real case, we have applied it on a case study whose objective was to analyze pedestrian walkways in urban areas. This example has provided answers to the problem of the influence of microclimate and comfort conditions on pedestrian behavior related to the practice of walking in urban areas. In fact we have created a conceptual schema to represent this case study and through the transformation roles of MADS conceptual schemas into object code, we were able to preserve the semantic richness of the original model and to highlight the ability of our model to manage and manipulate spatial and temporal queries. For a description of this example, the reader may refer to [12].

4 Conclusions

In this paper, we have presented our method for implementing MADS into Java. In fact, structural concepts and the spatial and temporal dimension of MADS are presented and implemented in structures directly manipulated by bd4o. Our

implementation allows preserving the semantic richness of the conceptual model as well as the direct access and storage of any type of data by db4o. Moreover, this transformation facilitates querying and visualizing objects from a GIS.

An important utility of our approach is the design template directed and framed the treatment. The conceptual scheme in our approach controls the data and adds a powerful framework to the methods with which data will be manipulated. In other words, our conceptual schemes rather model spatiotemporal applications. Then the implementation of the conceptual model in db4o allows having an object environment that ensures uniform storage and manipulation of data without the need to use or improve other data manipulation languages (like SQL).

As a perspective of this work, we aim to enrich the conceptual aspect MADS to enable the modeling of 3D spatial data as well as data with fuzzy or uncertain spatiality and temporality. Transformation of these concepts in Java is also planned.

References

1. Parent, C., Spaccapietra, S., Zimányi, E., et al.: Modeling /Spatial Data in the MADS Conceptual Model MADS. In: Spatial Data Handling 1998 Conference Proceedings, Vancouver, BC, Canada, pp. 138–150 (1998)
2. Parent, C., Spaccapietra, S., Zimányi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications: the MADS Approach. Springer, New York (2006)
3. Db4o, <http://www.db4o.com>
4. Zimanyi, E., Minout, M.: Preserving Semantics When Transforming Conceptual Spatio-temporal Schemas. In: Chung, S., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 1037–1046. Springer, Heidelberg (2005)
5. Pelekis, N., Theodoulidis, B., Kopanakis, I., Theodoridis, Y.: Literature review of spatio-temporal database models. Knowledge Engineering Review 19(3), 235–274 (2004)
6. Price, R.J., Tryfona, N., Jensen, C.S.: Extended SpatioTemporal UML: Motivations, Requirements and Constructs. Journal on Database Management, Special Issue on UML 11(4), 14–27 (2000)
7. Bedard, Y.: Visual Modeling of Spatial Databases Towards Spatial Extensions and UML. Geomatica 53(2), 169–186 (1999)
8. Fowler, M., Scott, K.: UML Distilled - Applying the Standard Object Modeling Language. Addison-Wesley, Reading (1998); ISBN 0-201-65783-X
9. Moïsuc, B., Gensel, J., Davoine, P.A.: Designing adaptive spatio-temporal information systems for natural hazard risks with ASTIS. In: Carswell, J.D., Tezuka, T. (eds.) W2GIS 2006. LNCS, vol. 4295, pp. 146–157. Springer, Heidelberg (2006)
10. Minout, M.: Modélisation des Aspects Temporels dans les Bases de Données Spatiales - thèse de doctorat: Université Libre de Bruxelles (2007)
11. Souleymane, T., DeSèdeMarceau, M.H., Parent, C.: COBALT: a design tool for geographic and temporal data application. In: Proceedings of the 6th AGILE Conference (2003)
12. Zaki, C., Zekri, E., Servières, M., Moreau, G., Hegron, G.: Urban Spatiotemporal Data Modeling: Application to the Study of Pedestrian Walkways. In: Intelligent Spatial Decision Analysis (ISDA 2010), Inner Harbor, Baltimore, Maryland, USA (2010)
13. JTS, <http://www.vividsolutions.com/jts/JTSHome.htm>
14. OGC, <http://www.opengeospatial.org/standards/sfa>

Preface to FP-UML 2011

The Unified Modeling Language (UML) has been widely accepted as the standard object-oriented language for modeling various aspects of software and information systems. The UML is an extensible language in the sense that it provides mechanisms to introduce new elements for specific domains; these include business modeling, database applications, data warehouses, software development processes, and web applications. Also, UML provides different diagrams for modeling different aspects of a software system. However, in most cases, not all of them need to be applied. Further, UML has grown more complex over the years, and new approaches are needed to effectively deal with these complexities. In general, we need heuristics and design guidelines that drive the effective use of UML in systems modeling and development.

The Seventh International Workshop on Foundations and Practices of UML (FP-UML'11) will be a sequel to the successful BP-UML'05 - FP-UML'10 workshops held in conjunction with the ER'05 - ER'10 conferences, respectively. The FP-UML workshops are a premier forum for researchers and practitioners around the world to exchange ideas on the best practices for using UML in modeling and systems development. For FP-UML'11, we received papers from nine countries: Poland, Bosnia and Herzegovina, France, Germany, Israel, Mexico, Spain, Tunisia, and United States. While the papers addressed a wide range of issues, the dominant topic was model-driven architectures, including the various challenges related to transformations and the complexities resulting from multi-model specifications.

The Program Committee selected three papers to include in the program. The first paper by Brdjanin and Maric shows how associations in class diagrams can be generated from activity diagrams. The second paper by Reinhartz-Berger and Tsoury compares two core asset modeling methods, Cardinality-Based Feature Modeling and Application-Based Domain Modeling, and discusses their benefits and limitations in terms of specification and utilization capabilities. Finally, the third paper by Marth and Ren introduces the Actor-eUML model for concurrent programming and formalizes the mapping between actors in the Actor model and Executable UML agents by unifying the semantics of actor behavior and the hierarchical state machine semantics of Executable UML agents.

We thank the authors for submitting their papers, the program committee members for their hard work in reviewing papers, and the ER 2011 organizing committee for all their support.

July 2011

Guido L. Geerts
Matti Rossi

On Automated Generation of Associations in Conceptual Database Model

Drazen Brdjanin and Slavko Maric

University of Banja Luka, Faculty of Electrical Engineering
Patre 5, 78000 Banja Luka, Bosnia and Herzegovina
{bdrazen,ms}@etfbl.net

Abstract. This paper considers the semantic capacity of object flows and action nodes in UML activity diagram for the automated generation of class associations in UML class diagram representing the conceptual database model. Based on the results of an analysis of action nodes regarding the number of different types of input and output objects as well as the weight of object flows, the formal transformation rules for the generation of object-object associations are defined and some experimental results of the corresponding ATL implementation are provided.

Keywords: Activity Diagram, Class Diagram, Conceptual Database Model, Transformation Rule, UML, ATL.

1 Introduction

The UML activity diagram (AD) is a widely accepted business modeling notation [1]. Several papers [2,3,4,5,6] take AD as the basis for (automated) conceptual data modeling, but with modest achievements in building the class diagram (CD) which represents the conceptual model. Emphasizing the insufficiently explored semantic capacity of AD for automated conceptual model design [5], these attempts have mainly resulted in (automated) generation of respective classes for extracted business objects [2,3,4,5,6] and business process participants [4,5,6], as well as a limited set of *participant-object* associations [4,5,6].

This paper considers the semantic capacity of object flows and action nodes in AD for automated generation of class associations in the target CD representing the initial conceptual database model (CDM). We performed an extensive analysis related to: (i) the nature of action nodes regarding the number of different types of input and output objects, and (ii) the weight of object flows. Based on its results, the formal transformation rules for the generation of *object-object* associations are defined and the experimental results of the corresponding ATL implementation, applied to a generic business model, are provided.

The rest of the paper is structured as follows. The second section introduces preliminary assumptions and definitions that will be used throughout the paper. In the third section we present the analysis of the semantic capacity of AD and define the formal transformation rules. ATL implementation is partly provided in the fourth section. The fifth section presents an illustrative example. Finally, the sixth section concludes the paper.

2 Preliminaries

2.1 Detailed Activity Diagram

We assume that each business process in the given business system is modeled by corresponding *detailed AD* (DAD) which represents activities at least at *complete* level (the related UML metamodel excerpt from the UML superstructure [7] is shown in Fig. 1 (up)), i.e.

- each step (action, activity, etc.) in the realization of the given business process is represented by a corresponding *action node* and will be shortly referred to as *activity* in the rest of the paper;
- each activity is performed by some business process participant (it can be *external*, e.g. buyer, customer, etc., or *internal*, e.g. worker, working group, organization unit, etc.) represented by a corresponding *activity partition*, usually called *swimlane*. In the rest of the paper, a business process participant is shortly referred to as *participant*;
- each activity may have a number of inputs and/or outputs represented by *object nodes*. In the rest of the paper they are referred to as *input objects* and *output objects*, respectively;
- objects and activities are connected with *object flows*. An object flow is a kind of *activity edge* which is directed from an input object toward the corresponding activity (*input object flow*) or from an activity toward the corresponding output object (*output object flow*); and
- each object flow has a *weight* attribute, whose value is by default one. The object flow weight represents the minimum number of *tokens* that must traverse the edge at the same time. We assume that constant weight represents not minimum, but the exact number of objects required for the activity if they are input objects, or the exact number of created objects in the activity if they are output objects. An unlimited weight (*) is used if the number of input/output objects is not constant.

Definition 1. Let $\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{F}$ be sets of participants, activities, objects and object flows in some business process, respectively. The **detailed activity diagram**, denoted by $DAD(\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{F})$, is an AD with the following properties:

- (1) each activity (a) is performed by only one participant (p), i.e.

$$\forall p \in \mathcal{P}, \forall a \in \mathcal{A} \mid a \in node(p) \Rightarrow inPartition(a) = \{p\};$$
- (2) each input object flow (if) is an object flow directed from exactly one input object (io) toward exactly one activity (a), i.e.

$$\forall io \in \mathcal{O}, \forall a \in \mathcal{A}, \forall if \in \mathcal{F}_{\mathcal{I}} \subseteq \mathcal{F} \mid if \in outgoing(io) \wedge if \in incoming(a) \\ \Rightarrow source(if) = \{io\} \wedge target(if) = \{a\};$$
- (3) each output object flow (of) is an object flow directed from exactly one activity (a) toward exactly one output object (oo), i.e.

$$\forall a \in \mathcal{A}, \forall oo \in \mathcal{O}, \forall of \in \mathcal{F}_{\mathcal{O}} \subseteq \mathcal{F} \mid of \in outgoing(a) \wedge of \in incoming(oo) \\ \Rightarrow source(of) = \{a\} \wedge target(of) = \{oo\}.$$

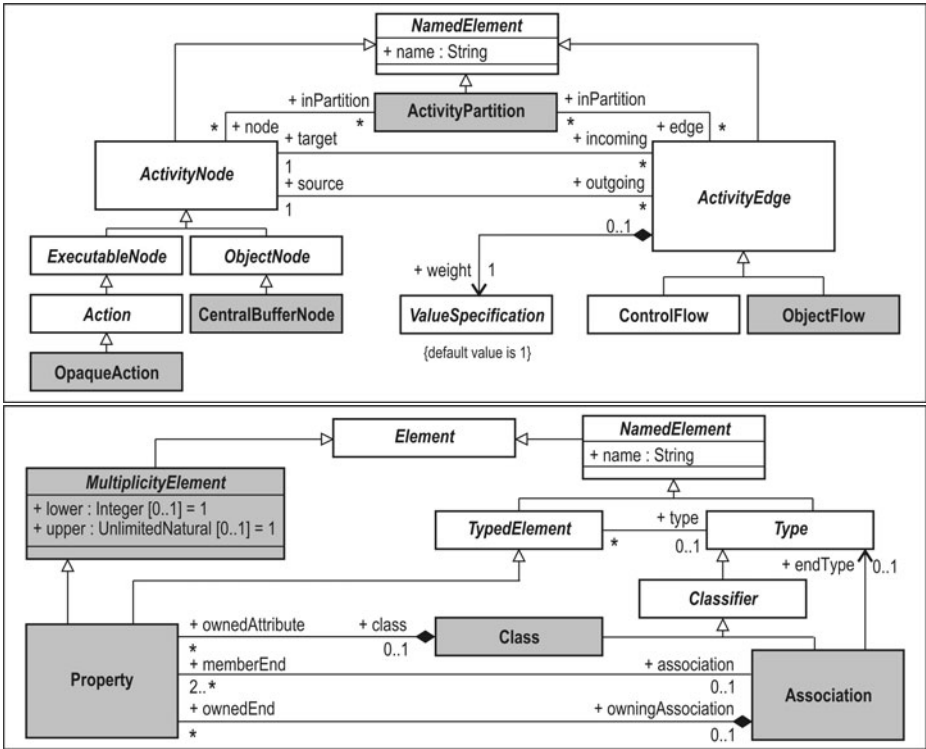


Fig. 1. UML metamodel excerpt used for representation of DAD (up) and CDM (down)

Definition 2. A *generated input object* (*gio*) is an input object created in the given process (object which also constitutes the output object from another activity in the same process), i.e.

$$\forall gio \in \mathcal{O}_G \subseteq \mathcal{O} \Rightarrow \exists of \in \mathcal{F}_O \mid target(of) = \{gio\}.$$

Definition 3. An *existing input object* (*eio*) is an input object which is not created in the given process but in some other process, i.e.

$$\forall eio \in \mathcal{O}_X \subseteq \mathcal{O} \Rightarrow \nexists of \in \mathcal{F}_O \mid target(of) = \{eio\}.$$

2.2 Conceptual Database Model

We use a CD to represent the CDM. The related UML metamodel excerpt from the UML infrastructure [8] is shown in Fig. 1 (down).

Definition 4. Let \mathcal{E} and \mathcal{R} be sets of classes and their associations, respectively. The *conceptual database model*, denoted by $CDM(\mathcal{E}, \mathcal{R})$, is a CD with the following properties:

- (1) each entity set is modeled by a corresponding class of the same name, whose each *ownedAttribute* (modeled by the *Property* metaclass) corresponds to an attribute of the given entity set, and

(2) each relationship is modeled by a corresponding class association of the same name, whose two *memberEnd* attributes (modeled by the *Property* metaclass) represent *source* and *target* association ends with the appropriate multiplicity corresponding to respective relationship cardinalities.

Proposed transformation rules in existing approaches for automated CDM design are illustrated in Fig. 2. According to [2,3,4,5,6] each *object* $o \in \mathcal{O}$ and each *participant* $p \in \mathcal{P}$ are directly mapped into corresponding classes $e_o \in \mathcal{E}_O$ (rule \mathcal{T}_O) and $e_p \in \mathcal{E}_P$ (rule \mathcal{T}_P), respectively. Hence, $\mathcal{E} = \mathcal{E}_O \cup \mathcal{E}_P$.

Existing approaches [4,5,6] define rules (denoted by \mathcal{T}_{PO}) only for automated generation of *participant-object* associations (associations between classes representing participants and objects) based on the activities performed on objects. Creation of *participant-object* associations (with cardinality "1:*") was suggested for each triplet $\langle \text{participant}, \text{activity}, \text{object} \rangle$. Making a distinction between existing and generated input objects implies that *participant-object* associations should be created not for all objects, but only for output objects and generated input objects. However, a deeper analysis and corresponding formal rules are outside the scope of this paper and will be focus of our further research.

In this paper we define the set of transformation rules (denoted by \mathcal{T}_{OO}) aimed at generating *object-object* associations.

3 Formal Rules for *object-object* Associations

As suggested in [3], associations between classes representing business objects could be generated based on activities having both input and output objects by direct mapping of such activities to the respective associations, but they don't explicitly propose any rule for the generation of multiplicities of association ends.

Before formally defining the transformation rules for automated generation of *object-object* associations, we will classify activities based on the number of different types of input and output objects. There are several possible situations (Fig. 3) and these are as follows: (i) *single input - single output (SISO)* activities, (ii) *multi input - single output (MISO)* activities, (iii) *single input - multi output (SIMO)* activities, and (iv) *multi input - multi output (MIMO)* activities.

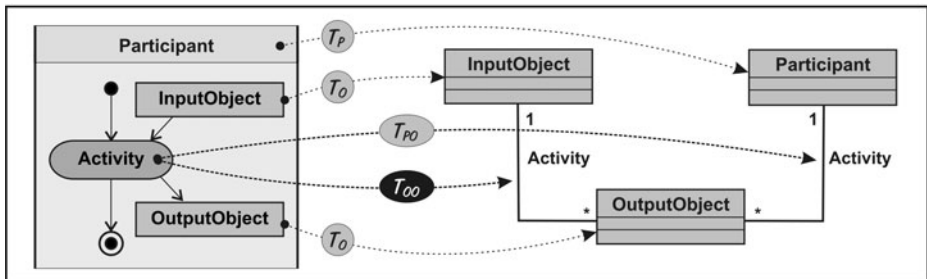


Fig. 2. Mapping of DAD into CDM

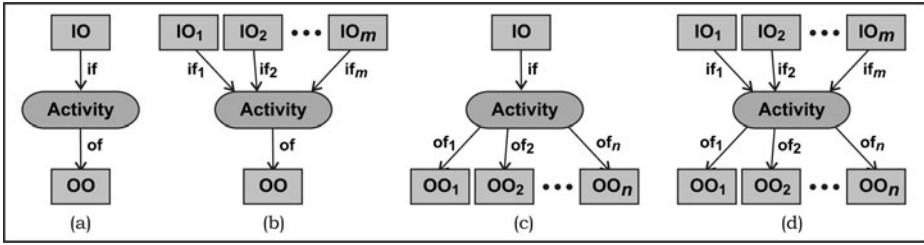


Fig. 3. Classification of activities: (a) SISO, (b) MISO, (c) SIMO, and (d) MIMO

SISO Activities. SISO cases and the corresponding transformation rules are illustrated in Fig. 4.

Regardless of whether the input object (IO) is existing (EIO) or generated (GIO), each output object (OO) depends on as many IOs as is the *weight* of the given input object flow. In this way, if the weight of the input object flow equals "1", then the OO depends on exactly one IO and the multiplicity of the respective source association end is exactly "1". If the input weight equals "*", then the OO depends on many IOs and the multiplicity of the respective source association end is "*". If the input weight is literal n which is greater than one, then the OO depends on exactly n IOs (like personal data containing data about two cities, where the first city represents the place of birth, while the second city represents the place of residence). In that case we have exactly n associations, where each association has the source end multiplicity equal to "1".

If the input object(s) is/are existing object(s), i.e. EIO, then the target end multiplicity (which corresponds to the OO) of each association always equals "*" and doesn't depend on the weight of the output object flow, because even

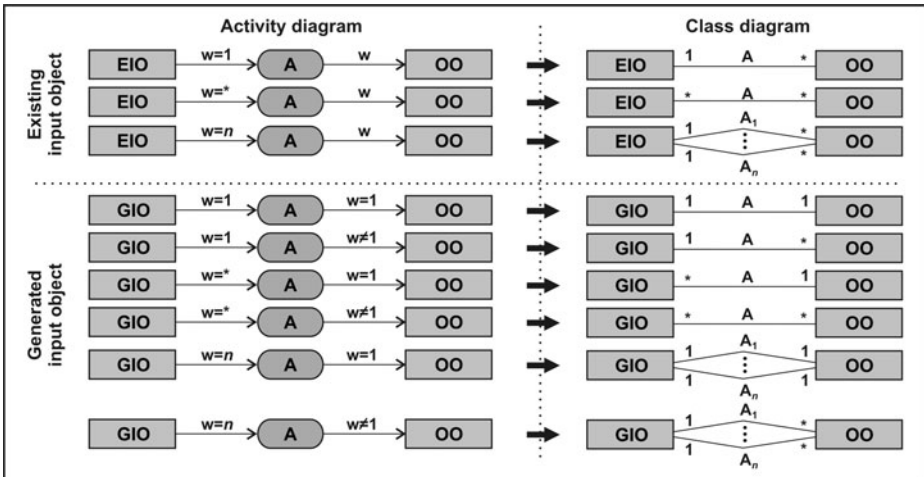


Fig. 4. Illustration of transformation rules for SISO activities

in cases when the output weight is exactly "1", with time, (the same EIO may be used in the creation of many OOs.

If the input object(s) is/are generated object(s), i.e. GIO, then the target end multiplicity depends only on the weight of the output object flow. If the output weight is exactly "1", then exactly one OO depends on input GIO(s) and the target end multiplicity should be exactly "1". Otherwise, the target end multiplicity should be "**", because more than one OO depend on input GIO(s).

Rule 1. (Object-object associations for SISO activities) Let $a \in \mathcal{A}$ be a SISO activity having input object(s) $io \in \mathcal{O}_G \cup \mathcal{O}_X$ and creating output object(s) $oo \in \mathcal{O}_O$, where $if \in \mathcal{F}_I$ and $of \in \mathcal{F}_O$ represent corresponding input and output object flows whose weights are denoted by w_{if} and w_{of} , respectively. Transformation rule T_{OO}^{siso} maps the SISO tuple $\langle io, if, a, of, oo \rangle$ into set $\mathcal{R}_{OO}^{siso}(a)$ containing exactly $n \in \mathbb{N}$ associations between classes e_{IO} and e_{OO} corresponding to the given input and output objects, respectively:

$$\begin{aligned}
 T_{OO}^{siso} : DAD(\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{F}) &\mapsto \mathcal{CM}(\mathcal{E}, \mathcal{R}) \stackrel{def}{\iff} \mathcal{R}_{OO}^{siso}(a) = T_{OO}^{siso}(\langle io, if, a, of, oo \rangle) \\
 \mathcal{R}_{OO}^{siso}(a) &= \left\{ r_{OO}^{(j)} \in \mathcal{R} \mid \forall j = 1, \dots, n \Rightarrow r_{OO}^{(j)} = T_{OO}^*(\langle io, if, a, of, oo \rangle) \right\} \\
 T_{OO}^*(\langle io, if, a, of, oo \rangle) &\stackrel{def}{=} r_{OO} \mid \left(name(r_{OO}) = name(a) \wedge \right. \\
 &\quad \left(memberEnd(r_{OO}) = \{ source, target \} \mid \right. \\
 &\quad \quad type(source) = e_{IO} \wedge multiplicity(source) = m_s \wedge \\
 &\quad \quad \left. type(target) = e_{OO} \wedge multiplicity(target) = m_t \right),
 \end{aligned}$$

where the corresponding source and target association end multiplicities and the total number of associations are as follows

$$m_s = \begin{cases} *, & w_{if} = * \\ 1, & otherwise \end{cases} \quad m_t = \begin{cases} *, & w_{of} \neq 1 \vee io \in \mathcal{O}_X \\ 1, & otherwise \end{cases} \quad n = \begin{cases} 1, & w_{if} \in \{1, *\} \\ w_{if}, & otherwise \end{cases} .$$

MISO Activities. Each MISO activity has input objects of more than one type and it creates output objects of exactly one type. Since all input objects are required for the start of the activity, we assume that the output object(s) depend(s) on all these input objects, i.e. output object(s) is/are directly related to each of the input objects. For example, an activity which results in creating an invoice takes data about sold goods and shipping details as well. This implies that SISO transformation rule T_{OO}^{siso} should be independently applied to each input object(s) - output object(s) pair of MISO activity.

Rule 2. (Object-object associations for MISO activities) Let $a \in \mathcal{A}$ be a MISO activity having $m \in \mathbb{N}$ different types of input objects $io_1, io_2, \dots, io_m \in \mathcal{O}_G \cup \mathcal{O}_X$ and creating the output object(s) $oo \in \mathcal{O}_O$, where $if_1, if_2, \dots, if_m \in \mathcal{F}_I$ and $of \in \mathcal{F}_O$ constitute the corresponding input and output object flows, respectively. Let $\mathcal{M}(a) = \{ \langle io_k, if_k, a, of, oo \rangle, 1 \leq k \leq m \}$ be the set of SISO tuples for the given activity $a \in \mathcal{A}$. Transformation rule T_{OO}^{miso} maps the $\mathcal{M}(a)$ set into the $\mathcal{R}_{OO}^{miso}(a)$ set of corresponding associations for the given activity:

$$\begin{aligned}
 \mathcal{T}_{\mathcal{O}\mathcal{O}}^{miso} : \mathcal{DAD}(\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{F}) &\mapsto \mathcal{CM}(\mathcal{E}, \mathcal{R}) \stackrel{def}{\iff} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{miso}(a) = \mathcal{T}_{\mathcal{O}\mathcal{O}}^{miso}(\mathcal{M}(a)) \\
 \mathcal{R}_{\mathcal{O}\mathcal{O}}^{miso}(a) &\stackrel{def}{=} \bigcup_{1 \leq k \leq m} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{(k)}(a), \quad \mathcal{R}_{\mathcal{O}\mathcal{O}}^{(k)}(a) = \mathcal{T}_{\mathcal{O}\mathcal{O}}^{siso}(\langle io_k, if_k, a, of, oo \rangle).
 \end{aligned}$$

The $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{miso}$ transformation rule is a general rule relevant for all *single output activities*, since a SISO activity is just a special case of MISO activities ($m = 1$).

SIMO Activities. A SIMO activity has input objects of exactly one type and creates output objects of more than one different types. Since the given activity is to result in the creation of all output objects, we assume that each output object is directly related to the given input object(s). This implies that SISO transformation rule $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{siso}$ should be independently applied to each *input object(s) - output object(s)* pair of a SIMO activity.

Rule 3. (Object-object associations for SIMO activities) Let $a \in \mathcal{A}$ be a SIMO activity having input object(s) $io \in \mathcal{O}_{\mathcal{G}} \cup \mathcal{O}_{\mathcal{X}}$ and creating $n \in \mathbb{N}$ different types of output objects $oo_1, oo_2, \dots, oo_n \in \mathcal{O}_{\mathcal{O}}$, where $if \in \mathcal{F}_{\mathcal{I}}$ and $of_1, of_2, \dots, of_n \in \mathcal{F}_{\mathcal{O}}$ represent the corresponding input and output object flows, respectively. Let $\mathcal{M}(a) = \{\langle io, if, a, of_k, oo_k \rangle, 1 \leq k \leq n\}$ be the set of SISO tuples for the given activity $a \in \mathcal{A}$. Transformation rule $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{simo}$ maps the $\mathcal{M}(a)$ set into the $\mathcal{R}_{\mathcal{O}\mathcal{O}}^{simo}(a)$ set of corresponding associations for the given activity:

$$\begin{aligned}
 \mathcal{T}_{\mathcal{O}\mathcal{O}}^{simo} : \mathcal{DAD}(\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{F}) &\mapsto \mathcal{CM}(\mathcal{E}, \mathcal{R}) \stackrel{def}{\iff} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{simo}(a) = \mathcal{T}_{\mathcal{O}\mathcal{O}}^{simo}(\mathcal{M}(a)) \\
 \mathcal{R}_{\mathcal{O}\mathcal{O}}^{simo}(a) &\stackrel{def}{=} \bigcup_{1 \leq k \leq n} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{(k)}(a), \quad \mathcal{R}_{\mathcal{O}\mathcal{O}}^{(k)}(a) = \mathcal{T}_{\mathcal{O}\mathcal{O}}^{siso}(\langle io, if, a, of_k, oo_k \rangle).
 \end{aligned}$$

MIMO Activities. A MIMO activity has $m \in \mathbb{N}$ different types of input objects and creates output objects of $n \in \mathbb{N}$ different types. Since direct application of the $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{siso}$ transformation rule to all SISO tuples would result in the creation of $m * n$ associations, it is recommended to transform the MIMO activity into a set of concurrent SIMO and/or MISO activities which could enable the application of the $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{simo}$ and/or $\mathcal{T}_{\mathcal{O}\mathcal{O}}^{miso}$ transformation rules, respectively. Such transformation could decrease the total number of associations generated for MIMO activities.

Therefore, assuming that all activities in the given DAD belong to MISO (including SISO as a special case) and/or SIMO activities, the total set of *object-object* associations is given with

$$\mathcal{R}_{\mathcal{O}\mathcal{O}} = \bigcup_{a \in \mathcal{A}^{simo}} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{simo}(a) \cup \bigcup_{a \in \mathcal{A}^{miso}} \mathcal{R}_{\mathcal{O}\mathcal{O}}^{miso}(a).$$

4 Implementation

We use **ATL**¹ [9] to implement formal transformation rules in the **Topcased** environment [10]. Due to the space limitations, implementation is just partly presented.

¹ ATLAS Transformation Language.

The main transformation rule, implemented as a *matched rule*, recognizes the context of each activity and invokes rules for creation of associations.

Object-object associations are created by invocation of *called rule* **SISO** (Listing 1) for each SISO tuple $\langle io, iof, ac, oof, oo \rangle$, where *io*, *iof*, *oof* and *oo* represent the corresponding input object, input object flow, output object flow and output object for the given activity *ac*, respectively. The proper *n*-arity of created associations is ensured by iteration through the sequence of literals recursively generated by *helper* `gC()`. For example, if the input weight equals "3", then the sequence will be $\{-1\}$ and only one association will be created, while in the case when the input weight equals "3", the sequence will be $\{3, 2, 1\}$ and exactly three associations will be created. Each *object-object* association is created by the invocation of the **TOO** called rule, which implements the \mathcal{T}_{OO}^* rule.

Listing 1. Implementation of transformation rule \mathcal{T}_{OO}^{siso}

```

helper def : gC (s:Sequence(Integer)) : Sequence(Integer) =
  if (s.last()>1) then thisModule.gC((s.append(s.last()-1))) else s endif;
rule SISO (io:uml!Element, iof:uml!Element, ac:uml!Element,
           oof:uml!Element, oo:uml!Element)
{
  using { ct:Sequence(Integer) = thisModule.gC(Sequence {iof.weight.value}); }
  do { for (c in ct) { thisModule.TOO(io,iof,ac,oof,oo); } }
}
rule TOO (io:uml!Element, iof:uml!Element, ac:uml!Element,
          oof:uml!Element, oo:uml!Element)
{ to
  d : uml!Association ( name<-ac.name, ownedEnd<-Sequence{os,od} ),
  os : uml!Property ( name<- 'source', type<-thisModule.resolveTemp(io,'d'),
                     upperValue<-us, lowerValue<-ls ),
  od : uml!Property ( name<- 'target', type<-thisModule.resolveTemp(oo,'d'),
                     upperValue<-ut, lowerValue<-lt ),
  us : uml!LiteralUnlimitedNatural
      (value<-if iof.weight.value=-1 then '-1'.toInteger() else 1 endif),
  ls : uml!LiteralInteger (value<-if iof.weight.value=-1 then 0 else 1 endif),
  ut : uml!LiteralUnlimitedNatural
      (value<- if (oof.weight.value<>1 or io.incoming.isEmpty())
               then '-1'.toInteger() else 1 endif),
  lt : uml!LiteralInteger
      (value<- if (oof.weight.value<>1 or io.incoming.isEmpty())
               then 0 else 1 endif)
  do { thisModule.cd.packagedElement <- d; }
}

```

5 Illustrative Example

The implemented generator has been applied to the generic DAD given in Fig. 5 (up). There are two participants (P1 and P2) in the given process. SIMO activity A, performed by P1, has one existing input object of the E1 type. Each execution of this activity results in the creation of two objects of the GA1 type and one object of the GA2 type. Both GA1 objects constitute the generated input objects in MISO activity B (performed by P2), which also has two existing input objects

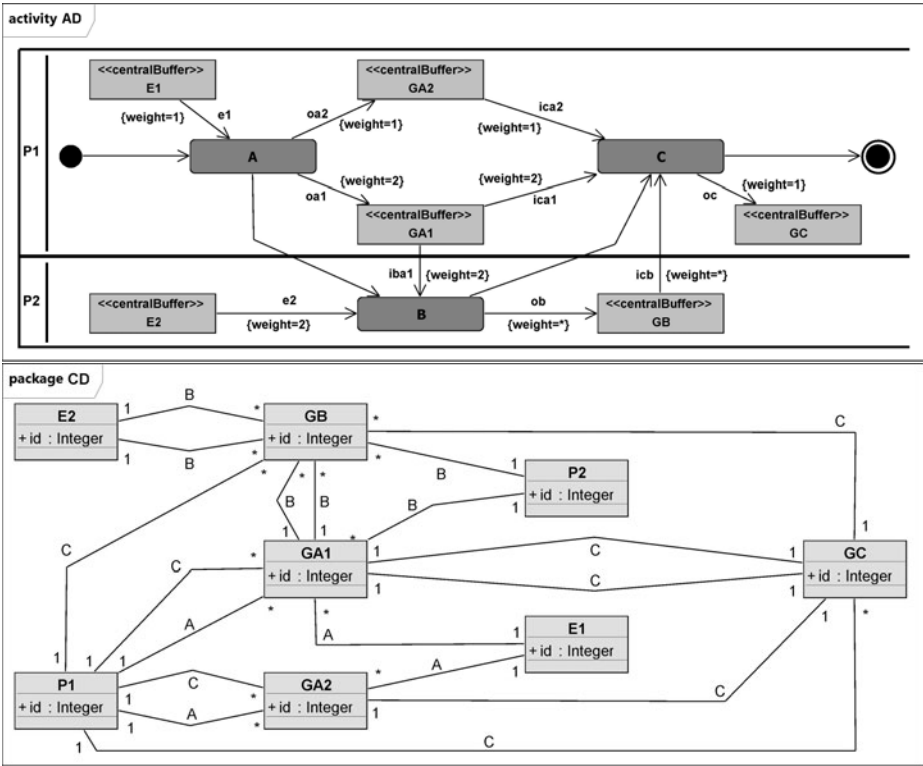


Fig. 5. Generic DAD (up) and corresponding automatically generated CDM (down)

of the E2 type. Each execution of activity B results in the creation of a number of GB objects and all of them, together with the GA1 and GA2 objects, constitute generated input objects in MISO activity C which creates one GC object.

After the execution of the ATL *module* and visualization in the Topcased environment, we have the CD representing the initial CDM given in Fig. 5 (down). The visualization result implies that the implemented generator has created all associations in accordance with the formal transformation rules.

The first group of associations are *object-object* associations created for all activities having existing input objects and output objects: $\langle E1, e1, A, oa1, GA1 \rangle$, $\langle E1, e1, A, oa2, GA2 \rangle$ and double association $\langle E2, ea, B, ob, GB \rangle$. The second group contains *object-object* associations created for all activities having generated input objects and output objects: double $\langle GA1, iba1, B, ob, GB \rangle$ association, double $\langle GA1, ica1, C, oc, GC \rangle$ association, $\langle GB, icb, C, oc, GC \rangle$ and $\langle GA2, ica2, C, oc, GC \rangle$.

Other associations are *participant-object* associations created for participants and output objects ($\langle P1, A, GA1 \rangle$, $\langle P1, A, GA2 \rangle$, $\langle P2, B, GB \rangle$ and $\langle P1, C, GC \rangle$), as well as for participants and generated input objects ($\langle P2, B, GA1 \rangle$, $\langle P1, C, GA1 \rangle$, $\langle P1, C, GA2 \rangle$ and $\langle P1, C, GB \rangle$).

6 Conclusion

This paper has considered the semantic capacity of object flows and action nodes in AD for automated generation of associations in CD representing the CDM. We have performed an analysis related to: (i) the nature of action nodes based on the number of different types of input and output objects, and (ii) the weight of object flows. By introducing the classification of activities into SISO, MISO, SIMO and MIMO activities and making a distinction between existing and generated input objects, we have defined formal transformation rules for generation of *object-object* associations. We have also provided some experimental results of corresponding ATL implementation applied to a generic business model.

In comparison with the few existing approaches, preliminary experimental results imply that proposed transformation rules significantly increase the already identified semantic capacity of AD for automated CDM design, since the existing approaches propose creation of associations for activities having both input and output objects, but don't propose any explicit rule for automated generation of association cardinalities.

Further research will focus on the full identification of the semantic capacity of AD for automated CDM design, formal definition of transformation rules (particularly for *participant-object* associations) and the evaluation on real business models.

References

1. Ko, R., Lee, S., Lee, E.: Business process management (BPM) standards: A survey. *Business Process Management Journal* 15(5), 744–791 (2009)
2. Garcia Molina, J., Jose Ortin, M., Moros, B., Nicolas, J., Troval, A.: Towards use case and conceptual models through business modeling. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) *ER 2000*. LNCS, vol. 1920, pp. 281–294. Springer, Heidelberg (2000)
3. Suarez, E., Delgado, M., Vidal, E.: Transformation of a process business model to domain model. In: *Proc. of WCE 2008, IAENG 2008*, pp. 165–169 (2008)
4. Brdjanin, D., Maric, S.: An example of use-case-driven conceptual design of relational database. In: *Proc. of Eurocon 2007*, pp. 538–545. IEEE, Los Alamitos (2007)
5. Brdjanin, D., Maric, S., Gunjic, D.: ADBdesign: An approach to automated initial conceptual database design based on business activity diagrams. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) *ADBIS 2010*. LNCS, vol. 6295, pp. 117–131. Springer, Heidelberg (2010)
6. Brdjanin, D., Maric, S.: Towards the initial conceptual database model through the UML metamodel transformations. In: *Proc. of Eurocon 2011*, pp. 1–4. IEEE, Los Alamitos (2011)
7. OMG: Unified Modeling Language: Superstructure, v2.2. OMG (2009)
8. OMG: Unified Modeling Language: Infrastructure, v2.2. OMG (2009)
9. Jouault, F., Allilaire, F., Bezivin, J., Kurtev, I.: ATL: A model transformation tool. *Science of Computer Programming* 72(1-2), 31–39 (2008)
10. TOPCASED Project: Toolkit in OPen-source for Critical Application & SysTEms Development, v3.2.0, <http://www.topcased.org>

Specification and Utilization of Core Assets: Feature-Oriented vs. UML-Based Methods

Iris Reinhartz-Berger and Arava Tsoury

Department of Information Systems,
University of Haifa, Haifa 31905, Israel
iris@is.haifa.ac.il, aravabt@gmail.com

Abstract. Core assets are reusable artifacts built to be used in different software products in the same family. As such, core assets need to capture both commonality that exists and variability that is allowed in the product family (line). These assets are later utilized for guiding the creation of particular valid products in the family. Feature-oriented and UML-based methods have been proposed for modeling core assets. In this work, we suggest a framework for analyzing and evaluating core assets modeling methods. We use this framework for comparing two specific methods: feature-oriented CBFM and UML-based ADOM. We found similar performance in modifying core assets in the two methods and some interesting differences in core assets utilization.

Keywords: variability, software product line engineering, domain analysis, UML, feature-orientation.

1 Introduction

Core assets are reusable artifacts built to be used in more than one product in the family (line) [1]. They are mainly utilized for creating particular *product artifacts* that satisfy specific requirements of software products (applications). Two commonly used ways to specify and model core assets are through feature-oriented [3, 5] and UML-based methods [4, 10]. While feature-oriented methods support specifying core assets as sets of characteristics relevant to some stakeholders and the relationships and dependencies among them, UML-based methods extend UML 2 metamodel or more commonly suggest profiles for handling core asset specification in different diagram types. In order to examine the capabilities of these kinds of modeling methods and their differences, we identified four main utilization and specification activities, namely (1) *guidance*, or reuse, which provides aids for creating product artifacts from core assets; (2) *product enhancement*, as exists while adding application-specific elements to satisfy the requirements in hand; (3) *product validation* with respect to the corresponding domain knowledge as specified in the core assets; and (4) *core asset modification*, which handles introducing changes or elaborations to existing core assets. All these activities refer to commonality and variability aspects of the given family of software products [11], [12]. *Commonality* mainly specifies the mandatory elements or features of the product line, i.e., the elements or the features that identify

the product family and all products that belong to that family must include them. However, commonality may refer also to optional elements which may add some value to the product when selected, but not all the products that belong to the family will include them. It can also refer to dependencies among (optional) elements that specify valid configurations. *Variability* [8] is usually specified in terms of variation points, which identify locations at which variable parts may occur, variants, which realize possible ways to create particular product artifacts at certain variation points, and rules for realizing variability (e.g., in the form of open and closed variation points, binding times, and selection of variants in certain variation points).

We examined how advanced information systems students, who studied a domain engineering course, performed the four aforementioned utilization and specification activities, separately referring to commonality and variability aspects. For this purpose, we used two particular methods: Cardinality-Based Feature Modeling (CBFM) [3], which is a feature-oriented method, and Application-based Domain Modeling (ADOM), which is defined through a UML 2 profile [10]. We chose these two particular methods due to their extended expressiveness in their categories [9] and we found that modification of core assets in the two methods results in quite similar performance. Regarding utilization, UML-based ADOM outperformed feature-oriented CBFM in commonality aspects of both guidance and product validation, while CBFM outperformed ADOM in product enhancement and variability aspects of product validation.

The remainder of this paper is organized as follows. Section 2 introduces our framework for analyzing the specification and utilization aids of core assets modeling methods, while Section 3 briefly introduces CBFM and ADOM. Section 4 describes the evaluation we have performed, discussing the research questions, settings, results, and threats to validity. Finally, Section 5 concludes and refers to future research directions.

2 The Core Assets Specification and Utilization Framework

Core assets and product artifacts rely in two separate layers: core assets, which are the basis for production or development of products in a product line, reside at a more abstract layer than product artifacts. Figure 1 shows these two layers, along with their artifacts and main activities. The core assets specification activities include creation and modification. Creation of core assets can be done by extracting knowledge from particular product artifacts, constructing analogy from other core assets (in parallel domains), or reviewing and studying the domain of interest. Modification takes core assets in a certain domain and introduces changes in order to enlarge the domain scope, specify newly discovered constraints, fix inaccuracies or errors, and so on.

A typical scenario of core assets utilization includes three steps. First, the core asset is used for guiding the creation of a particular product artifact. This step includes reusing the common kernel of the domain as captured in the core asset and choosing particular available variants. Then, the product artifact is enhanced, adding application-specific elements in order to satisfy the specific requirements of the product to be developed. Finally, the specification of the enhanced product is checked with respect to the core asset, in order to avoid violation of the domain constraints specified in the core asset or the accompanying production plan [2].

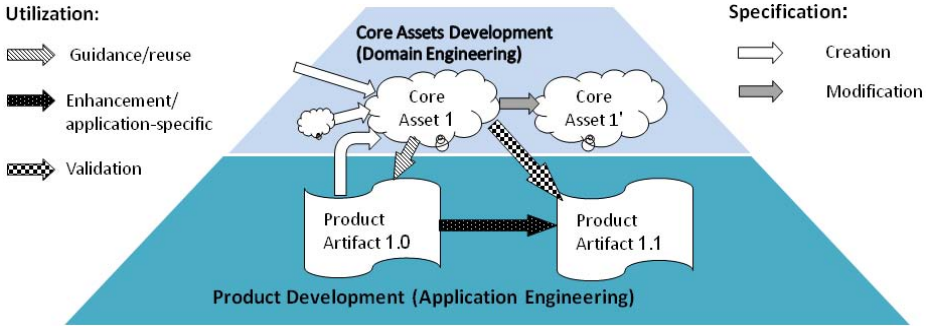


Fig. 1. A Framework for Analyzing Core Assets Specialization and Utilization

The above framework is used in the paper to examine core assets specification and utilization in two methods, which are presented next.

3 CBFM and ADOM

The *Cardinality-Based Feature Modeling (CBFM)* approach [3] builds, as other feature-oriented methods, on FODA's feature diagrams [5]. The nodes in these diagrams are features, i.e., end-user characteristics of systems or distinguishable characteristics of concepts that are relevant to some stakeholders of the concepts. Features can be decomposed into sub-features and the edges represent dependencies between features. Commonality is specified via mandatory and optional features or sub-features. Guidance is partially supported via XOR and OR constructs or via explicit textual constraints and composition rules. CBFM extends FODA's feature diagrams with five main aspects: (1) *cardinality*, which denotes how many clones of a feature can be included in a concrete product; (2) *feature groups*, which enable organizing features and defining how many group members can be selected at certain points; (3) *attribute types*, indicating that attribute values can be specified during configuration; (4) *feature model references*, which enable splitting a feature diagram into different diagrams; and (5) *OCL constraints*, which enable describing different types of relationships and dependencies between features. As feature-oriented methods in general and CBFM in particular do not explicitly support product enhancement, we used a blank feature with three dots as its name for specifying the ability to add application-specific elements in certain points (e.g., in "open" variation points that enable addition of application-specific variants).

Figure 2(a) is a part of a CBFM model of the Virtual Office (VOF) domain that describes peripherals. A peripheral, which is a device attached to a host computer and is dependent on the host, is characterized by its physical location and possibly by its model and manufacturer, all of which are of type String. Possible variants of peripherals are fax machines, printers, scanners, and others. A fax machine, for example, is characterized by a phone number, possible paper sizes, and possible produced image qualities.

A possible utilization of the above core asset is for creating a specific brokers' application, which has a single peripheral, called All-in-One that consists of a fax machine, a printer, and a scanner. The first step is choosing the mandatory and relevant optional paths in the feature diagram of the core asset. The only mandatory path in the VOF domain is that to Physical Location of Peripheral, but several relevant optional paths exist, e.g., the paths to Model, to Manufacturer, and so on. Application-specific features can be added in this domain only under the Peripheral feature group. Finally, in the third step of product validation, the product artifact, i.e., the brokers' application, is validated with respect to the core asset of the VOF domain. In particular, this step includes checking that the cardinality constraints are satisfied.

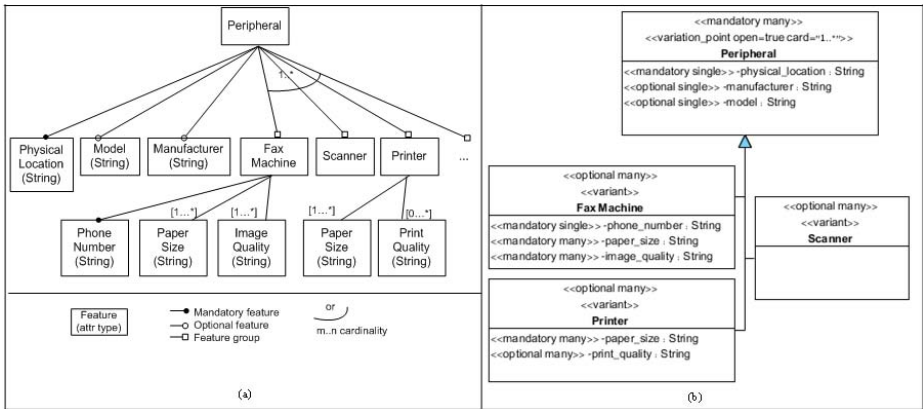


Fig. 2. Partial peripheral models in (a) CBFM and (b) ADOM (class diagram)

The *application-based Domain Modeling (ADOM)* defines a profile for specifying commonality and variability aspects of core assets [10]. Figure 3 depicts the main stereotypes and tagged values used in ADOM's profile, namely: (1) «multiplicity», which specifies the range of product elements that can be classified as the same core element¹, (2) «variation point», which indicates locations where variability may occur, including rules to realize the variability in these locations, (3) «variant», which refers to possible realizations of variability and is associated to the corresponding variation points, (4) «requires» and (5) «excludes», which determine dependencies between elements (and possibly between variation points and variants).

To exemplify the ADOM method, Figure 2(b) describes the peripheral concept of the VOF domain in terms of a class diagram. The peripheral is defined as a mandatory variation point with different possible variants, namely fax machines, printers, and scanners. When utilizing this core asset for creating particular product artifacts, like the brokers' application, the optional and mandatory elements from the core asset, as

¹ For clarity purposes, four commonly used multiplicity groups are defined on top of this stereotype: «optional many», where min=0 and max=∞, «optional single», where min=0 and max=1, «mandatory many», where min=1 and max=∞, and «mandatory single», where min=max=1. Nevertheless, any multiplicity interval constraint can be specified using the general stereotype «multiplicity min=m₁ max=m₂».

well as the relevant variants, are first selected and adapted to the application in hand. The adaptation starts with providing these elements with specific names that best fit the given application, besides the domain names that appear as stereotypes in the application model. Note that the same specific (application) element may be stereotyped by several core asset elements to denote that it plays multiple roles in the domain. All-in-One, for example, will be simultaneously stereotyped as Fax Machine, Printer, and Scanner. Nevertheless, when selecting particular variants in a certain variation point, the name of the variation point does not appear as a stereotype. Thus, All-in-One will not be stereotyped as Peripheral, but just as Fax Machine, Printer, and Scanner. Then, the application model is enhanced by adding application specific elements, which can be completely new elements or elements that are created from (open) variation points without reusing particular variants. Respectively, completely new elements are visualized as elements without stereotypes, while variation point-derived elements are decorated with the variation point names as their stereotypes. In any case, these elements cannot violate the domain constraints, which are validated using the application stereotypes as anchors to the domain.

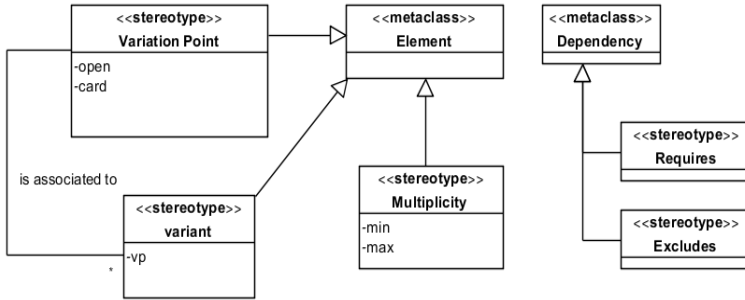


Fig. 3. The UML profile at the basis of ADOM

Note that there is a very significant difference between the utilization of core assets in CBFM and their utilization in ADOM. While CBFM reuses core assets mainly by configuring them to the particular needs of a given application, ADOM enables specialization of core assets and addition of details in the model level (denoted M1 in the MOF framework [7]). However, the usage of multiple diagram types in ADOM may raise consistency and comprehension difficulties that need to be examined. The next section described a preliminary experiment in this direction.

4 Experimenting with CBFM and ADOM Specification and Utilization

In order to experiment with the capabilities of CBFM and ADOM to specify and utilize core assets, we gave advanced undergraduate and graduate information systems students at the University of Haifa, Israel core assets of the VOF domain in either CBFM or ADOM. We asked the students to perform some modification tasks

on the models and to create models of specific product artifacts according to given requirements. We did not refer to core assets creation in our study, due to the nature of our subjects and the complication of such task that mainly originates from the diversity of the required sources. Instead, we contented with modification tasks for examining core assets specification. Nevertheless, while creating the particular product artifacts, the students were implicitly asked to reuse portions of the core assets and to add application-specific elements. They were also asked to list requirements that cannot be satisfied in the given domain, since they somehow violates the core assets specification. Respectively, we phrased the following four research questions: (1) The specifications of which method are more modifiable (i.e., easy to be modified) and to what extent? (2) The specifications of which method are more guidable and to what extent? (3) The specifications of which method help enhance particular product artifacts and to what extent? (4) The specifications of which method help create valid product artifacts and to what extent?

4.1 Study Settings

Due to the difficulties to carry out such experiments on real developers in industrial settings [6], the subjects of our study were 18 students who took a seminar course on domain engineering during the winter semester of the academic year 2010-2011. All the subjects had previous knowledge in systems modeling and specification, as well as initial experience in industrial projects. During the course, the students studied various domain engineering techniques, focusing on CBFM and ADOM and their ability to specify core assets and utilize them for creating valid product artifacts. The study took place towards the end of the course as a class assignment, which worth up to 10 points of the students' final course grades.

The students were divided into four similarly capable groups of 4-5 students each, according to their knowledge in the studied paradigms (feature-oriented vs. UML-based), previous grades, and degrees (bachelor vs. master students). Each group got a CBFM or ADOM model of the entire VOF domain and a dictionary of terms in the domain. The ADOM model included a use case diagram and a class diagram, while the CBFM model included two related feature diagrams (using feature model references). Two groups got modification tasks, while the two other groups got utilization tasks (see the exact details on the experiment setting in Table 1).

Table 1. Experiment design

Group #	Method	Task	# of students
A	CBFM	Modification	5
B	ADOM	Modification	5
C	CBFM	Utilization	4
D	ADOM	Utilization	4

As noted, the modification tasks mainly required extensions to the core asset and not inventing new parts from scratch. An example of a modification task in the study is:

For checking devices two strategies are available: distance-based and attribute-based. In the distance-based strategy the application locates the nearest available

devices for the task in hand ... In the attribute-based strategy, the employee needs to supply values to different relevant parameters of the device, such as the paper size and the print quality for printers and the paper size and the image quality for fax machines. Each application in VOF domain must support the distance-based strategy, but may support both strategies.

The utilization tasks required creating valid portions of a brokers' application in the domain, according to a predefined list of requirements, and listing the parts of the requirements that cannot be satisfied with the given core assets. An example of requirements in these tasks is:

While operating a peripheral in the brokers' application, notifications are sent to both employees (via emails) and log files. Furthermore, when performing a task on a device, three main checks are performed: accessibility, feasibility, and profitability.

All tasks referred to both commonality and variability aspects, and three experts checked that these questions can be answered via the two models separately. After the experiment took place, we conducted interviews with the students about their answers in order to understand what leads them to answer as they did, what their difficulties were, and how they reached their conclusions.

4.2 Study Results

We used predefined solutions for evaluating the students' outcomes. Due to differences in the grammars of ADOM and CBFM, normalized scores were calculated in which the achieved scores were divided by the desired scores. We grouped the elements according to their sources in the suggested framework: modification, guidance, product enhancement, and product validation. The modification, guidance, and product validation groups were further divided into commonality and variability aspects². The results are presented in Table 2. Since the number of students in each group was very small (4-5), we could not perform statistical analysis and, thus, we did not phrase null hypotheses from the research questions. Instead, we pointed out problems, advantages, and disadvantages which were revealed while checking the students' outcomes and interviewing them.

Table 2. Results achieved in the modification and utilization tasks

Group	Core Assets Specification		Core assets Utilization				
	Modification		Guidance		Product Enhancement	Product Validation	
	Comm.	Var.	Comm.	Var.		Comm.	Var.
CBFM	0.59	0.57	0.45	0.64	0.68	0.12	0.5
ADOM	0.57	0.56	0.69	0.62	0.47	0.25	0

The scores in the core assets modification category, of both commonality and variability-related issues, were very similar in the two modeling methods, with slight advantages in favor of CBFM. Nevertheless, some sources of difficulties to perform this task were identified in the students' interviews. First, while CBFM promotes

² This separation is irrelevant for product enhancement, which deals with application-specific additions.

hierarchical specifications in the form of trees, ADOM is basically non-hierarchical and its models may include multiple elements at the top level. Thus, tasks that involve modification of elements that reside at the same sub-tree in the CBFM model were easier to be performed following CBFM method, while cross cutting aspects were easier to be done in ADOM. An important example of this conclusion is the specification of dependencies between elements. In CBFM some of these dependencies are inherently specified in the tree structure, while in ADOM these dependencies had to use several "requires" dependencies each. Indeed, the dependency specification in ADOM yields more complex models than in CBFM. Furthermore, ADOM requires explicitly specifying meta-information, such as the stereotypes and the tagged values. Some of the students did not use this meta-information and built on the large expressiveness of UML. Another source of difficulty in ADOM was updating different types of diagrams, namely use case and class diagrams, for the same tasks. This led to confusion whether the update needs to be made only on the use case diagram, only on the class diagram, or on both. Consequently, some inconsistencies between these diagrams in the obtained modified models were found.

In CBFM, the main difficulty in core assets modification was to understand group cardinality. This may be due to the fact that group cardinality in CBFM is interpreted according to the location of the feature group in the feature diagram and not globally in the context of the entire application. This is also the case in ADOM, but ADOM enables explicit specification as tagged values of the corresponding variation points. Another observation regarding CBFM is that introduction of new elements was very difficult to be performed, and the main source of this difficulty was identifying the exact locations at which these elements need to be integrated in the tree structure. The tendency was to add new features in lower locations than required in the tree, maybe since low level elements seemed more detailed. Finally, the usage of feature model references was not performed well, probably because of inexperience of the subjects in the advanced features of the CBFM method.

Regarding core assets utilization, we found similar scores in variability aspects of core assets guidance. However, ADOM outperformed CBFM in commonality-related issues of guidance (0.69 vs. 0.45) and in commonality-related issues of product validation (0.25 vs. 0.12), while CBFM outperformed ADOM in product enhancement issues (0.68 vs. 0.47) and in variability-related issues of product validation (0.5 vs. 0). Analyzing the sources of difficulties, we first found that the existence of inheritance relations in ADOM helped understand the hierarchical structure in general and the exact attributes of low level elements in particular. For example, the students utilizing CBFM had difficulties in understanding that fax machines, scanners, and printers, which are all peripherals, exhibit also physical locations and manufacturers. This was better understood by the students who had the ADOM model and, thus, their performance in commonality-related issues of guidance was better. Furthermore, in CBFM, the students experienced difficulties in creating several instances of the same core asset element. During their interviews, the students claimed that the multiplicity specification of members in feature groups was very confusing. They thought that these multiplicities constrain up to a single variant in the whole application, while their actual meaning is up to a single variant *in each* instance of their feature group (i.e., in each realization of the corresponding group).

Regarding product enhancement, students utilizing the ADOM model believed that they are not allowed to add elements which are not specified in the core asset of the domain. Thus, they pointed on the corresponding requirements as violating the domain constraints rather than application-specific additions. This result may be attributed to the relatively rich notation of UML and ADOM with respect to CBFM: the students mainly relied on the VOF specification as expressed in the ADOM model and tended not to extend it with new requirements.

In both methods, the main source of difficulties in questions that refer to commonality-related issues of product validation was comprehension of dependencies between elements. In particular, traceability of OCL constraints was found as a very difficult task. Since ADOM enables (through UML) visual specification of some OCL constraints with {or} and {xor} constructs, the students utilizing ADOM succeeded a little bit more in this category.

Finally, only one task referred to variability-related issues in product validation. The violation in this task referred to a dependency between two variants. None of the students who utilized ADOM found this violation, while half of the students who utilized CBFM found it. We believe that the difference in this case is due to relatively crowd specifications in ADOM with respect to CBFM: for finding the violation, the class diagram, which included the variation point, the possible variants, and the exhibited attributes, had to be consulted. The corresponding specification in CBFM was much simpler and involved hierarchical structure of features. However, since only one task referred to this aspect, we cannot state any more general conclusions.

4.3 Threats to Validity

The main threats to validity are of course the small numbers of subjects (18 overall), the nature of the subjects (students and not experienced developers and domain engineers), and the relatively simplified tasks and models. In order to overcome these threats we took the following actions. First, the students had to fill pre-questionnaires that summarized their level of knowledge and skills. They were trained throughout the course with the modeling methods and reached a high level of familiarity with these methods. Moreover, we used additional information regarding the subjects, such as their grades, degrees, and previous knowledge, in order to divide them into similarly capable groups. Since we could not perform a statistical analysis due to the low number of subjects, we conducted interviews with the subjects in order to get more insights to the achieved results. We further motivated the students to produce qualitative outcomes by giving them up to 10 points to their final grades according to their performance in the assignment. Second, we carefully chose the domain and specified the corresponding models so that they will refer to different domain engineering-related challenges. Three experts checked the models and their equivalent expressiveness before the experiment. However, we aim at keeping the models simple, yet realistic, so that the subjects will be able to respond in reasonable time. Third, we conducted a comparative analysis between existing feature-oriented and UML-based methods. The methods used in the experiment were selected based on this analysis (the main reasons for this choice are mentioned in [9]).

5 Summary and Future Work

Utilization and specification of core assets is important when dealing with software product line engineering. In this paper, we analyzed the main difficulties in performing these tasks in two methods from different leading modeling paradigms: CBFM, which is a feature-oriented method, and ADOM, which is based on a UML 2 profile. We found similar results in specifying, or more accurately modifying, core assets in the two methods, while some difficulties and problems in core assets utilization were encountered in the two modeling paradigms. Despite the low number of subjects, we tried to provide some explanations to these problems, analyzing the subjects' outcomes and interviews.

Only further studies, both empirical and theoretical ones, may confirm or disconfirm whether our results can be generalized to more experienced subjects, more complicated models and tasks, and other modeling methods. In particular, we intend to compare the expressiveness, comprehensibility, and specification and utilization capabilities of different product line engineering methods and to conduct similar experiments on larger classes of domain engineering students.

References

1. Bachmann F., Celments, P., C.: Variability in Software Product Lines. Technical Report CMU/SEI-2005-TR-012 (2005), <http://www.sei.cmu.edu/library/abstracts/reports/05tr012.cfm>
2. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley, Reading (2002)
3. Czarnecki, K., Kim, C.H.P.: Cardinality based feature modeling and constraints: A progress report. In: Proceedings of the OOPSLA Workshop on Software Factories (2005)
4. Halmans, G., Pohl, K., Sikora, E.: Documenting Application-Specific Adaptations in Software Product Line Engineering. In: Bellahsene, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 109–123. Springer, Heidelberg (2008)
5. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990)
6. Kitchenham, B.A., Lawrence, S., Lesley, P., Pickard, M., Jones, P.W., Hoaglin, D.C., Emam, K.E.: Preliminary Guidelines for Empirical Research. IEEE Transactions on Software Engineering 28(8), 721–734 (2002)
7. OMG. Meta Object Facility (MOF) Specification – version 2.4, <http://www.omg.org/spec/MOF/2.4/Beta2/PDF/>
8. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, New York (2005)
9. Reinhartz-Berger, I., Tsoury, A.: Experimenting with the Comprehension of Feature-Oriented and UML-Based Core Assets. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBP, vol. 81, pp. 468–482. Springer, Heidelberg (2011)
10. Reinhartz-Berger, I., Sturm, A.: Utilizing Domain Models for Application Design and Validation. Information and Software Technology 51(8), 1275–1289 (2009)
11. Sinnema, M., Deelstra, S.: Classifying Variability Modeling Techniques. Information and Software Technology 49(7), 717–739 (2007)
12. Svahnberg, M., Van Gorp, J., Bosch, J.: A Taxonomy of Variability Realization Techniques. Software Practice & Experience 35(8), 705–754 (2005)

Actor-eUML for Concurrent Programming

Kevin Marth and Shangping Ren

Illinois Institute of Technology
Department of Computer Science
Chicago, IL USA
martkev@iit.edu

Abstract. The advent of multi-core processors offers an opportunity to increase the usage of Executable UML. Researchers are advocating the division of software systems into a productivity layer and an efficiency layer to shield mainstream programmers from the complexities of parallelism. Such separation of application and platform concerns is the foundation of Executable UML. To leverage this opportunity, an approach to Executable UML must address the complexity of the UML standard and provide a formal model of concurrency. In this paper, we introduce the Actor-eUML model and formalize the mapping between actors in the Actor model and Executable UML agents (active objects) by unifying the semantics of actor behavior and the hierarchical state machine (HSM) semantics of Executable UML agents. The UML treatment of concurrency is simplified, and the Actor model is extended to enable a set of actor behaviors to specify the HSM for an Executable UML active class.

1 Introduction

Multi-core processors have entered the computing mainstream, and many-core processors with 100+ cores are predicted within this decade. The increasing hardware parallelism and the absence of a clear software strategy for exploiting this parallelism have convinced leading computer scientists that many practicing software engineers cannot effectively program state-of-the-art processors [8]. We believe that a basis for simplifying parallel programming exists in established software technology, including the Actor model [1] and Executable UML. The advent of multi-core processors has galvanized interest in the Actor model, as the Actor model has a sound formal foundation and provides an intuitive parallel programming model. To leverage the Actor model, software systems should be specified in a language that provides first-class support for the Actor model and exposes its rather abstract treatment of parallelism. A leading parallel research program has advocated dividing the “software stack” into a productivity layer and an efficiency layer [7]. Parallel concerns are addressed in the efficiency layer by expert parallel programmers, and the productivity layer enables mainstream programmers to develop applications while being shielded from the parallel hardware platform. This separation of application concerns (productivity layer) and platform concerns (efficiency layer) is the foundation of Executable UML.

Fortunately, the Actor model and Executable UML are readily unified. In this paper, we introduce the Actor-eUML model and formalize the mapping between actors in the Actor model and agents (active objects) in Executable UML by unifying the semantics of actor behavior and the hierarchical state machine (HSM) semantics of Executable UML agents. Simply stated, an Executable UML agent is an actor whose behavior is specified as a HSM. To facilitate the definition of unified semantics for Actor-eUML, we simplify the UML treatment of concurrency and extend the Actor model to enable a set of actor behaviors to specify the HSM for an Executable UML active class. Section 2 presents an overview of the Actor-eUML model. Section 3 presents the operational semantics of the Actor-eUML model. Section 4 concludes the paper.

2 Overview of the Actor-eUML Model

2.1 Related Work

The separation of application and platform concerns is embodied in the Model-Driven Architecture (MDA) [5]. Executable UML uses profiles of the Unified Modeling Language [11] to support the MDA and enable the specification of an *executable* platform-independent model (PIM) of a software system that can be translated to a platform-specific implementation (PSI) using a model compiler. Several approaches to Executable UML exist [4, 6, 9], and each approach enables a software system to be specified using the following process.

- The software system is decomposed into domains (concerns).
- Each domain is modeled in a class diagram using several classes.
- Each class has structural and behavioral properties, including associations, attributes, operations, and a state machine.
- A formal action language is used to specify the implementation of operation methods and state machine actions.

In both xUML [4] and xtUML [9], only simple state machines are supported, and many HSM features are not available. In contrast, all standard UML HSM features are supported in Actor-eUML, with the exception of features that imply concurrency within a HSM. The foundational subset for Executable UML models (fUML) [12] precisely specifies the semantics of the UML constructs considered to be used most often. As such, the fUML specification does not address all state machine features and explicitly does not support state machine features such as call events, change events, and time events.

The Actor-eUML model has a formal concurrency model (the Actor model), while existing approaches to Executable UML lack a formal treatment of concurrency beyond the operational requirement for the modeler and/or the model compiler to synchronize the conceptual threads of control associated with active class instances. Some HSM features, such as deferring certain messages when an actor is in a given state, have been implemented in actor-based programming languages using reflective mechanisms that modify the behavior of the mail queue for an actor [10], but these actor-based languages lack full-featured HSM support.

2.2 Hierarchical State Machines in Actor-eUML

The Actor-eUML model promotes HSM usage because state-based behavior is fundamental to object-based programming. Hierarchical states facilitate *programming by difference*, where a substate inherits behavior from superstates and defines only behavior that is specific to the substate. A design invariant can be specified once at the appropriate level in a state hierarchy, eliminating redundancy and minimizing maintenance effort. Standard UML supports a variant of Harel statecharts [3] that enables behavior to be specified using an extended HSM that combines Mealy machines, where actions are associated with state transitions, and Moore machines, where actions are associated with states. Actor-eUML supports internal, external, local, and start transitions as specified in standard UML and provides the following support for events and states.

Events. As in the Actor model, agents in Actor-eUML communicate using only asynchronous message passing. A message received by an agent is dispatched to its HSM as a *signal* - a named entity with a list of parameters. Actor-eUML supports three kinds of HSM events:

- a *signal* event that occurs when a signal is dispatched,
- a *time* event that occurs when a timer expires after a specified duration, and
- a *change* event that occurs when a Boolean expression becomes true.

Events are processed serially and to completion. Although there can be massive parallelism among agents, processing within each agent is strictly sequential.

States. A state in an Actor-eUML HSM has several features: an optional name, entry actions, exit actions, transitions, deferred events, and a nested state machine. *Entry* actions and *exit* actions are executed when entering and exiting the state, respectively. *Deferred* events are queued and handled when the state machine is in another state in which the events are not deferred. A state in a state machine can be either simple or composite. A composite state has a nested state machine.

In standard UML, a composite state can have multiple orthogonal regions, and each region has a state machine. Orthogonal regions within a composite state introduce concurrency within a HSM, since the state machine within each region of a composite state is active when the composite state is active. Standard UML also supports a *do* activity for each state that executes concurrently with any *do* activity elsewhere in the current state hierarchy. The Actor model avoids concurrency within an actor. To align with the Actor model, Actor-eUML does not allow concurrency within a HSM and consequently does not support *do* activities or orthogonal regions. In practice, orthogonal regions are often not independent and share data. The UML standard states that orthogonal regions should interact with signals and should not explicitly interact using shared memory. Thus, replacing orthogonal regions in Actor-eUML by coordinated peer agents is appropriate.

2.3 Simplified Concurrency in Actor-eUML

To align the Actor-eUML model with the Actor model, it is necessary to eliminate HSM features that introduce concurrency within an active object, but additional simplification is required to complete the alignment. The treatment of concurrency in standard UML is heterogeneous and complex. An active object (i.e. agent) has a dedicated conceptual thread of control, while a passive object does not. The calls to operations for active classes and passive classes can be either synchronous or asynchronous, and it is possible to combine operations and a state machine when defining the behavior of an active class. An active object in standard UML can be either internally sequential or internally concurrent, depending upon whether it has a state machine and whether the state machine uses operation calls as state machine triggers. A passive object can also be either internally sequential or internally concurrent, since each operation of a passive class is defined to be sequential, guarded, or concurrent.

It is apparent that the multiple interacting characteristics of the features of active and passive classes add complexity to the treatment of concurrency in standard UML. The treatment of concurrency in existing Executable UML approaches (including fUML) is simpler, but it is still possible to have multiple threads of control executing concurrently within an agent. Actor-eUML further streamlines the treatment of concurrency.

- A passive class can define only synchronous, sequential operations.
- A passive object is encapsulated within one agent, and an agent interacts with its passive objects only through synchronous operation calls.
- Agents interact only through asynchronous signals sent to state machines.
- An active class can define operations, but a call to an agent operation is simply notation for an implicit synchronous signal exchange with the agent.

A call to an agent operation sends a signal with the same signature to the HSM for the agent and blocks the caller until the signal is processed and a reply signal is received. Thus, any communication with an agent is a signal event that is interleaved serially with other HSM events in the thread of control for the agent. This treatment of agent interaction ensures that agents are internally sequential and avoids the complexities of concurrent access to the internal state of an agent. With these simplifications, the Actor-eUML concurrency model aligns with the Actor model and is safer than multi-core programming models that require the programmer to explicitly synchronize concurrent access to shared memory.

2.4 Actors in Actor-eUML

The Actor model [1] is a formal theory of computation and concurrency based on active, autonomous, encapsulated objects that communicate exclusively through asynchronous message passing. The Actor model has a sound mathematical foundation but is also influenced by implementation concerns and the laws of physics. The Actor model acknowledges that messages can encounter bounded but indeterminate delays in transmission and can therefore be delivered out of order.

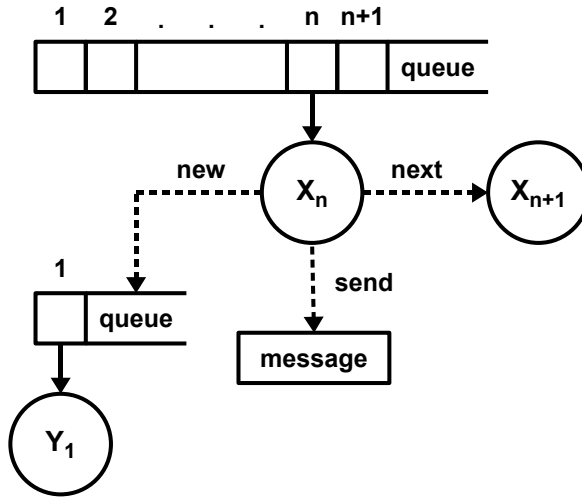


Fig. 1. An Actor in the Actor Model [1]

As illustrated in Fig. 1, in response to each message received from its abstract mailbox (external queue), an actor X can:

- create a finite number of new actors,
- send a finite number of messages to other actors, and
- select the behavior used to process the next message.

The Actor model is characterized by inherent concurrency among actors. An actor is allowed to pipeline the processing of messages by selecting the behavior used to process the next message and actually dispatching the next message for processing before the processing of the current message has completed. However, pipelined actor behaviors cannot share internal state, and the Actor model does not require message pipelining. The ability to pipeline messages is not compatible with HSM semantics, as the exit and entry actions for a transition must execute before the next transition can be triggered, so actors in the Actor-eUML model that realize HSM behavior do not attempt message pipelining. However, other actors in the Actor-eUML model can use message pipelining.

An actor can send messages to its own mailbox, but a message an actor sends to itself is interleaved with messages received from other actors and is not guaranteed to be the next message dispatched. This consideration and the requirement that an actor consume a message with each behavior change lead to a continuation-passing style that uses cooperating auxiliary actors to process a single client message. This style of programming adds conceptual overhead for programmers who find it confusing and adds implementation overhead that cannot always be eliminated by smart compilers and sophisticated schemes aimed at minimizing the performance impact of actor creation and communication.

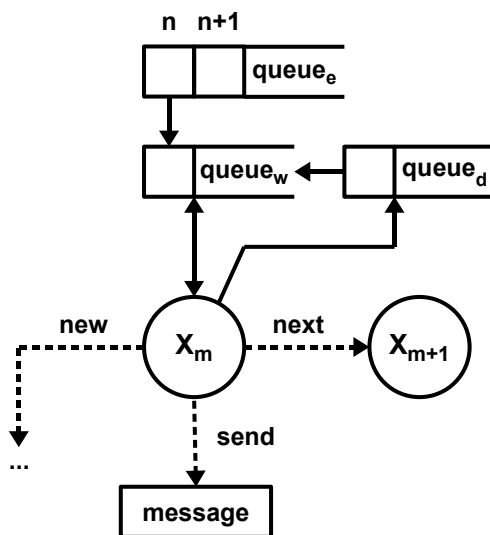


Fig. 2. An Actor in the Actor-eUML Model

The Actor-eUML model retains the essence of the pure Actor model while adding capabilities that simplify actor programming and enable HSM behavior to be expressed directly and conveniently. As illustrated in Fig. 2, the actor interface to its abstract mailbox has been extended with two internal queues: a working queue ($queue_w$) for internal messages used while processing a single external message, and a defer queue ($queue_d$) used to defer messages based on the current state in a HSM. The external queue ($queue_e$) that receives messages sent to an actor has been retained. When a message is dispatched from $queue_e$, the message is moved to $queue_w$ and then dispatched for processing by the next behavior. As the behavior executes, messages can be added to $queue_w$. When the behavior completes, the message at the head of $queue_w$ is dispatched to the next behavior. If the $queue_w$ is empty when a behavior completes, the next message is dispatched from $queue_e$. A message dispatched from $queue_w$ can be deferred to $queue_d$. The messages in $queue_d$ are moved to $queue_w$ to revisit them.

The additional internal queues enable a single actor to completely process a client message without creating and communicating with auxiliary actors and also facilitate the expression of HSM behavior. Each state in a HSM is mapped to an actor behavior, and a signal is delegated from the current state to its parent state by adding the signal to $queue_w$ and then selecting the behavior for its parent state as the next behavior. A sequence of start, entry, and exit actions is executed during a state transition by adding specialized messages to $queue_w$ and then selecting the behavior of the next state in the sequence. A signal that is deferred in the current state is added to $queue_d$. Deferred messages are revisited after a state transition by moving the messages from $queue_d$ to $queue_w$.

3 Actor-eUML Semantics

The Actor-eUML model defines the following actor primitives, where \mathbb{B} is a behavior, \mathbb{M} is a message, and \mathbb{V} is a list of parameter values. The call $\mathbb{B}(\mathbb{V})$ returns a closure, which is a function and a referencing environment for the non-local names in the function that binds the nonlocal names to the corresponding variables in scope at the time the closure is created. The closure returned by the call $\mathbb{B}(\mathbb{V})$ captures the variables in \mathbb{V} , and the closure expects to be passed \mathbb{M} as a parameter when called subsequently.

- **actor-new**(\mathbb{B}, \mathbb{V}): create a new actor with initial behavior $\mathbb{B}(\mathbb{V})$.
- **actor-next**($A_i, \mathbb{B}, \mathbb{V}$): select $\mathbb{B}(\mathbb{V})$ as the next behavior for actor A_i .
- **actor-send**(A_i, \mathbb{M}): send \mathbb{M} to the tail of queue_e for actor A_i .
- **actor-push**(A_i, \mathbb{M}): push \mathbb{M} at the head of queue_w for actor A_i .
- **actor-push-defer**(A_i, \mathbb{M}): push \mathbb{M} at the head of queue_d for actor A_i .
- **actor-move-defer**(A_i): move all messages in queue_d to queue_w for actor A_i .

The $\{\text{actor-new}, \text{actor-next}, \text{actor-send}\}$ primitives are inherited from the Actor model, and the $\{\text{actor-push}, \text{actor-push-defer}, \text{actor-move-defer}\}$ primitives are extensions to the Actor model introduced by the Actor-eUML model. When transforming a PIM to a PSI, a model compiler for an Actor-eUML implementation translates the HSM associated with each active class to a target programming language in which the actor primitives have been embedded.

At any point in a computation, an actor is either quiescent or actively processing a message. The term $\text{actor}_4(A_i, Q, C, M)$ denotes a quiescent actor, where A_i uniquely identifies the actor, Q is the queue for the actor, C is the closure used to process the next message dispatched by the actor, and M is the local memory for the actor. The queue Q is a 3-tuple $\langle Q_e, Q_w, Q_d \rangle$, where Q_e is the external queue where messages sent to the actor are received, Q_w is the work queue used when processing a message \mathbb{M} dispatched by the actor, and Q_d is used to queue messages deferred after dispatch. At points in a computation, a component of Q can be empty and is denoted by Q_\perp . The term $\text{actor}_5(A_i, Q, C_\perp, M, E \vdash S)$ denotes an active actor and extends the actor_4 term to represent a computation in which statement list S is executing in environment E . An active actor has a null C , denoted by C_\perp .

A transition relation between actor configurations is used to define the Actor-eUML operational semantics, as in [2]. A configuration in an actor computation consists of actor_4 , actor_5 , and send terms. The $\text{send}(A_i, \mathbb{M})$ term denotes a message \mathbb{M} sent to actor A_i that is in transit and not yet received. The specification of structural operational semantics for Actor-eUML uses rewrite rules to define computation as a sequence of transitions among actor configurations.

Rules (1) and (2) define the semantics of message receipt. A message \mathbb{M} sent to actor A_i can be received and appended to the external queue for actor A_i when the actor is quiescent (1) or active (2). The send term is consumed and eliminated by the rewrite. The message receipt rules illustrate several properties explicit in the Actor model. An actor message is an asynchronous, reliable, point-to-point communication between two actors. The semantics of message receipt

are independent of the message sender. Each message that is sent is ultimately received, although there is no guarantee of the order in which messages are received. A message cannot be broadcast and is received by exactly one actor.

$$\begin{aligned} \text{send}(A_i, \mathbb{M}) \text{ actor}_4(A_i, \langle Q_e, Q_w, Q_d \rangle, C, M) \\ \longrightarrow \text{actor}_4(A_i, \langle Q_e:\mathbb{M}, Q_w, Q_d \rangle, C, M) \end{aligned} \quad (1)$$

$$\begin{aligned} \text{send}(A_i, \mathbb{M}) \text{ actor}_5(A_i, \langle Q_e, Q_w, Q_d \rangle, C_\perp, M, E \vdash S) \\ \longrightarrow \text{actor}_5(A_i, \langle Q_e:\mathbb{M}, Q_w, Q_d \rangle, C_\perp, M, E \vdash S) \end{aligned} \quad (2)$$

Rules (3) and (4) define the semantics of message dispatch. In rule (3), the quiescent actor A_i with non-empty Q_e and empty Q_w initiates the dispatch of the message \mathbb{M} at the head of Q_e by moving \mathbb{M} to Q_w . In rule (4), the quiescent actor A_i completes message dispatch from Q_w and becomes an active actor by calling the closure C to process \mathbb{M} in the initial environment E_c associated with C . The message dispatch rules enforce the serial, run-to-completion processing of messages and the demand-driven relationship between Q_e and Q_w in the Actor-eUML model. An actor cannot process multiple messages concurrently, and a message is dispatched from Q_e only when Q_w is empty.

$$\text{actor}_4(A_i, \langle \mathbb{M}:Q_e, Q_\perp, Q_d \rangle, C, M) \longrightarrow \text{actor}_4(A_i, \langle Q_e, \mathbb{M}, Q_d \rangle, C, M) \quad (3)$$

$$\begin{aligned} \text{actor}_4(A_i, \langle Q_e, \mathbb{M}:Q_w, Q_d \rangle, C, M) \\ \longrightarrow \text{actor}_5(A_i, \langle Q_e, Q_w, Q_d \rangle, C_\perp, M, E_C \vdash (\text{call } C \ \mathbb{M})) \end{aligned} \quad (4)$$

Rules (5), (6), and (7) define the semantics of the **actor-new**, **actor-next**, and **actor-send** primitives, respectively. In rule (5), the active actor A_i executes the **actor-new** primitive to augment the configuration with an **actor₄** term that denotes a new actor A_n with empty Q , uninitialized local memory (M_\perp), and initial behavior closure $C = \mathbb{B}(\mathbb{V})$. In rule (6), the active actor A_i executes the **actor-next** primitive to select its next behavior closure $C = \mathbb{B}(\mathbb{V})$ and becomes a quiescent actor. In rule (7), the active actor A_i executes the **actor-send** primitive to send the message \mathbb{M} to actor A_j , where both $i = j$ and $i \neq j$ are well-defined.

$$\begin{aligned} \text{actor}_5(A_i, Q, C_\perp, M, E \vdash \text{actor-new}(\mathbb{B}, \mathbb{V}); S) \\ \longrightarrow \text{actor}_5(A_i, Q, C_\perp, M, E \vdash S) \text{ actor}_4(A_n, \langle Q_\perp, Q_\perp, Q_\perp \rangle, C, M_\perp) \end{aligned} \quad (5)$$

$$\begin{aligned} \text{actor}_5(A_i, Q, C_\perp, M, E \vdash \text{actor-next}(A_i, \mathbb{B}, \mathbb{V}); S) \\ \longrightarrow \text{actor}_4(A_i, Q, C, M) \end{aligned} \quad (6)$$

$$\begin{aligned} \text{actor}_5(A_i, Q, C_\perp, M, E \vdash \text{actor-send}(A_j, \mathbb{M}); S) \\ \longrightarrow \text{actor}_5(A_i, Q, C_\perp, M, E \vdash S) \text{ send}(A_j, \mathbb{M}) \end{aligned} \quad (7)$$

Rules (8), (9), and (10) define the semantics of the **actor-push**, **actor-push-defer**, and **actor-move-defer** primitives, respectively.

$$\begin{aligned} & \text{actor}_5(A_i, \langle Q_e, Q_w, Q_d \rangle, C_\perp, M, E \vdash \text{actor-push}(A_i, \mathbb{M}); S) \\ \longrightarrow & \text{actor}_5(A_i, \langle Q_e, \mathbb{M}:Q_w, Q_d \rangle, C_\perp, M, E \vdash S) \end{aligned} \quad (8)$$

$$\begin{aligned} & \text{actor}_5(A_i, \langle Q_e, Q_w, Q_d \rangle, C_\perp, M, E \vdash \text{actor-push-defer}(A_i, \mathbb{M}); S) \\ \longrightarrow & \text{actor}_5(A_i, \langle Q_e, Q_w, \mathbb{M}:Q_d \rangle, C_\perp, M, E \vdash S) \end{aligned} \quad (9)$$

$$\begin{aligned} & \text{actor}_5(A_i, \langle Q_e, Q_w, Q_d \rangle, C_\perp, M, E \vdash \text{actor-move-defer}(A_i); S) \\ \longrightarrow & \text{actor}_5(A_i, \langle Q_e, Q_w:\text{reverse}(Q_d), Q_\perp \rangle, C_\perp, M, E \vdash S) \end{aligned} \quad (10)$$

The actor primitives intrinsic to the Actor-eUML model are the foundation for other abstractions useful in realizing an implementation of HSM behavior. The following HSM abstractions are typically defined as macros in the target programming language. The HSM abstraction macros implement HSM behavior using exclusively the Actor-eUML primitives **actor-push** and **actor-next** and the internal work queue Q_w , in combination with parameter passing between actor behaviors. The HSM abstraction macros facilitate a direct translation from a HSM specification to its implementation.

- The **HSM-state-start** macro realizes the start transition for a state.
- The **HSM-state-entry** macro realizes the entry actions for a state.
- The **HSM-state-exit** macro realizes the exit actions for a state.
- The **HSM-state-reset** macro returns the HSM to its current state prior to delegating a signal up the HSM hierarchy without consuming the signal.
- The **HSM-state-next** macro delegates a signal to a superstate.
- The **HSM-state-transition** macro realizes a local or external transition from a source state to a target state.
- The **HSM-state-transition-internal** macro realizes an internal transition in a state.

The **actor-push-defer** and **actor-move-defer** primitives implement deferred signals within a HSM, and the **actor-send** primitive is used to send an asynchronous signal from the HSM for an agent to the HSM for a target agent.

A reference implementation of the Actor-eUML model and the associated HSM abstraction macros has been developed in Common Lisp, confirming that a direct and efficient realization of the model is practical. A C++ implementation of the Actor-eUML model is also in development and will be the target language for a model compiler, enabling software engineers to specify Executable UML models oriented to the problem space that abstract the programming details of the Actor-eUML model in the solution space. However, the mapping from the UML agents in the problem space to the Actor-eUML actors in the solution space is direct, reducing the semantic distance between a UML specification and its realization and ensuring that UML specifications are founded on a formal model of concurrency that provides a logically sound and intuitive basis for reasoning about parallel behavior and analyzing run-time performance.

4 Summary and Conclusion

The advent of multi-core processors signaled a revolution in computer hardware. We believe that it is possible to program multi-core and many-core processors by using an evolutionary approach that leverages established software technology, notably the Actor model and Executable UML. The Actor model has a sound formal foundation and provides an intuitive and safe concurrent programming model. Executable UML consolidates and standardizes several decades of experience with object-based programming. Unifying the Actor model and Executable UML in the Actor-eUML model provides a concurrency model that exploits massive inter-agent parallelism while ensuring that agent behaviors retain the familiarity and simplicity of sequential programming. The HSM is the foundation of agent behavior in Actor-eUML. The Actor-eUML model streamlines the UML concurrency model, eliminates HSM features that imply intra-agent concurrency, and introduces conservative extensions to the structure and behavior of message dispatch in the Actor model. A definition of the operational semantics of the actor primitives provided by the Actor-eUML model was presented, and a reference implementation of the Actor-eUML model is available.

References

1. Agha, G.: *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge (1986)
2. Agha, G., Mason, I.A., Smith, S.F., Talcott, C.L.: A Foundation for Actor Computation. *Journal of Functional Programming*, 1–72 (1997)
3. Harel, D.: *Statecharts: A Visual Formalism for Complex Systems*. *Science of Computer Programming* 8(3), 231–274 (1987)
4. Mellor, S.J., Balcer, S.J.: *Executable UML: A Foundation for Model-Driven Architecture*. Addison-Wesley, Reading (2002)
5. Mellor, S.J., Kendall, S., Uhl, A., Weise, D.: *MDA Distilled*. Addison-Wesley, Reading (2004)
6. Milicev, D.: *Model-Driven Development with Executable UML*. Wiley, Chichester (2009)
7. Patterson, D., et al.: A View of the Parallel Computing Landscape. *Communications of the ACM* 52(10), 56–67 (2009)
8. Patterson, D.: The Trouble with Multi-Core. *IEEE Spectrum* 47(7), 28–32 (2010)
9. Raistrick, C., Francis, P., Wright, J., Carter, C., Wilkie, I.: *Model Driven Architecture with Executable UML*. Cambridge University Press, Cambridge (2004)
10. Tomlinson, C., Singh, V.: Inheritance and Synchronization with Enabled Sets. *SIGPLAN Notices* 24(10), 103–112 (1989)
11. Object Management Group: *UML Superstructure Specification, Version 2.1.2*, <http://www.omg.org/docs/formal/07-11-02.pdf>
12. Object Management Group: *Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0*, <http://www.omg.org/spec/FUML>

Preface to the Posters and Demonstrations

This volume also contains the papers presented at Posters and Demonstrations session of the 30th International Conference on Conceptual Modeling, held on October 31 to November 3, 2011 in Brussels.

The committee decided to accept 9 papers for this session. We hope that you find the contributions beneficial and enjoyable and that during the session you had many opportunities to meet colleagues and practitioners. We would like to express our gratitude to the program committee members for their work in reviewing papers, the authors for submitting their papers, and the ER 2011 organizing committee for all their support.

July 2011
Brussels

Roland Billen
Pierre Hallot

An Eclipse Plugin for Validating Names in UML Conceptual Schemas

David Aguilera, Raúl García-Ranea, Cristina Gómez, and Antoni Olivé

Department of Service and Information System Engineering
BarcelonaTech – Universitat Politècnica de Catalunya
Barcelona, Spain
{daguilera, cristina, olive}@essi.upc.edu,
raul.garcia-ranea@est.fib.upc.edu

Abstract. Many authors agree on the importance of choosing good names for conceptual schema elements. Several proposals of naming guidelines are available in the literature, but the support offered by current CASE tools is very limited and, in many cases, insufficient. In this demonstration we present an Eclipse plugin that implements a specific proposal of naming guidelines. The implemented proposal provides a guideline for every kind of named element in UML. By using this plugin, the modelers can automatically check whether the names they gave to UML elements are grammatically correct and generate a verbalization that can be analysed by domain experts.

Keywords: Naming Guidelines, Eclipse, Conceptual Schemas.

1 Introduction

Names play a very important role on the understandability of a conceptual schema. Many authors agree that choosing good names for schema elements make conceptual schemas easier to understand for requirements engineers, conceptual modelers, system developers and users [5,6].

Choosing good names is one of the most complicated activities related to conceptual modeling [8, p.46]. There have been several proposals of naming guidelines for some conceptual schema elements in the literature [3,7] but, as far as we know, few CASE tools support this activity. One example is [2], which controls that the capitalization of some elements is “correct”, like “classes should start with a capital letter”.

In this demonstration, we present an Eclipse plugin that adds naming validation capabilities to the UML2Tools framework. This plugin can assist modelers during the naming validation process of named elements in UML, following the complete naming guidelines presented in [1].

2 Overview of the Naming Guidelines

There are several naming guidelines available in the literature on how to name conceptual schema elements. We implemented the proposal presented in [1]

because it is complete: for each kind of element to which a modeler may give a name in UML, it provides a guideline on how to name it. As an example, two guidelines are summarized in the following:

Guideline for Entity Types

- \mathcal{G}_{1f} The name of an entity type should be a noun phrase whose head is a countable noun in singular form. The name should be written in the Pascal case.
- \mathcal{G}_{1s} If N is the name of an entity type, then the following sentence must be grammatically well-formed and semantically meaningful:
 An instance of this entity type is [a|an] *lower*¹(N)

Guideline for Boolean Attributes

- \mathcal{G}_{2f} The name A should be a verb phrase in third-person singular number, in the Camel case.
- \mathcal{G}_{2s} The following sentence must be grammatically well-formed and semantically meaningful:
 [$A|An$] *lower*(E) *lower*(*withOrNeg*²(A)) [*, or it may be unknown*].
 where the last optional fragment is included only if *min* is equal to zero.

As stated in [1], a name given by a conceptual modeler complies with the guideline if: a) it has the corresponding grammatical form \mathcal{G}_f , and b) the sentence generated from the pattern sentence \mathcal{G}_s and the given name is grammatically well-formed and semantically meaningful.

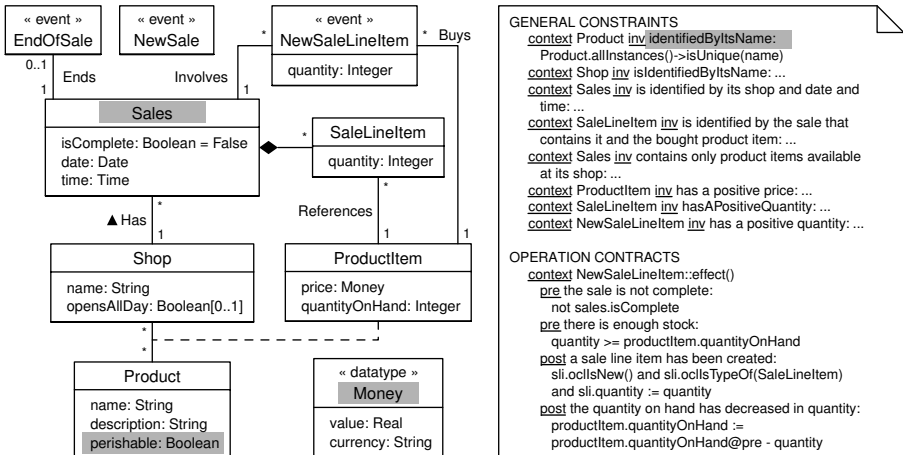


Fig. 1. Example of a conceptual schema with some names violating their guidelines

¹ lower(N) is a function that gives N in lower case and using blanks as delimiters.
² withOrNeg(A) extends A with the insertion of the negative form of the verb of A .

Figure 1 shows an example where some names, which are highlighted, violate their naming guidelines. The next section describes how to use the developed Eclipse plugin to detect those errors.

3 Naming Validation Plugin for Eclipse

Eclipse is an open, extensible development environment (IDE) written in Java. It is a small kernel with a plugin loader surrounded by hundreds of plugins [4]. Eclipse, in combination with the UML2Tools plugin, can manage UML files and permits modelers to define conceptual schemas.

We conceived our tool as an Eclipse plugin that extends the UML2Tools framework by adding two main functionalities. The first one checks if the names of the schema follow the grammatical form defined in their corresponding guidelines. The second functionality verbalizes the schema in a document.

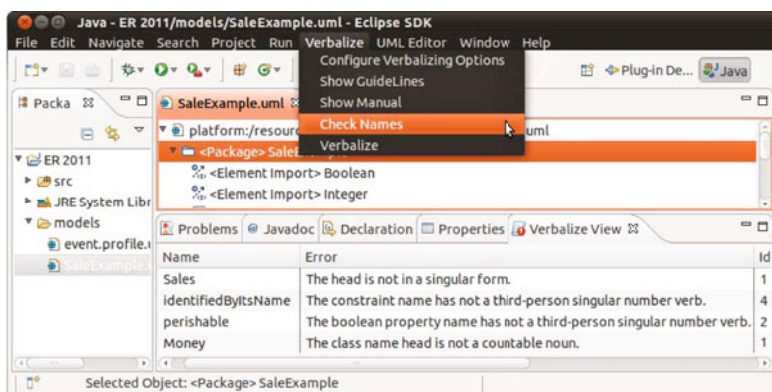


Fig. 2. Screenshot of Eclipse showing those names that violate their guidelines

Figure 2 shows a screenshot of the first functionality in action. By selecting the Package and then clicking on menu **Verbalize** \triangleright **Check Names**, our tool checks the whole conceptual schema looking for errors. If the modeler selected a few elements instead of the package, only those elements are checked instead of

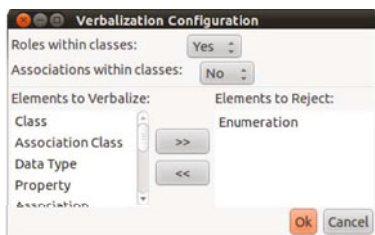


Fig. 3. Screenshot of the configuration window

the whole schema. If one or more names are incorrect, the errors are shown in a new Eclipse view.

The second functionality introduces schema verbalization. Our tool generates a PDF file containing the pattern sentences defined in the naming guidelines and the names of the selected elements. Some aspects of the resulting document can be configured by using the configuration window shown in Fig. 3. In order to generate the document, the modeler has to click on **Verbalize** > **Verbalize**. Figure 4 shows the verbalization of the schema of Fig. 1 after correcting the errors previously detected. Then, the modeler or a domain expert may check the document and detect whether the generated sentences are grammatically well-formed and semantically meaningful.

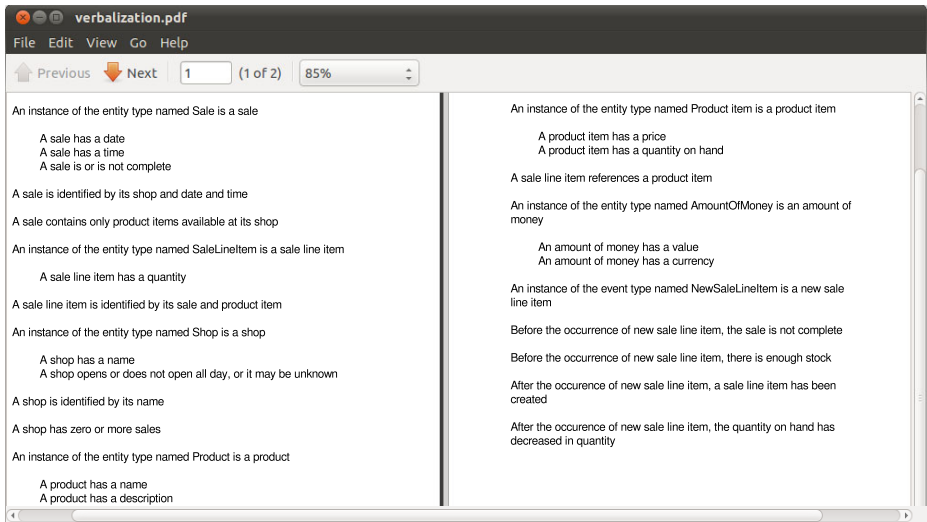


Fig. 4. Screenshot of the resulting document with the schema verbalization

Acknowledgements. Our thanks to the people in the GMC research group. This work has been partly supported by the *Ministerio de Ciencia y Tecnología* under TIN2008-00444 project, *Grupo Consolidado*, and by BarcelonaTech – *Universitat Politècnica de Catalunya*, under FPI-UPC program.

References

1. Aguilera, D., Gómez, C., Olivé, A.: A complete set of guidelines for naming UML conceptual schema elements (submitted for publication, 2011)
2. ArgoUML: ArgoUML, <http://argouml.tigris.org>
3. Chen, P.: English sentence structure and entity-relationship diagrams. *Inf. Sci.* 29(2-3), 127–149 (1983)
4. Clayberg, E., Rubel, D.: Eclipse Plug-ins. Addison-Wesley, Reading (2008)
5. Deissenboeck, F., Pizka, M.: Concise and consistent naming. *Softw. Qual. Control* 14, 261–282 (2006)

6. Meyer, B.: Reusable Software: the Base object-oriented component libraries. Prentice-Hall, Englewood Cliffs (1994)
7. Meziane, F., Athanasakis, N., Ananiadou, S.: Generating natural language specifications from UML class diagrams. *Requir. Eng.* 13(1), 1–18 (2008)
8. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-oriented modeling and design. Prentice-Hall, Englewood Cliffs (1991)

KEYRY: A Keyword-Based Search Engine over Relational Databases Based on a Hidden Markov Model*

Sonia Bergamaschi¹, Francesco Guerra¹, Silvia Rota¹, and Yannis Velegrakis²

¹ Università di Modena e Reggio Emilia, Italy

firstname.lastname@unimore.it

² University of Trento, Italy

velgias@disi.unitn.eu

Abstract. We propose the demonstration of KEYRY, a tool for translating keyword queries over structured data sources into queries in the native language of the data source. KEYRY does not assume any prior knowledge of the source contents. This allows it to be used in situations where traditional keyword search techniques over structured data that require such a knowledge cannot be applied, i.e., sources on the hidden web or those behind wrappers in integration systems. In KEYRY the search process is modeled as a Hidden Markov Model and the List Viterbi algorithm is applied to computing the top- k queries that better represent the intended meaning of a user keyword query. We demonstrate the tool's capabilities, and we show how the tool is able to improve its behavior over time by exploiting implicit user feedback provided through the selection among the top- k solutions generated.

1 Introduction

The vast majority of existing keyword search techniques over structured data relies heavily on an a-priori creation of an index on the contents of the database. At run time, the index is used to locate in the data instance the appearance of the keywords in the query provided by the user, and then associate them by finding possible join paths. This approach makes the existing solutions inapplicable in all the situations where the construction of such an index is not possible. Examples include databases on the hidden web, or behind wrappers in integration systems. To cope with this issue we have developed KEYRY (from KEYword to queRY) [2], a system that converts keyword queries into structured queries expressed in the native language of the source, i.e., SQL. The system is based only on the semantics provided by the source itself, i.e., the schema, auxiliary semantic information that is freely available, i.e., public ontologies and thesauri, a Hidden Markov Model (HMM) and an adapted notion of authority [4]. Using this information, it builds a ranked list of possible interpretations of the keyword query in terms of the underlying data source schema structures. In practice, the tool computes only the top- k most prominent answers and not the whole answer space. One of the features of KEYRY is that the order and the proximity of the keywords in the queries play a central role in determining the k most prominent interpretations.

* This work was partially supported by project "Searching for a needle in mountains of data" <http://www.dbgroup.unimo.it/keymantic>.

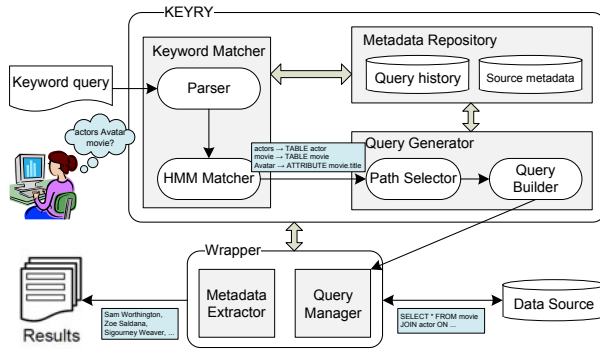


Fig. 1. KEYRY functional architecture

Apart from the obvious use case of querying hidden web sources, KEYRY finds two additional important applications. First, it allows keyword queries to be posed on information integration systems that traditionally supports only structured queries. Second, it can be used as a database exploration tool. In particular, the user may pose some keyword query, and the system will generate its possible interpretations on the given database. By browsing these interpretations, i.e., the generated SQL queries, the user may obtain information on the way the data is stored in the database.

2 KEYRY at a Glance

Interpreting a keyword query boils down to the discovery of a bipartite assignment of keywords to database structures. In some previous work we have studied the problem from a combinatorial perspective [1] by employing the the Hungarian algorithm [3]. The tool we demonstrate here is instead based on a Hidden Markov Model. A detailed description of the methodology can be found on our respective research paper [2]. A graphic illustration of the different components of the tool can be found in Figure 1.

The first step when a keyword query has been issued is to associate each keyword to some database structure. That structure may be a table name, a class, an attribute, or an actual value. Each assignment of the keywords to database structures is a *configuration*. The computation of the different configurations is done by the *Keyword Matcher* component. A configuration describes possible meanings of the keywords in the query. Given the lack of access to the instance data that we assume, finding the configurations and subsequently the interpretations of a keyword query and selecting k most prominent is a challenging task. Consider for instance the query “movie Avatar actors” over a relational database collecting information about movies. The query may be intended to find actors acting in the movie “Avatar” which means that the keywords *movie* and *actors* could be respectively mapped into the tables *movie* and *actor* while *Avatar* could be a value of the attribute *title* of the table *movie*. Different semantics are also possible, e.g. *Avatar* could be an actor or a character, or something else.

The *Keyword Matcher* is the main component of the system. It implements a Hidden Markov Model (HMM), where the observations represent the keywords and the states the data source elements. The main advantages of this representation are that the HMM

takes into account both the likelihood of single keyword-data source element associations, and the likelihood of the match between the whole keyword sequence in the query to the data source structure. In this way, the assignment of a keyword to a data source element may increase or decrease the likelihood that another keyword corresponds to a data source element. In order to define a HMM, the values of a number of parameters need to be specified. This is usually done using a training algorithm that, after many iterations, converges to a good solution for the parameter values. Therefore, finding a suitable dataset for training the HMM is a critical aspect for the effective use of HMM-based approaches in real environments. In our scenario, the parameters are initialized by exploiting the semantic information collected in the *Metadata Repository* (explained below), providing that way keyword search capabilities even without any training data, as explained in more details in our paper [2].

To obtain more accurate results, the HMM can be trained, i.e. the domain-independent start-up parameters may be specialized for a specific data source thanks to the users' feedbacks. We train KEYRY with a semi-supervised approach that exploits both the feedbacks (when available) implicitly provided by the users on the system results as supervised data and the query log as unsupervised data. By applying the List Viterbi algorithm [5] to an HMM, the top-k state sequences which have the highest probability of generating the observation sequence are computed. If we consider the user keyword query as an observation sequence, the algorithm retrieves the state sequences (i.e., sequences of HMM states representing data source terms) that more likely represent the intended meaning of the user query.

The HMM Matcher can present the computed top-k configurations to the user who, in turn, selects the one that better represents the intended meaning of his/her query. This allows the tool to reduce the number of queries to be generated (the ones related to the configuration selected) and to train the HMM parameters.

Given a configuration, the different ways that the data structures on which the keywords have been mapped can be connected, need to be found (e.g., by discovering join paths). A configuration alongside the join paths describe some possible semantics of the whole keyword query, and can be expressed into some query in the native query language of the source. Such queries are referred to as *interpretations*. As an example, consider the configuration mentioned above in which the keywords `movie` and `actor` are mapped into the tables `movie` and `actor`, respectively, while the keyword `Avatar` to the `title` of the table `movie`, may represent the actors that acted in the movie "Avatar", or the movies where acted actors of the movie Avatar, etc., depending on the path selected and the tables involved. The path computation is the main task of the *Query Generator* module. Different strategies have been used in the literature to select the most prominent one, or provide an internal ranking based on different criteria, such as the length of the join paths. KEYRY uses two criteria: one is based on the shortest path and the other is using the HITS algorithm [4] to classify the relevance of the data structures involved in a path.

The tasks of generating the various configurations and subsequently the different interpretations are supported by a set of auxiliary components such as the *Metadata Repository* that is responsible for the maintenance of the metadata of the data source structures alongside previously executed user queries. KEYRY has also a set of *Wrappers* for managing the heterogeneity of the data sources. Wrappers are in charge of

extracting metadata from data sources and formulating the queries generated by KEYRY in the native source languages.

3 Demonstration Highlights

In this demonstration we intend to illustrate the use of KEYRY and communicate to the audience a number of important messages. The demonstration will consist of a number of different application scenarios such as querying the IMDB database (www.imdb.com), the DBLP collection (dblp.uni-trier.de) and the Mondial database (<http://www.dbis.informatik.uni-goettingen.de/Mondial>). We will first show the behavior of KEYRY without any training. We will explain how the metadata of the sources is incorporated into our tool and how heuristic rules allow the computation of the main HMM parameters. We will run a number of keyword queries against the above sources and explain the results. These queries are carefully selected to highlight the way the tool deals with the different possible mappings of the keywords to the database structures. The participants will also have the ability to run their own queries. Furthermore, we will consider the parameters obtained by different amounts of training and we will compare the results to understand how the amount of training affects the final result.

The important goals and messages we would like to communicate to the participants though the demo are the following. First, we will demonstrate that keyword-based search is possible even without prior access to the data instance, and is preferable from formulating complex queries that require skilled users who know structured query languages and how/where the data is represented in the data source. Second, we will show that using a HMM is a successful approach in generating SQL queries that are good approximations of the intended meaning of the keyword queries provided by the user. Third, we will illustrate how previously posed queries are used to train the search engine. In particular, we will show that the implicit feedback provided by the user selecting an answer among the top-k returned by the system can be used for supervised training. We will also demonstrate that, even in the absence of explicit users' feedbacks, the results computed by the tool may still be of high enough quality. We will finally demonstrate that each user query may be associated to several possible interpretations which can be used to reveal the underline database structure.

References

1. Bergamaschi, S., Domnori, E., Guerra, F., Lado, R.T., Velegrakis, Y.: Keyword search over relational databases: a metadata approach. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) SIGMOD Conference, pp. 565–576. ACM, New York (2011)
2. Bergamaschi, S., Guerra, F., Rota, S., Velegrakis, Y.: A Hidden Markov Model Approach to Keyword-based Search over Relational Databases. In: De Troyer, O., et al. (eds.) ER 2011 Workshops. LNCS, vol. 6999, pp. 328–331. Springer, Heidelberg (2011)
3. Bourgeois, F., Lassalle, J.-C.: An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of ACM* 14(12), 802–804 (1971)
4. Li, L., Shang, Y., Shi, H., Zhang, W.: Performance evaluation of hits-based algorithms. In: Hamza, M.H. (ed.) *Communications, Internet, and Information Technology*, pp. 171–176. IASTED/ACTA Press (2002)
5. Seshadri, N., Sundberg, C.-E.: List Viterbi decoding algorithms with applications. *IEEE Transactions on Communications* 42(234), 313–323 (1994)

VirtualEMF: A Model Virtualization Tool

Cauê Clasen, Frédéric Jouault, and Jordi Cabot

AtlanMod Team (INRIA, École des Mines de Nantes, LINA) – France
{caue.avila_clasen, frederic.jouault, jordi.cabot}@inria.fr

Abstract. Specification of complex systems involves several heterogeneous and interrelated models. Model composition is a crucial (and complex) modeling activity that allows combining different system perspectives into a single cross-domain view. Current composition solutions fail to fully address the problem, presenting important limitations concerning efficiency, interoperability, and/or synchronization. To cope with these issues, in this demo we introduce VirtualEMF: a model composition tool based on the concept of a virtual model, i.e., a model that do not hold concrete data, but that redirects all its model manipulation operations to the set of base models from which it was generated.

1 Introduction

Complex systems are usually described by means of a large number of interrelated models, each representing a given aspect of the system at a certain abstraction level. Often, the system view a user needs does not correspond to a single model, but is a cross-domain view in which the necessary information is scattered in several models. This integrated view is provided by the means of model composition which is, in its simplest form, a *modeling process that combines two or more input (contributing) models into a single output (composed) model*. Model composition can be very challenging, due to the heterogeneous nature of models and the complex relationships that can exist between them.

Composition has been extensively studied from various perspectives: its formal semantics [2], composition languages [3], or also targeting different families of models (UML [1], Statecharts [4], database models [5], ...). A commonality of the vast majority of approaches is the fact that the composed model is generated by copying/cloning information from its contributing models, what poses some important limitations in terms of synchronization (updates in the composed model are not propagated to the base ones, or the other way round), creation time (copying many elements is time consuming, and composition must be re-executed every time contributing models are modified), and memory usage (data duplication can be a serious bottleneck when composing large models).

In this demo we present VirtualEMF: a model composition tool that allows overcoming these limitations by applying the concept of *virtual models*, i.e., models that do not hold concrete data (as opposed to *concrete models*), but that access and manipulate the original contributing data contained in other models. The tool was built on top of Eclipse/EMF¹.

¹ Eclipse Modeling Framework: <http://www.eclipse.org/modeling/emf/>

2 Virtual (Composed) Models

In short, a virtual model is a model whose (virtual) elements are proxies to elements contained in other models. It provides to tools/users the *illusion* of working with a regular model while, in fact, all access and manipulation requests are transparently redirected to the set of models from which it was generated (completely integrating contributing model elements into the composed model). Model virtualization brings the following properties to model composition:

- **Interoperability:** virtual models are handled as normal models and therefore they can be exchanged between standard modeling tools;
- **Synchronization:** virtual and contributing models share the same element instances. Thus, updates are automatically propagated in both directions;
- **Faster creation time:** no information cloning is required, and as elements are synchronized, generation is executed only once;
- **Less memory usage:** as virtual models do not contain concrete data, no extra memory is used.

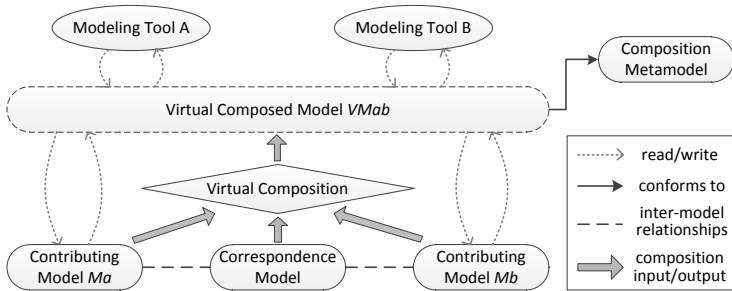


Fig. 1. Model virtualization artefacts

Fig. 1 introduces the main idea of model virtualization and the involved artefacts. Tools (editors, analysis and transformation tools, ...) see and use the *virtual model* as a normal model. The virtual model delegates modeling operations to its set of *contributing models*, locating referenced element(s), and translating them into virtual elements to be used by the tool. Contributing elements are composed at runtime, on an on-demand basis.

Contributing elements (and their properties) can be composed/translated into virtual elements in different manners. Some may be filtered; others, simply reproduced. Another possibility is when contributing elements are related to each other and the virtual element is a combination of them (e.g. as in merge or override relationships). A *correspondence model* links contributing elements and identifies which *translation rule* should be used for composing each element.

A virtual composed model conforms to the same *composition metamodel* a concrete composed model would. This composition metamodel states the core concepts that can populate the composed model.

3 The VirtualEMF Tool

The VirtualEMF tool is an Eclipse plug-in built on top of EMF in order to support the transparent usage of virtual models by EMF-based tools. Here we provide a brief explanation on how to create and use virtual models.

To create a virtual model, users must provide, besides the contributing models to be virtualized (and their metamodels), three elements:

1. A **composition metamodel** that specifies the virtual model concepts. It can be manually defined or be the result of a separate composition process;
2. A **correspondence model** (defined with the AMW² tool) containing *virtual links* that relate contributing elements and specify how they should be composed (i.e. which translation rule is to be applied to them);
3. A **.virtualmodel file** specifying the physical location of the resources involved in the virtual composition process (see Fig. 2).

VMab.virtualmodel
<pre>compositionMetamodel = {\MMab.ecore} contributingMetamodels = {\MMA.ecore, \MMb.ecore} contributingModels = {\Ma.xmi, \Mb.xmi} correspondenceModel = {\MatoMb.amw}</pre>

Fig. 2. A sample virtual model file

The `.virtualmodel` file extension is automatically associated in Eclipse with our specific virtual model handler. Therefore, the loading of a virtual model involves only providing this file as input to any EMF-based tool (e.g., double-clicking a `.virtualmodel` file will automatically trigger VirtualEMF, that loads the virtual model in the standard model viewer). No extra information is required, as the actual data used by the virtual model is retrieved from contributing models and the correspondence model defines how to combine them.

VirtualEMF refines the full set of operations available for a regular model. Thus, usage is also completely transparent (as in Fig. 3). When virtual elements are accessed, VirtualEMF checks which are the referenced contributing element(s), if they are virtually linked, and then translates it(them) accordingly (e.g. with *filter*, *merge*, *override*, *inherit*, or *associate* rules). If no virtual link is specified for a contributing element, it is simply reproduced.

During the demo it will be presented how virtual models are created and used with VirtualEMF, which are its main elements, and how users can define them. This will be shown through several step-by-step examples, with different types of models and specifying different kinds of relationships between them. Finally we will present a set of experiments used to prove that our approach fulfils the desired properties mentioned in section 2.

² *AtlanMod Model Weaver*: <http://www.eclipse.org/gmt/amw/>

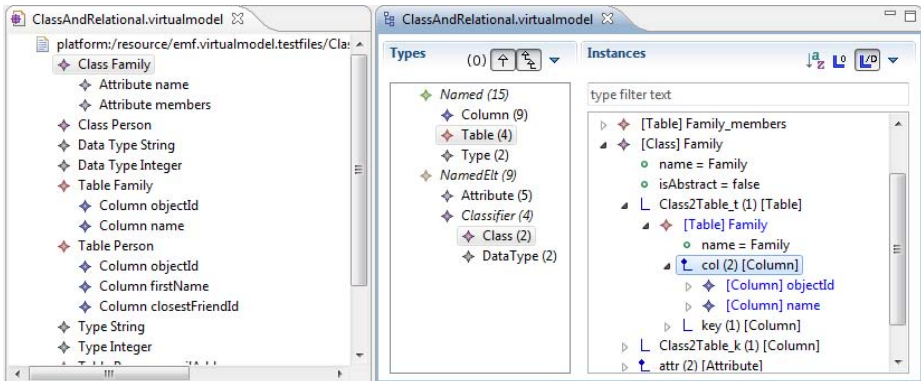


Fig. 3. A virtual model (composition of a UML class model with a relational database model, where the latter derives from the former and virtual associations are used to display traceability links between them) handled in two different EMF tools: Sample Ecore Editor (left) and MoDisco Model Browser (right)

4 Conclusion

Model virtualization is a powerful mechanism that provides a more efficient model composition process, while maintaining perfect synchronization between composition resources. This demo presents VirtualEMF³, our model virtualization tool. The tool is extensible and supports different types of virtual links and new semantics for them. As further work we intend to explore new types of inter-model relationships, and to use state-of-the-art matching techniques to automatically identify relationships and generate the correspondence model. Several experiments have been conducted to prove the scalability of our solution.

References

1. Anwar, A., Ebersold, S., Coulette, B., Nassar, M., Kriouile, A.: A Rule-Driven Approach for composing Viewpoint-oriented Models. *Journal of Object Technology* 9(2), 89–114 (2010)
2. Herrmann, C., Krahn, H., Rumpe, B., Schindler, M., Völkel, S.: An Algebraic View on the Semantics of Model Composition. In: Akehurst, D.H., Vogel, R., Paige, R.F. (eds.) *ECMDA-FA. LNCS*, vol. 4530, pp. 99–113. Springer, Heidelberg (2007)
3. Kolovos, D., Paige, R., Polack, F.: Merging Models with the Epsilon Merging Language (EML). In: Wang, J., Whittle, J., Harel, D., Reggio, G. (eds.) *MoDELS 2006. LNCS*, vol. 4199, pp. 215–229. Springer, Heidelberg (2006)
4. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and Merging of Statecharts Specifications. In: *ICSE 2007*, pp. 54–64. IEEE Computer Society, Los Alamitos (2007)
5. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 334–350 (2001)

³ VirtualEMF website: www.emn.fr/z-info/atlanmod/index.php/VirtualeMF

Towards a Model-Driven Framework for Web Usage Warehouse Development*

Paul Hernández, Octavio Glorio, Irene Garrigós, and Jose-Norberto Mazón

Lucentia – DLSI – University of Alicante, Spain
{phernandez,oglorio,igarrigos,jnmazon}@dlsi.ua.es

Abstract. Analyzing the usage of a website is a key issue for a company to improve decision making regarding the business processes related to the website, or the evolution of the own website. To study the Web usage we need advanced data analysis tools which require the development of a data warehouse to structure data in a multidimensional model. In this paper, we describe two possible scenarios that could arise and we claim that a model-driven approach would be useful for obtaining a multidimensional model in a comprehensive and structured way. This model will drive the development of a data warehouse in order to enhance the analysis of Web usage data: the Web usage warehouse.

Web usage analysis is the process of finding out what users are looking for on the Internet. This information is extremely valuable for understanding how a user “walks” through a website in, thus supporting decision making process.

Commercial tools for Web usage data analysis have some drawbacks: (i) significant limitations performing advanced analytical tasks, (ii) uselessness when trying to understand navigational patterns of users, (iii) inability to integrate and correlate information from different sources, or (iv) unawareness of the conceptual schema of the application. For example, one of the most known analysis tools is Google Analytics (<http://www.google.com/analytics>) which has emerged as a major solution for Web traffic analysis, but it has a limited drill-down capability and there is no way of storing data efficiently. Worse still, the user does not own the data, Google does.

There are several approaches [3,4] that define a multidimensional schema in order to analyze the Web usage by using the Web log data. With these approaches, once the data is structured, it is possible to use OLAP or data mining techniques to analyze the content of the Web logs, tackling the aforementioned problems. However, there is a lack of agreement about a methodological approach in order to detect which would be the most appropriate facts and dimensions: some of them let the analysts decide the required multidimensional elements, while others decide these elements by taking into consideration a specific Web log format. Therefore, the main problem is that the multidimensional elements are informally chosen according to a specific format, so the resulting multidimensional

* This work has been partially supported by the following projects: SERENIDAD (PEII-11-0327-7035) from Castilla-La Mancha Ministry, and MESOLAP (TIN2010-14860) from the Spanish Ministry of Education and Science.

model may be incomplete. Regarding current Web engineering approaches, to the best of our knowledge, none of them [2,11] provide mechanisms for defining a multidimensional model at the same time that the rest of the website in order to represent the Web usage. To overcome these drawbacks, a model-driven framework for developing a Web usage warehouse is proposed considering two different scenarios (see Fig. 1).

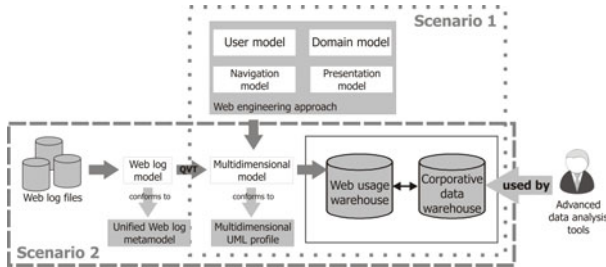


Fig. 1. Model-driven framework for Web usage warehouse development

In our first scenario (*Web usage warehouse within model-driven Web engineering*), several conceptual models have to be defined when designing a website (navigation model, user model, data model, etc.). However, none of these models are intended to represent and understand the Web usage. Therefore, multidimensional concepts (facts, dimensions, hierarchies, etc.) should be identified within the conceptual models of a given application in order to build a Web usage warehouse in an integrated and structured manner. Due to the fact that conceptual models of websites may not be available or out-of-date, within our second scenario (*Web usage warehouse from Web log data*), a Web usage warehouse is developed without requiring these conceptual models, but using Web log files. To this aim, a Web log metamodel is defined which contains the elements and the semantics that allow building a conceptual model from Web log files, which represents, in a static way, the interaction between raw data elements (i.e. the client remote address) and usage concepts (i.e. session, user).

References

1. Ceri, S., Fraternali, P., Bongio, A.: Web modeling language (WebML): a modeling language for designing web sites. *Computer Networks* 33(1-6), 137–157 (2000)
2. Garrigós, I.: A-OOH: Extending web application design with dynamic personalization (2008)
3. Joshi, K.P., Joshi, A., Yesha, Y.: On using a warehouse to analyze web logs. *Distributed and Parallel Databases* 13(2), 161–180 (2003)
4. Lopes, C.T., David, G.: Higher education web information system usage analysis with a data webhouse. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3983, pp. 78–87. Springer, Heidelberg (2006)

CRESKO: Construction of Evidence Repositories for Managing Standards Compliance

Rajwinder Kaur Panesar-Walawege^{1,2}, Torbjørn Skyberg Knutsen^{1,2},
Mehrdad Sabetzadeh¹, and Lionel Briand^{1,2}

¹ Simula Research Laboratory, Lysaker, Norway

² University of Oslo, Oslo, Norway

{rpanesar,mehrdad,briand}@simula.no, torbjsk@ifi.uio.no

Abstract. We describe CRESKO, a tool for Construction of Evidence REpositories for Managing Standards COMpliance. CRESKO draws on Model Driven Engineering (MDE) technologies to generate a database repository schema from the evidence requirements of a given standard, expressed as a UML class diagram. CRESKO in addition generates a web-based user interface for building and manipulating evidence repositories based on the schema. CRESKO is targeted primarily at addressing the tool infrastructure needs for supporting the collection and management of safety evidence data. A systematic treatment of evidence information is a key prerequisite for demonstration of compliance to safety standards, such as IEC 61508, during the safety certification process.

Keywords: Safety Certification, Conceptual Modeling, MDE, IEC 61508.

1 Introduction

Safety critical systems are typically subject to safety certification based on recognized safety standards as a way to ensure that these systems do not pose undue risks to people, property, or the environment. A key prerequisite for demonstrating compliance to safety standards is collecting structured evidence in support of safety claims. Standards are often written in natural language and are open to subjective interpretation. This makes it important to develop a precise and explicit interpretation of the evidence requirements of a given standard. In previous work [43], we have proposed conceptual modeling for formalizing the evidence requirements of safety standards. This approach on the one hand helps develop a shared understanding of the standards and on the other hand, provides a basis for the automation of various evidence collection and management tasks.

In this paper, we describe CRESKO, a flexible tool infrastructure for creating repositories to store, query, and manipulate standards compliance evidence. Additionally, CRESKO generates a web-based user interface for interacting with these repositories. Our work was prompted by an observed need during our collaboration with companies requiring IEC 61508 compliance. In particular, we observed that little infrastructure support has been developed to date to support management of safety evidence based on a specific standard. This issue has

also been noted in the literature as an important gap in the safety certification process [2,5]. While CRESKO is general and can be used in conjunction with different standards, we ground our discussion in this paper on IEC 61508, which is a key standard for safety certification of programmable electronic systems.

In the rest of this paper, we will describe the key components of CRESKO. For the actual demonstration, we will follow, and expand where necessary, our presentation in this paper. Specifically, the demo will begin with motivational material – similar to this paper’s introduction and augmented with snippets of the conceptual model in [4]. We then go on to describe the overall architecture of the tool, as shown in Figure 1. In the next step, we will use a combination of pre-recorded and live demonstrations to illustrate the main functions of the tool, discussed in Sections 2. Finally, as we outline in Section 3, our demonstration will provide information about the tool’s implementation based on our existing documentation [1], and give details about availability.

2 Tool Overview

Users can interact with CRESKO in two roles: the administrator and general user. The administrator is responsible for importing the conceptual model into CRESKO, running the transformations and setting up and starting the web server. Once the server is started, the general users – typically experts from the supplier company or certification body – can add, view and manipulate the data in the database. In this section we provide an overview of the main components of CRESKO as shown in Figure 1(a).

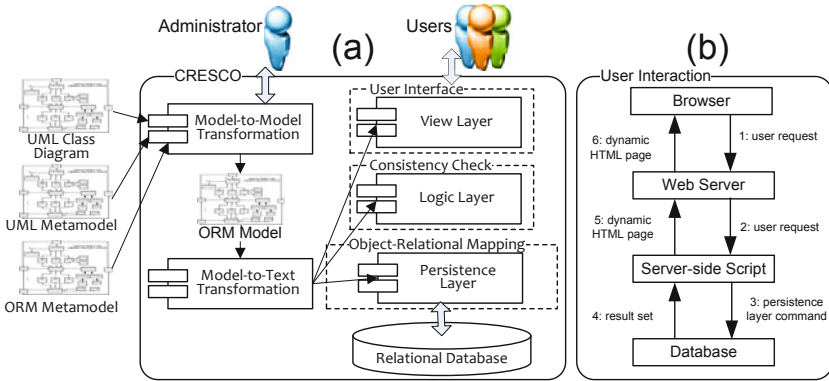


Fig. 1. Components of CRESKO and How the User Interacts With It

2.1 Generation of Database Schema and UI

The generation of the database and the user interface code involves two steps: a model-to-model (M2M) transformation and a model-to-text (M2T) transformation. The M2M transformation takes as input a conceptual model of a standard

in the form of a *UML class diagram* [6]. This model can be created in any UML tool and then imported into CRESCO. An in-depth description of the conceptual model is beyond the scope of this demonstration paper – further details are available in [4]. The M2M transformation makes use of the UML meta-model [6] and a meta-model for an object-relational mapping (ORM) that we have created – see [1]. This ORM meta-model enables the storage (in a relational database) of objects that have been created based on a UML class diagram. The ORM meta-model includes a database schema (with tables, columns, foreign keys, etc) and object-oriented concepts, mainly generalization. The M2M transformation iterates over the conceptual model and transforms it into a model that corresponds to the ORM meta-model.

The user interface is generated from the ORM model created during the M2M transformation. The M2T transformation iterates over the elements of the ORM model and generates the database implementation as well as all the code for accessing and updating the database via a web interface. The generated code is a combination of server-side Java code and HTML (see Section 3). Figure 1(b) shows how the user interaction is processed via the generated code. Figure 2 shows the user interface generated. The left hand pane lists all the tables that have been generated and the right hand pane is used to manipulate the rows in a selected table. The 'New' button shown is used to add a new row into the selected table. Figure 2 shows the table for the concept of **Agent**, who is an entity that carries out an activity required during system development. An **Activity** is a unit of behavior with specific input and output **Artifacts**. Each activity utilizes certain **Techniques** to arrive at its desired output and requires certain kind of **Competence** by the agents performing it. The agent itself can be either an individual or an organization and is identified by the type of role it plays. In CRESCO, one can: (1) create instances of concepts such as **Agent**, **Activity**, **Artifact**, (2) fill out their attributes, (3) and establish the links between the concept instances. For illustration, we show in Figure 2, the addition of an agent record into the **Agent** table.

2.2 Consistency Checking

The consistency check is a means of verifying that the state of the database is in accordance with the multiplicity constraints defined in the conceptual model. The consistency check is derived from the multiplicities of UML associations. We have chosen not to preserve the multiplicities in the database schema, where all associations are represented as many-to-many. This flexibility is required so that we can tolerate inconsistencies during the construction of the database. Trying to maintain consistency at all time would be intrusive, as this would enforce

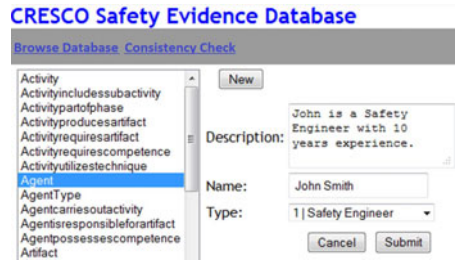


Fig. 2. CRESCO User Interface

an unnecessary order on how the evidence items have to be entered. While our choice allows more freedom for the user when adding entries in the database, it also calls for the implementation of a consistency checker, to verify that the data in the database is in accordance with the constraints defined in the UML class diagram. For example, an `Activity` must have at least one `Agent` who is responsible for carrying out this activity (defined in the `Agentcarriesoutactivity` table shown in Figure 2). Such constraints are checked by CRESCO's consistency checker and any violations are highlighted to the user for further investigation.

3 Implementation and Availability

CRESCO is implemented in Eclipse for Java Enterprise Edition. We use two plugins, one for Kermeta that is used for the M2M transformations and the other is MOFScript for the M2T transformations. The M2M and M2T code are approx. 800 and 1500 lines, respectively. The total number of lines of code generated depends on the size of the input conceptual model. For the IEC 61508 model, the resulting code was in excess of 20,000 lines. Hence, significant manual effort can be saved by applying CRESCO. We use Apache Derby as the underlying database which is accessed by the Java code via Hibernate. The user interface for populating the database is via the web and we use Apache Tomcat as the web server and JavaServer Pages, Apache Struts and JavaScript to present and manipulate the objects residing in the database as well as to provide the navigation of the user interface. For our demonstration, we will present the import process of the conceptual model, the execution of the the two transformations and the user-interaction with the web-based user-interface. Due to space restrictions, we do not describe the technologies underlying CRESCO. See [1] for details and references. CRESCO is freely available at <http://home.simula.no/~rpanesar/cresco/>.

4 Conclusion and Future Work

We presented CRESCO a tool for the generation and manipulation of evidence repositories for demonstrating standards compliance during certification. CRESCO provides a centralized repository for keeping diverse data which, in the current state of practice, is often not collected systematically and needs to be extracted and amalgamated during certification. Our goal was to show feasibility via a coherent combination of existing open-source technologies. While our current tool provides a flexible infrastructure for managing compliance evidence, further work is required to turn it into a tool that can be deployed in a production environment. In particular, we are considering adding more sophisticated query facilities such that complex queries can be posed as well as professional reporting facilities in order to extract data from the database to create reports that can be directly given to the certification body.

References

1. Knutsen, T.: Construction of information repositories for managing standards compliance evidence (2011) Master Thesis, University of Oslo, <http://vefur.simula.no/~rpanesar/cresco/knutsen.pdf>
2. Lewis, R.: Safety case development as an information modelling problem. In: Safety-Critical Systems: Problems, Process and Practice, pp. 183–193. Springer, Heidelberg (2009)
3. Panesar-Walawege, R.K., Sabetzadeh, M., Briand, L.: Using UML profiles for sector-specific tailoring of safety evidence information. In: De Troyer, O., et al. (eds.) ER 2011 Workshops. LNCS, vol. 6999, Springer, Heidelberg (2011)
4. Panesar-Walawege, R.K., Sabetzadeh, M., Briand, L., Coq, T.: Characterizing the chain of evidence for software safety cases: A conceptual model based on the iec 61508 standard. In: ICST 2010, pp. 335–344 (2010)
5. Redmill, F.: Installing IEC 61508 and supporting its users – nine necessities. In: 5th Australian Workshop on Safety Critical Systems and Software (2000)
6. UML 2.0 Superstructure Specification (August 2005)

Modeling Approach for Business Networks with an Integration and Business Perspective

Daniel Ritter and Ankur Bhatt

SAP AG, Technology Development – Process and Network Integration,
Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany

{daniel.ritter, ankur.bhatt}@sap.com

<http://www.sap.com>

Abstract. Business Network Management (BNM) allows enterprises to manage their application integration and partner networks by making technical integration, business and social aspects visible within a network view and set them into context to each other. This allows various personas, like business user, integration expert and IT support to analyze, operate and develop business processes by collaborating on these contexts. Defining a model sufficient to represent the BNM domain with different layers of abstraction, from business to technical views and perspectives of network-structured data requires a standard, human- and machine readable notation. Modeling of real-world technical and business artifacts is widely addressed by UML [5], SCA [4] which cover parts of these requirements for BNM. However, none of them accounts for the combined business and technical nature of most enterprises, nor of the semantic network and data linking aspects. In this paper, we present design decisions for a model based on BPMN 2.0, that is sufficient for BNM and represents inter-related business and technical perspectives within the enterprise network.

Keywords: Business Network, Modeling Approach, BPMN 2.0.

1 Introduction

Nowadays, enterprises participate in value chains of suppliers and customers while competing in business networks rather than being isolated. To remain competitive, e.g. by quickly implementing new process variants or proactively identify flaws within existing processes, enterprises need visibility into their business network, the relevant applications and the processes. Business Network Management is the capability of managing intra and inter enterprise networks.

2 Modeling a Business Network

A model for business network needs to represent the existing enterprise information models like process, integration, applications, and to interrelate them into a network of linked data. The model shall serve as a visual representation and

exchange format, thus requiring a human and computer readable notation. It shall allow aggregation of and drill-into network entities with extension points for information models. The notation has to be a well-established standard covering requirements for representing the business network. For technical aspects SCA [4] and UML [5] provide a solid foundation and are both human readable. SCA defines *Components* and *Wires* to represent a network and out-performs UML by explicitly defining *Services* and *References* with various bindings. This is similar to *Pools*, i.e. *Participants*, and *Flows* with BPMN 2.0 *ConversationDiagrams* and service extension point package (see M. Owen et al [1]). For grouping of flows, BPMN defines *Conversations*, *SubConversations*, resp., while SCA allows for component nesting with *Composites*. However, the requirement to visualize and link to real-world entities like applications and semantically relate the context end-to-end, e.g. from process to system is possible in BPMN. The perspectives are represented with specializations of BPMN *Participant* and *MessageFlow* while linked via *ParticipantAssociation* and a new extension *FlowAssociation*. Thus using BPMN 2.0 as foundation allows all personas to work with the same model, and serve as a standardized exchange format. BPMN is a standard for defining, visualising and exchanging business procedures within and across enterprises and is widely used within disciplines like BPM. Business networks pose additional challenges on the model, like structural support for large networks, which need to be addressed with extensions to BPMN 2.0.

3 Conclusions

In this paper, we presented the design decisions to use BPMN for a network domain, BNM. We discuss the requirements for a model in this domain and show that a network model can be based on BPMN 2.0 while extending it for network specific entities and attributes. The specific extensions to BPMN 2.0 are discussed in detail in [2].

Acknowledgments. We thank Ivana Trickovic for support on BPMN 2.0 and Gunther Rothermel for guidance and sponsorship.

References

1. Owen, M., Stuecka, R.: BPMN und die Modellierung von Geschäftsprozessen. Whitepaper, Telelogic (2006)
2. Ritter, D., Ackermann, J., Bhatt, A., Hoffmann, F. O.: Building a Business Graph System and Network Integration Model based on BPMN. In: 3rd International Workshop on BPMN, Luzern (accepted, 2011)
3. Specification of Business Process Modeling Notation version 2.0 (BPMN 2.0), <http://www.omg.org/spec/BPMN/2.0/PDF>
4. Service Component Architecture (SCA), <http://www.osoa.org/display/Main/Service+Component+Architecture+Home>
5. Unified Modeling Language (UML), <http://www.uml.org/>

Mosto: Generating SPARQL Executable Mappings between Ontologies^{*,**}

Carlos R. Rivero, Inma Hernández, David Ruiz, and Rafael Corchuelo

University of Sevilla, Spain

{carlosrivero,inmahernandez,druiz,corchu}@us.es

Abstract. Data translation is an integration task that aims at populating a target model with data of a source model, which is usually performed by means of mappings. To reduce costs, there are some techniques to automatically generate executable mappings in a given query language, which are executed using a query engine to perform the data translation task. Unfortunately, current approaches to automatically generate executable mappings are based on nested relational models, which cannot be straightforwardly applied to semantic-web ontologies due to some differences between both models. In this paper, we present Mosto, a tool to perform the data translation using automatically generated SPARQL executable mappings. In this demo, ER attendees will have an opportunity to test this automatic generation when performing the data translation task between two different versions of the DBpedia ontology.

Keywords: Information Integration, Data Translation, Semantic-web Ontologies, SPARQL executable mappings.

1 Introduction

Data translation is an integration task that aims at populating a target model with data of a source model, which is becoming a major research task in the semantic-web context [5,12]. Mediators are pieces of software that help perform this task, which rely on mappings that relate source and target models [4].

To reduce integration costs, some techniques automatically generate a set of uninterpreted mappings, a.k.a. correspondences, which must be interpreted to perform the data translation task [4]. They are hints that usually relate a source entity with a target entity, although they may be more complex [4]. The main issue regarding correspondences is that there is not a unique interpretation of them, i.e., different approaches interpret correspondences in different ways [2].

* Supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

** An implementation and examples regarding this paper are available at:
<http://tdg-seville.info/carlosrivero/Mosto>

Executable mappings encode an interpretation of correspondences in a given query language [6,11]. These mappings are executed by means of a query engine to perform the data translation task. The main benefit of using these mappings is that, instead of relying on ad-hoc programs that are difficult to create and maintain, a query engine performs the data translation task [6].

In the bibliography, Ressler et al. [10] devised a visual tool to specify hand-crafted SPARQL executable mappings. However, it is well-known that hand-crafted executable mappings increase integration costs [8]. Furthermore, there are a number of visual tools to specify correspondences between source and target models, such as Clio, Muse, or Clip [1,6,9]. After specifying the correspondences, these tools automatically generate a set of executable mappings based on them. Unfortunately, these tools focus on nested relational models, and they are not straightforwardly applicable to semantic-web ontologies due to a number of inherent differences between them [7,11].

In this paper, we present *Mosto*, a tool to perform the data translation task between OWL ontologies using automatically generated SPARQL executable mappings. To the best of our knowledge, this is the first tool to automatically generate executable mappings in the semantic-web context. To describe it, we use a demo scenario integrating two different versions of the DBpedia ontology [3].

This paper is organised as follows: Section 2 describes the system, and Section 3 deals with the demo that ER attendees will have the opportunity to test.

2 Mosto

In this section, we present *Mosto*, our tool to perform the data translation task between two OWL ontologies by means of SPARQL executable mappings. Performing this task in our tool comprises four steps, namely: 1) Selecting source and target ontologies; 2) Specifying restrictions and correspondences; 3) Generating SPARQL executable mappings; and 4) Executing SPARQL executable mappings. These steps are described in the rest of the section.

The first step deals with the selection of source and target ontologies to be integrated. In our demo scenario (cf. Figure 1), we integrate DBpedia ontology v3.2 with DBpedia ontology v3.6, which are shown in a tree-based notation in which classes, data properties and object properties are represented by circles, squares and pentagons, respectively. Note that subclasses are represented between brackets, e.g., *dbp:Artist* is subclass of *dbp:Person* is represented as “*dbp:Artist [dbp:Person]*”. In addition, the domain of a property is represented by nesting the property in a class, and the range is represented between ‘<’ and ‘>’, e.g., the domain of *dbp:director* is *dbp:Film* and its range is *dbp:Person*. After selecting the ontologies, *Mosto* extracts a number of implicit restrictions, which are restrictions that are due to the modelling language of source and target ontologies, in our case, the OWL ontology language.

In the second step, the user specifies explicit restrictions in the source and target ontologies, and correspondences between them. Explicit restrictions are necessary to adapt existing ontologies to the requirements of a specific scenario,

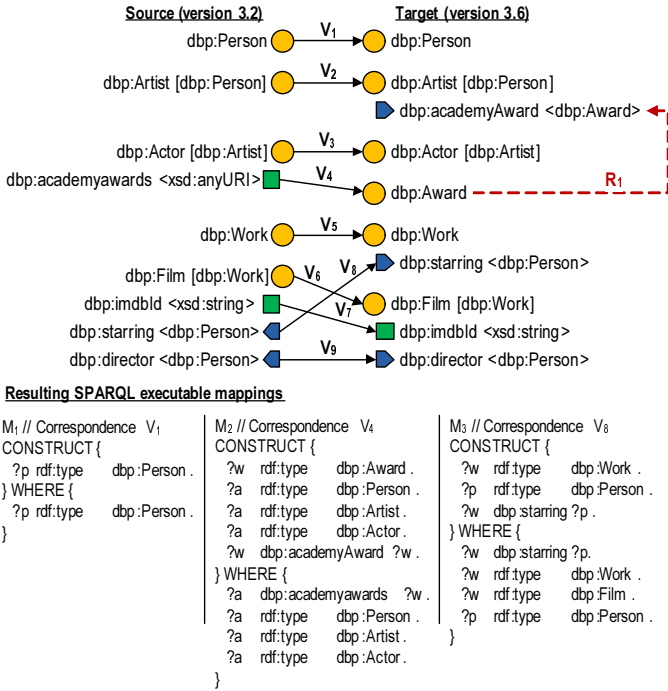


Fig. 1. Evolution in DBpedia (demo scenario)

e.g., R_1 is an explicit restriction by which *dbp:academyAward* has a minimal cardinality of one with respect to *dbp:Award*. Correspondences are represented as arrows in Figure 1, e.g., V_8 is a class correspondence that relates *dbp:starring* object property in both source and target ontology versions. Note that Mosto allows to load previously defined restrictions and/or correspondences, or to visually generate new restrictions and/or correspondences according to user preferences.

In the third step, Mosto generates a set of SPARQL executable mappings using (implicit and explicit) restrictions and correspondences. The technique to automatically generate these mappings is described in [11], which is based on clustering those source restrictions, target restrictions and other correspondences that we must take into account to produce coherent target data when performing the data translation task. Note that, if we use each correspondence in isolation to translate data, we may produce incoherent target data, e.g., correspondence V_8 cannot be used in isolation since we do not know how to translate the domain and range of *dbp:starring*. Therefore, our technique clusters V_1 , V_5 and V_8 , which translate the domain and range of *dbp:starring*, respectively. In addition, every cluster is transformed into a SPARQL executable mapping that encode an interpretation of correspondences. Figure 1 shows three examples of SPARQL executable mappings generated with our tool: M_1 , M_2 and M_3 are the resulting executable mappings of correspondences V_1 , V_4 and V_8 , respectively.

Finally, in the fourth step, Mosto is able to perform the data translation task by executing the previously generated SPARQL executable mappings over the source ontology to produce instances of the target ontology. Note that, thanks to our SPARQL executable mappings, we are able to automatically translate the data from a previous version of an ontology to a new version.

3 The Demo

In this demo, ER attendees will have an opportunity to use Mosto to test the automatic generation of SPARQL executable mappings using our demo scenario, which integrates different versions of the DBpedia ontology. We will show how the addition or removal of correspondences and restrictions affect the resulting executable mappings. Furthermore, we will perform the data translation task using these mappings, and check whether resulting target data are as expected.

Expected evidences in our demo scenario are the following, namely: 1) the time to generate executable mappings is less than one second; 2) Mosto facilitates the specification of restrictions and correspondences in complex scenarios; and 3) the resulting target data are coherent with expected results.

References

1. Alexe, B., Chiticariu, L., Miller, R.J., Tan, W.C.: Muse: Mapping understanding and design by example. In: ICDE, pp. 10–19 (2008)
2. Bernstein, P.A., Melnik, S.: Model management 2.0: manipulating richer mappings. In: SIGMOD, pp. 1–12 (2007)
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. *J. Web Sem.* (2009)
4. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer, Heidelberg (2007)
5. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336(1), 89–124 (2005)
6. Haas, L.M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio grows up: from research prototype to industrial tool. In: SIGMOD, pp. 805–810 (2005)
7. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. *J. Web Sem.* 7(2), 74–89 (2009)
8. Petropoulos, M., Deutsch, A., Papakonstantinou, Y., Katsis, Y.: Exporting and interactively querying web service-accessed sources: The CLIDE system. *ACM Trans. Database Syst.* 32(4) (2007)
9. Raffio, A., Braga, D., Ceri, S., Papotti, P., Hernández, M.A.: Clip: a tool for mapping hierarchical schemas. In: SIGMOD, pp. 1271–1274 (2008)
10. Ressler, J., Dean, M., Benson, E., Dorner, E., Morris, C.: Application of ontology translation. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 830–842. Springer, Heidelberg (2007)
11. Rivero, C.R., Hernández, I., Ruiz, D., Corchuelo, R.: Generating SPARQL executable mappings to integrate ontologies. In: Jeusfeld, M., et al. (eds.) ER 2011. LNCS, vol. 6998, pp. 118–131. Springer, Heidelberg (2011)
12. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. *IEEE Int. Sys.* 21(3), 96–101 (2006)

The CSTL Processor: A Tool for Automated Conceptual Schema Testing

Albert Tort, Antoni Olivé, and Maria-Ribera Sancho

Universitat Politècnica de Catalunya – Barcelona Tech
{atort,olive,ribera}@essi.upc.edu

Abstract. In this demonstration paper, we present the CSTL Processor, a tool to support the validation of two fundamental quality properties of conceptual schemas (correctness and completeness) by testing. The CSTL Processor supports the management, execution and automatic computation of the verdicts of test cases which formalize stakeholders' needs and expectations.

Keywords: Conceptual modeling, Validation, Quality, Automated testing, UML/OCL.

1 Introduction

Two fundamental quality properties of conceptual schemas are correctness (i.e. all the defined knowledge is true for the domain) and completeness (i.e. all the relevant knowledge is defined in the conceptual schema) [1]. The validation of these properties is still an open challenge in conceptual modeling [2].

In [3], we proposed a novel environment for testing conceptual schemas. The main purpose of conceptual schema testing is the validation of conceptual schemas according to stakeholders' needs and expectations. Conceptual schemas can be tested if (1) the conceptual schema is specified in an executable form, and (2) a representative set of concrete scenarios are formalized as test cases.

In this paper, we present the CSTL Processor [4], a tool that supports the execution of test sets written in the Conceptual Schema Testing Language (CSTL) [3]. The CSTL Processor makes the proposed testing environment feasible in practice.

This tool may be used in different application contexts in which UML/OCL conceptual schemas may be tested. The CSTL Processor supports test-last validation (in which correctness and completeness are checked by testing after the schema definition) or test-first development of conceptual schemas (in which the elicitation and definition is driven by a set of test cases). Our testing environment proposes the use of automated tests [5], which include assertions about the expected results that may be automatically checked. This is an essential feature in order to allow regression testing [5].

In the next section, we present the CSTL Processor and its components. In Section 3, we reference example applications which will be illustrated in the demonstration and complementary case studies, tutorials and documentation.

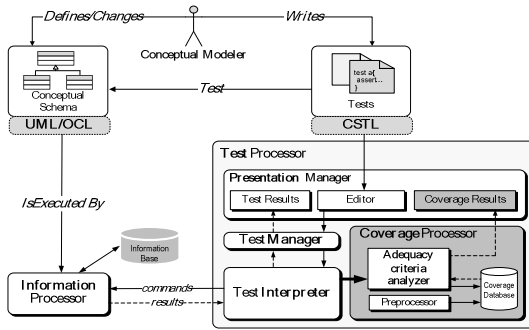


Fig. 1. The CSTL Processor testing environment

2 The CSTL Processor

The *CSTL Processor* is a research prototype which works as a standalone application and supports the specification, management and execution of automated tests of executable conceptual schemas.

In the implemented release used in the demonstration presented in this paper, schemas are defined in UML/OCL. An extended USE [6] syntax is used to specify the conceptual schemas under test in an executable form. Automated conceptual test cases are written in the *Conceptual Schema Testing Language* (CSTL). Test cases consist of sequences of expected Information Base (IB) states (which may change through the occurrence of events) and assertions about them.

The main features of the *CSTL Processor* are: (1) The definition of executable conceptual schemas under test; (2) the definition and management of CSTL test programs; (3) the execution of the test set and the automated computation of its verdicts, including reports of error and failing information; and (4) the automatic analysis of testing coverage according to a basic set of testing adequacy criteria.

Figure 1 shows the main components of the CSTL Processor environment. The execution of conceptual schemas is done by the *Information Processor*, while the execution of test programs is done by the *Test Processor*. Moreover, automatic coverage analysis for a basic set of test adequacy criteria is provided by the *Coverage Processor*. In the following, we briefly present each main tool component.

2.1 The Information Processor

The information processor is able to setup IB states by creating, deleting and changing entities, attributes and associations. It also evaluates OCL expressions of test assertions. We implemented the information processor extending the USE [6] core in order to be able to deal with several new language features such as derived attributes, default values, multiplicities in attributes, domain events, temporal constraints, initial integrity constraints and generalization sets.



Fig. 2. Test execution screenshot

2.2 The Test Processor

The test processor implements the execution of the test cases and consists of the presentation manager, the test manager and the test interpreter.

The test manager stores the CSTL programs in order to make it possible to execute the test set at any time. When the conceptual modeler requests the execution of the test programs, the test manager requests the test interpreter to execute them. The test manager also keeps track of the test results and maintains test statistics.

The test interpreter reads and executes CSTL programs. For each test case, the interpreter sets up the common fixture (if any), executes the statements of each test case and computes the verdicts. The interpreter invokes the services of the information processor to build the IB states according to each test case and check the specified assertions.

The presentation manager provides means for writing CSTL test programs and for displaying the results of their execution. Built-in editors are provided for the definition of the conceptual schema, its methods and the test programs. Moreover, after each execution of the test set, test programs verdicts are displayed. Figure 2 shows a screenshot of the verdicts presentation screen. The test processor indicates the number of the lines where test cases have failed and gives an explanation of the failure in natural language.

2.3 The Coverage Processor

The *Coverage Processor* provides automatic coverage analysis according to a basic set of test adequacy criteria which guarantees the relevance and the satisfiability of the defined schema elements. It consists of the *preprocessor*, the *coverage database*, and the *adequacy criteria analyzer*. The implemented criteria allows to automatically analyze which base types, derived types, valid type configurations and event occurrences have been tested in at least one valid execution of a test case.

The *preprocessor* initializes the *coverage database*, which maintains the set of covered elements for each test adequacy criterion. The *test interpreter* communicates information about the tests execution to the *adequacy criteria analyzer* which is the responsible of updating the *coverage database* according to the defined criteria.

After the execution of all test programs, the *adequacy criteria analyzer* queries the *coverage database* in order to obtain the sets of covered and uncovered elements for each criterion and computes statistical information about the coverage results.

3 Case Studies, Examples and Documentation

The CSTL Processor has been developed in the context of Design Research [7]. The tool has been experimentally used in the testing of the conceptual schema of two e-commerce systems (*osCommerce* [8] and *Magento*), in the test-driven development of the schema of a bowling game system [9] and in the reverse engineering development of the schema of the *osTicket* System [10]. Moreover, it is being used in a requirements engineering course, in which groups of master students are challenged to develop the conceptual schema of a new information system assisted by the tool.

Video tutorials with the examples which will be used in the demonstration session may be found in the project website [4]. Additional information and resources such as source files, screenshots and complementary documentation may also be found in [4].

References

1. Lindland, O.I., Sindre, G., Solvberg, A.: Understanding Quality in Conceptual Modeling. *IEEE Software* 11(2), 42–49 (1994)
2. Olivé, A.: *Conceptual Modeling of Information Systems*. Springer, Berlin (2007)
3. Tort, A., Olivé, A.: An approach to testing conceptual schemas. *Data Knowl.Eng.* 69(6), 598–618 (2010)
4. Tort, A.: The CSTL Processor project website, <http://www.essi.upc.edu/~atort/cstlprocessor>
5. Janzen, D., Saiedian, H.: Test-Driven Development: Concepts, Taxonomy, and Future Direction. *Computer* 38(9), 43–50 (2005)
6. Gogolla, M., Bohling, J., Richters, M.: Validating UML and OCL Models in USE by Automatic Snapshot Generation. *Software & Systems Modeling* 4(4), 386–398 (2005)
7. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *Mis Quarterly*, 75–105 (2004)
8. Tort, A.: Testing the osCommerce Conceptual Schema by Using CSTL. Research Report UPC (2009), <http://hdl.handle.net/2117/6289>
9. Tort, A.: Development of the conceptual schema of a bowling game system by applying TDCM. Research Report UPC (2011), <http://hdl.handle.net/2117/11196>
10. Tort, A.: Development of the conceptual schema of the osTicket system by applying TDCM. Research Report UPC (2011), <http://hdl.handle.net/2117/12369>

A Tool for Filtering Large Conceptual Schemas

Antonio Villegas, Maria-Ribera Sancho, and Antoni Olivé

Department of Service and Information System Engineering
Polytechnic University of Catalonia
Barcelona, Spain
{avillegas,ribera,olive}@essi.upc.edu

Abstract. The wealth of knowledge the conceptual schemas of many real-world information systems contain makes them very useful to their potential target audience. However, the sheer size of those schemas makes it difficult to extract knowledge from them. There are many information system development activities in which people needs to get a piece of the knowledge contained in a large conceptual schema. We present an information filtering tool in which a user focuses on one or more entity types of interest for her task at hand, and the tool automatically filters the schema in order to obtain a reduced conceptual schema including a set of entity and relationship types (and other knowledge) relevant to that task.

Keywords: Large Schemas, Filtering, Entity Types, Importance.

1 Introduction

The conceptual schemas of many real-world information systems are too large to be easily managed or understood. There are many information system development activities in which people needs to get a piece of the knowledge contained in a conceptual schema. For example, a conceptual modeler needs to check with a domain expert that the knowledge is correct, a database designer needs to implement that knowledge into a relational database, a software tester needs to write tests checking that the knowledge has been correctly implemented in the system components, or a member of the maintenance team needs to change that knowledge. Currently, there is a lack of computer support to make conceptual schemas usable for the goal of knowledge extraction.

Information filtering [3] is a rapidly evolving field to handle large information flows. The aim of information filtering is to expose users only to information that is relevant to them. We present an interactive tool in which the user specifies one or more concepts of interest and the tool automatically provides a (smaller) subset of the knowledge contained in the conceptual schema that is likely to be relevant. The user may then start another interaction with different concepts, until she has obtained all knowledge of interest. We presented the theoretical background behind this tool in [4,5].

2 The Filtering Process

In this section we describe how a large conceptual schema can be filtered using the method implemented by our tool, which corresponds to the demonstration we intend to perform. The main idea is to extract a reduced and self-contained view from the large schema, that is, a filtered conceptual schema with the knowledge of interest to the user. Figure 1 presents the three steps of our filtering process.

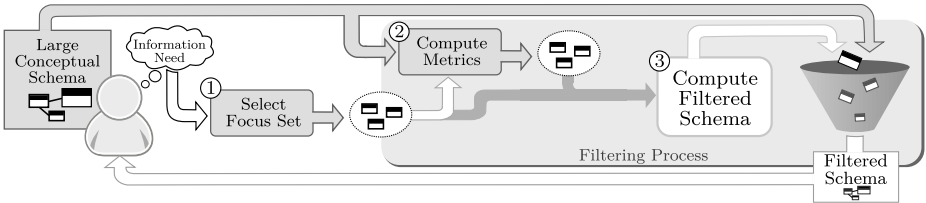


Fig. 1. Method overview

The first step consists in preparing the required information to filter the large schema according to the specific needs of the user. Basically, the user focus on a set of entity types she is interested in and our method surrounds them with additional related knowledge from the large schema. Therefore, it is mandatory for the user to select a non-empty initial focus set of entity types of interest.

During the second step our method computes the required metrics to automatically select the most interesting entity types to extend the knowledge selected in the focus set of the first step. The main goal of these metrics is to discover those entity types that are relevant in the schema but also that are close (in terms of structural distance over schema) to the entity types of the focus set. We presented a detailed definition of such metrics in [4].

Finally, the last step receives the set of most interesting entity types selected in the previous step and puts it together with the entity types of the focus set in order to create a filtered conceptual schema with the entity types of both sets. The main goal of this step consists in filtering information from the original schema involving entity types in the filtered schema. To achieve this goal, the method explores the relationships and generalizations/specializations in the original schema that are defined between those entity types and includes them in the filtered schema to obtain a connected schema.

3 The Filtering Tool

Our filtering tool is developed as a web client that interacts with a web service following the SOAP protocol. The filtering web service we have developed makes use of a modified version of the core of the USE tool [2] to load and maintain the knowledge of the large schema the user wants to explore. In our demonstration the large schema to filter consists of a subset of the HL7 V3 schemas [1] containing more than 2,500 entity types from healthcare domains [5].

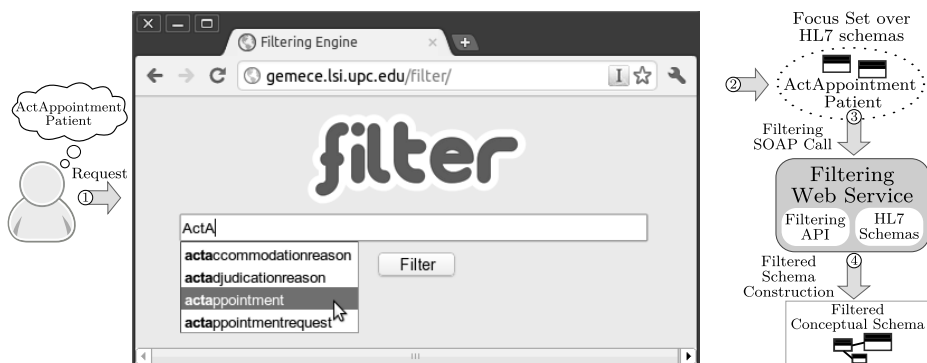


Fig. 2. Request of a user to our filtering tool

Figure 2 depicts the main components that participate in a user request to our filtering tool. The user writes in the search field the names of the entity types she is interested in. Our web client automatically suggests names while the user is writing to simplify that task and to help her discovering additional entity types. In the example of Fig. 2 the user focuses on the entity types *ActAppointment* and *Patient*. Once the request is complete our web client processes the focus set and uses the filtering API of the web service through a SOAP call. The web service analyses the request and constructs the related filtered conceptual schema following the filtering process described in the previous section.

Figure 3 shows the components of the response. The reduced schema produced by our web service is an XMI file containing 8 entity types. In order to increase the understandability of the schema we make use of an external service (<http://yum1.me>) to transform the filtered schema from a textual representation to a graphical one. As a result, the user can rapidly comprehend from the schema of Fig. 3 that *SubjectOfActAppointment* connects the entity types *ActAppointment* and *Patient*, which means that in the HL7 V3 schemas a patient is the subject of a medical appointment. Subsequently, the user can start again the cycle with a new request if required.

4 Summary

We have presented a tool to assist users to deal with large conceptual schemas that allows to focus on a set of entity types of interest and automatically obtains a reduced view of the schema in connection with that focus. Our implementation as a web service provides interoperability and simplifies the interaction with users. A preliminary version of the filtering tool can be found in <http://gемеce.lsi.upc.edu/filter/>.

Our immediate plans include the improvement of our tool by adding a more dynamic view of the filtered schema instead of the static image obtained by the present external service. As a result, we are introducing more interactive features such as selection of schema elements and new filtering interactions from that selection.

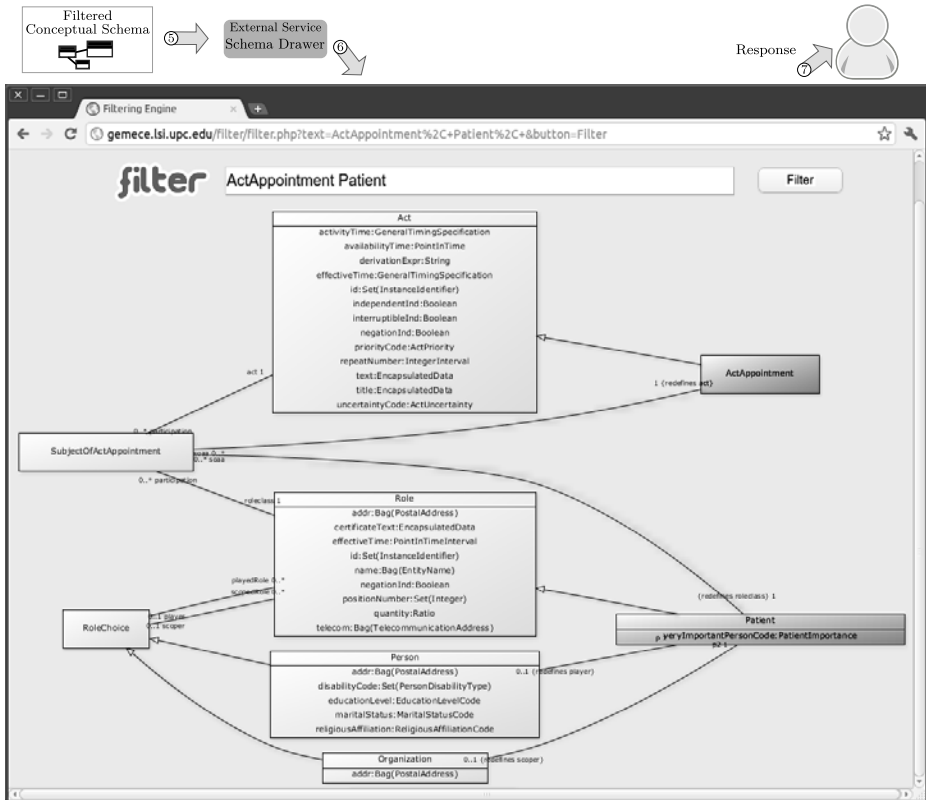


Fig. 3. Response of our filtering tool to the user

Acknowledgements. This work has been partly supported by the Ministerio de Ciencia y Tecnologia and FEDER under project TIN2008-00444/TIN, Grupo Consolidado, and by Polytechnic University of Catalonia under FPI-UPC program.

References

1. Beeler, G.W.: HL7 version 3 – an object-oriented methodology for collaborative standards development. *International Journal of Medical Informatics* 48(1-3), 151–161 (1998)
2. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-based specification environment for validating UML and OCL. *Science of Computer Programming* (2007)
3. Hanani, U., Shapira, B., Shoval, P.: Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction* 11(3), 203–259 (2001)
4. Villegas, A., Olivé, A.: A method for filtering large conceptual schemas. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) *ER 2010*. LNCS, vol. 6412, pp. 247–260. Springer, Heidelberg (2010)
5. Villegas, A., Olive, A., Vilalta, J.: Improving the usability of HL7 information models by automatic filtering. In: *IEEE 6th World Congress on Services*, pp. 16–23 (2010)

Preface to the Industrial Track

The aim of the ER'11 Industrial Track was to serve as a forum for high quality presentations on innovative commercial software, systems, and services for all facets of conceptual modeling methodologies and technologies as described in the list of topics of the ER 2011 conference. We strongly believe that bringing together researchers and practitioners is important for the progress and success of research on conceptual modeling. We do hope that this track will become stronger year by year and will serve as an excellent opportunity to discuss current practices and modern and future market trends and needs.

The 2011 edition formed two interesting sessions on advanced and novel conceptual model applications.

The first session included three papers on business intelligence applications. The first two papers present tools that assist the data warehouse designer. QBX is a case tool that facilitates data mart design and deployment. TARGIT BI Suite assists the designer to add associations between measures and dimensions to a traditional multidimensional cube model and facilitates a process where users are able to ask questions to a business intelligence system without the constraints of a traditional system. The third paper presents a tool that implements an entity resolution method for topic-centered expert identification based on bottom-up mining of online sources.

The second session included three papers on emerging industrial applications of conceptual modeling. The first paper presents a model-driven solution toward the provision of secure messaging capabilities to the financial services industry through the standardization of message flows between industry players. The second paper presents the underlying scientific theories, methodology, and software technology to meet the requirements of high quality technical documentation. The third paper presents a real case of using business semantics management for integrating and publishing research information on an innovation information portal.

We hope that you will enjoy the industrial track proceedings and find useful information and motives to extend your research to new horizons. We would like to express our gratitude to all authors who submitted papers and talk proposals, the members of the program committee for their help and efforts in organizing this track, and the ER 2011 organizing committee and ER steering committee for all their support.

July 2011

Alkis Simitsis
Hans Van Mingroot

QBX: A CASE Tool for Data Mart Design

Antonino Battaglia¹, Matteo Golfarelli², and Stefano Rizzi²

¹ Theorematica S.p.A., Rome, Italy
a.battaglia@theorematica.it

² DEIS - University of Bologna, Italy
{matteo.golfarelli, stefano.rizzi}@unibo.it

Abstract. QBX is a CASE tool for data mart design resulting from a close collaboration between academy and industry. It supports designers during conceptual design, logical design, and deployment of ROLAP data marts in the form of star/snowflake schemata, and it can also be used by business users to interactively explore project-related knowledge at different levels of abstraction. We will demonstrate QBX functionalities focusing on both forward and reverse engineering scenarios.

1 Introduction and Motivation

The continuous market evolution and the increasing competition among companies solicit organizations to improve their ability to foresee customer demand and create new business opportunities. In this direction, data warehouses have become an essential element for strategic analyses. However, data warehouse systems are characterized by a long and expensive development process that hardly meets the ambitious requirements of today's market. This is one of the main causes behind the low penetration of data warehouse systems in small-medium firms, and even behind the failure of whole projects [4].

A data warehouse is incrementally built by designing and implementing one data mart at a time; so, one of the directions to increase the efficiency of the data warehouse development process is to automate design of single data marts. Several techniques for automating some phases of data mart design have been proposed in the literature (e.g., [2] for conceptual design, [5] for logical design, [3] for physical design, [6] for designing the ETL process), and some research prototypes of CASE tools have been developed (e.g., [1]). On the other hand, commercial tools such as Oracle Warehouse Builder are oriented to a single platform and should be considered as design wizards rather than CASE tools.

In this paper we introduce *QBX*, a CASE tool resulting from a close collaboration between academy and industry; in particular, industrial partners took care of the executive design and implementation of the tool. QBX includes two separate components: *QB-Xpose* (read *cube-expose*) and *QB-Xplore* (read *cube-explore*). The first is used by designers for conceptual design, logical design, and deployment of ROLAP data marts in the form of star/snowflake schemata; the design process is further streamlined by letting QBX read from and write to the

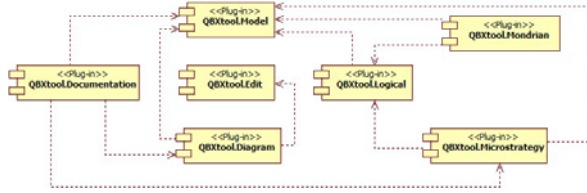


Fig. 1. Simplified component diagram for QB-Xpose

metadata repositories of the Mondrian and Microstrategy multidimensional engines. The second is accessed via browser by business users and technical experts to interactively explore project-related knowledge at different levels of abstraction, ranging from technical documentation to glossaries of terms and concepts based on the business users vocabulary.

2 Architecture

As already mentioned, QBX includes two integrated software tools.

QB-Xpose gives designers an effective support by automating conceptual and logical design of data marts, and by making deployment on ROLAP platforms easier. It was implemented based on *Eclipse* [7], an open source project providing an extensible development platform. The QB-Xpose components were developed in accordance with the Eclipse plug-in contract using three complementary frameworks: Eclipse Modeling Framework (EMF), Graphical Editing Framework (GEF), and Graphical Modeling Framework (GMF). Figure 1 shows the main components of QB-Xpose and their dependencies. Three components implement the model-view-controller design pattern: QBXtool.Model uses the EMF and is responsible for managing the QBX model; QBXtool.Edit and QBXtool.Diagram use the EMF and the GMF, and play the roles of the controller and of the viewer, respectively. QBXtool.Logical supports logical design, while QBXtool.Mondrian and QBXtool.Microstrategy manage the conversion of the QBX meta-model to/from the Mondrian and Microstrategy meta-models.

QB-Xplore enables business users to interactively browse and annotate the project documentation, both at business and technical levels. QB-Xplore is a web application implemented using the Google Web Toolkit. The underlying model was built using the EMF to achieve higher efficiency when exchanging information with QB-Xpose.

3 Functional Overview

From a methodological point of view, QBX supports both classical scenarios of data mart design [8]:

- *Demand-driven approach*, where designers draw their data mart starting from user requirements, possibly by composing existing hierarchies and

reusing conformed dimensions. Establishing a mapping with the source operational schema is postponed to the ETL design phase.

- *Supply-driven approach*, where a data mart schema is semi-automatically derived from the schema of a source operational database. User requirements help designers choose facts, dimensions, and measures. Mappings between the data mart schema and the source schema make ETL design simpler.¹

A basic feature of QBX is that of using conceptual schemata for multidimensional design. Conceptual modeling provides a high level of abstraction in describing the multidimensional repository, aimed at achieving independence of implementation issues. It is widely recognized to be the necessary foundation for building a database that is well-documented and fully satisfies user requirements; usually, it relies on a graphical notation that facilitates writing, understanding, and managing conceptual schemata by both designers and business users. The conceptual model adopted by QBX is the Dimensional Fact Model (DFM) [2]. The DFM was born in the academic context and it has been widely experimented in the industrial world; its main goals are to lend effective support to conceptual design, to make communication possible between designers and end-users with the goal of formalizing requirement specifications, to build a stable platform for logical design, and to provide clear and expressive design documentation. The DFM gives a graphical and intuitive representation of facts, measures, and dimensions; dimensional, descriptive, and cross-dimensional attributes; optional and multiple arcs; convergences; shared, incomplete, and recursive hierarchies; additivity; temporal scenarios.

The adoption of a conceptual model breaks design into two distinct but inter-related phases, that are largely independent of the features of the OLAP engine chosen for deployment:

1. *Conceptual Design*. This phase maps user requirements into a conceptual schema of the data mart, that is, a platform-independent, non-ambiguous, comprehensive representation of the facts for decision support that gives a multidimensional picture of the data mart content to both designers and business users. In a supply-driven approach, conceptual design is automated by choosing relevant facts on a source operational database schema and letting QBX draw hierarchies using the approach in [2]. In a demand-driven approach, a conceptual schema is created from scratch by manually drawing hierarchies and reusing conformed dimensions.
2. *Logical Design*. Starting from a conceptual schema, QBX implements a large set of best practices to create an optimized logical schema for the data mart, that is, a set of star/snowflake schemata. The resulting logical schema can be fine-tuned based on the expected data volume and on the designer's preferences.

¹ This function will be made available starting from release 2.0. In case two or more databases are used to feed the data mart, QBX derives the data mart schema from the schema of the *operational data store* (ODS) that integrates the source databases. Database integration is out of the scope of QBX.

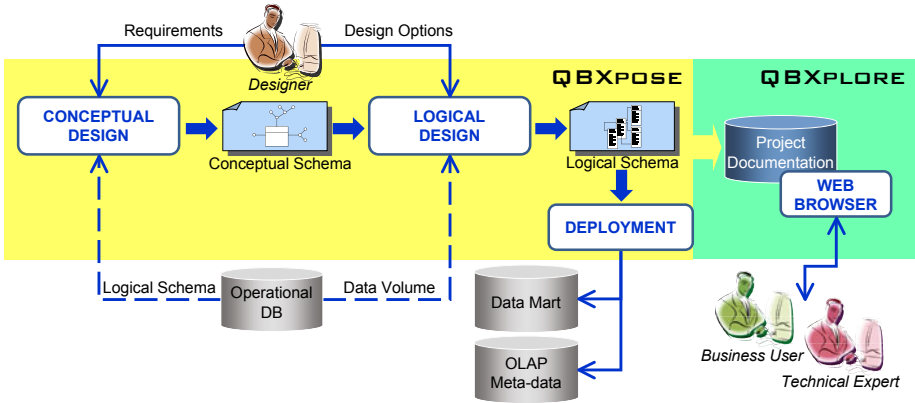


Fig. 2. Forward engineering with QBX

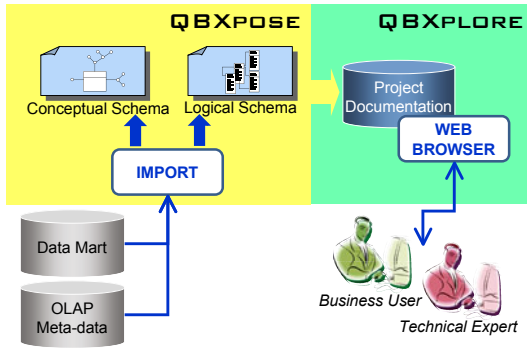


Fig. 3. Reverse engineering with QBX

In this forward engineering scenario (sketched in Figure 2), logical design is followed by a third phase:

3. *Deployment.* Based on both a conceptual and a logical schema, QBX generates SQL code for the underlying relational DBMS and writes the corresponding meta-data on the chosen OLAP engine (either Microstrategy or Mondrian in this release). Since the expressiveness of the DFM (in terms of multidimensional constructs) is wider than the ones of the meta-models of both Microstrategy and Mondrian, the actual structure of the deployed schemata depends on the capabilities of the OLAP engine selected. Similarly, QBX adapts the syntax of the generated SQL statements to the one of the adopted DBMS.

Further QBX functionalities include:

- *Reverse engineering.* To enable designers and business users to enjoy a more expressive view of an existing data mart, QBX can also be employed to

acquire metadata from an OLAP engine and translate them into a DFM conceptual schema. This scenario is sketched in Figure 3.

- *Logical design preferences.* To enable a finer tuning of logical schemata, designers can express a set of preferences about logical design, including e.g. how to deal with degenerate dimensions, shared hierarchies, and cross-dimensional attributes.
- *Data volume.* QBX enables designers to specify the data volume for a data mart, in terms of expected cardinalities for both facts and attributes. This is done manually in a demand-driven scenario, automatically in a supply-driven scenario. When a data volume has been specified, designers can ask QBX to optimize ROLAP schemata from the point of view of their storage space.

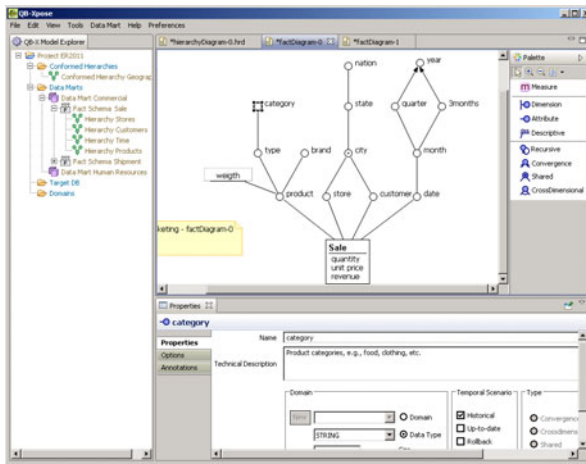


Fig. 4. Conceptual design with QBX

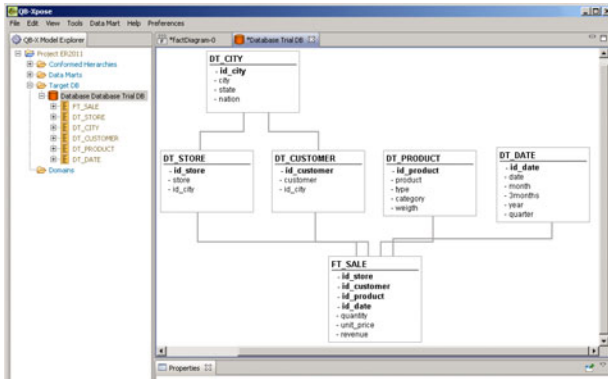


Fig. 5. Logical design with QBX

- *Project-related knowledge.* QB-Xpose automatically creates the documentation for data marts, including fact schemata, conformed dimensions, glossaries, and data volumes. This technical documentation, that can be published on the web and easily browsed via QB-Xplore, is only a part of the project knowledge, that also includes the business descriptions of terms and concepts appearing in reports and analyses. This precious information can be collected from domain experts during requirement analysis using QB-Xpose, or it can be progressively acquired in the form of annotations made by business users using QB-Xplore.

4 Demonstration Scenarios

The demonstration will focus on both forward and reverse engineering scenarios. In the forward engineering scenario, we will adopt a demand-driven approach to interactively draw a conceptual schema (Figure 4), showing how QB-Xpose automatically checks for hierarchy consistency in presence of advanced constructs of the DFM. Then, we will let QB-Xpose generate alternative logical schemata using different design preferences, and critically compare the results (Figure 5). Finally, we will let QB-Xpose create a comprehensive documentation for the project.

In the reverse engineering scenario, the relational schema and metadata of an existing data mart will be imported from the Mondrian engine, and the effectiveness of their translation to the DFM will be discussed.

References

1. Golfarelli, M., Rizzi, S.: WAND: A CASE tool for data warehouse design. In: Proc. ICDE, pp. 7–9 (2001)
2. Golfarelli, M., Rizzi, S.: Data warehouse design: Modern principles and methodologies. McGraw-Hill, New York (2009)
3. Golfarelli, M., Rizzi, S., Saltarelli, E.: Index selection for data warehousing. In: Proc. DMDW, pp. 33–42 (2002)
4. Ramamurthy, K., Sen, A., Sinha, A.P.: An empirical investigation of the key determinants of data warehouse adoption. *Decision Support Systems* 44(4), 817–841 (2008)
5. Theodoratos, D., Sellis, T.: Designing data warehouses. *Data & Knowledge Engineering* 31(3), 279–301 (1999)
6. Vassiliadis, P., Simitis, A., Georgantas, P., Terrovitis, M., Skiadopoulou, S.: A generic and customizable framework for the design of ETL scenarios. *Information Systems* 30(7), 492–525 (2005)
7. Vv. Aa.: Eclipse. <http://www.eclipse.org/platform/> (2011)
8. Winter, R., Strauch, B.: A method for demand-driven information requirements analysis in data warehousing projects. In: Proc. HICSS, pp. 1359–1365 (2003)

The Meta-Morphing Model Used in TARGIT BI Suite

Morten Middelfart¹ and Torben Bach Pedersen²

¹ TARGIT A/S

² Aalborg University – Department of Computer Science
morton@targit.com, tbp@cs.aau.dk

Abstract. This paper presents the meta-morphing model and its practical application in an industry strength business intelligence solution. The meta-morphing model adds associations between measures and dimensions to a traditional multi-dimensional cube model, and thus facilitates a process where users are able to ask questions to a business intelligence (BI) system without the constraints of a traditional system. In addition, the model will learn the user's presentation preferences and thereby reduce the number of interactions needed to present the answer. The nature of meta-morphing means that users can ask questions that are incomplete and thereby experience the system as a more intuitive platform than state-of-art.

1 Introduction

According to leading industry analyst, Gartner, ease of use has surpassed functionality for the first time as the dominant business intelligence platform buying criterion [5]. This change represents a shift from prioritizing the IT department's need to standardize to prioritizing the ability for casual users to conduct analysis and reporting.

Ease of use, in the decision processes that managers and employees go through, has been the focal point in the development of TARGIT BI Suite since its early version in 1995. However, different from other solutions that seek the same objective, TARGIT has methodically applied the CALM philosophy [1], which seeks to create synergy between humans and computers as opposed to using computers simply as a tool to create efficiency. In the CALM philosophy, the entire organization is divided into multiple observe-orient-decide-act (OODA) loops and computing is applied to make users cycle these loops as fast as possible, i.e., with as few interactions as possible. The patented meta-morphing [4], described in the following section, allows users to analyze data by stating their *intent*, and thus facilitates users cycling OODA loops with *few interactions*.

2 The Meta-Morphing Model

The meta-morphing model is an extension of the cube model traditionally used in a data warehouse where associations between measures and dimensions, as well as, presentation preferences are included. As a prerequisite for the meta-morphing model, we assume that the data has been organized in measures (numerical data) and dimensions (entities for which measures will be listed, e.g., time periods, products, etc.), and that these measures and dimensions are organized in one or more cubes. Whether the cube structures are materialized or virtual is irrelevant to the functionality.

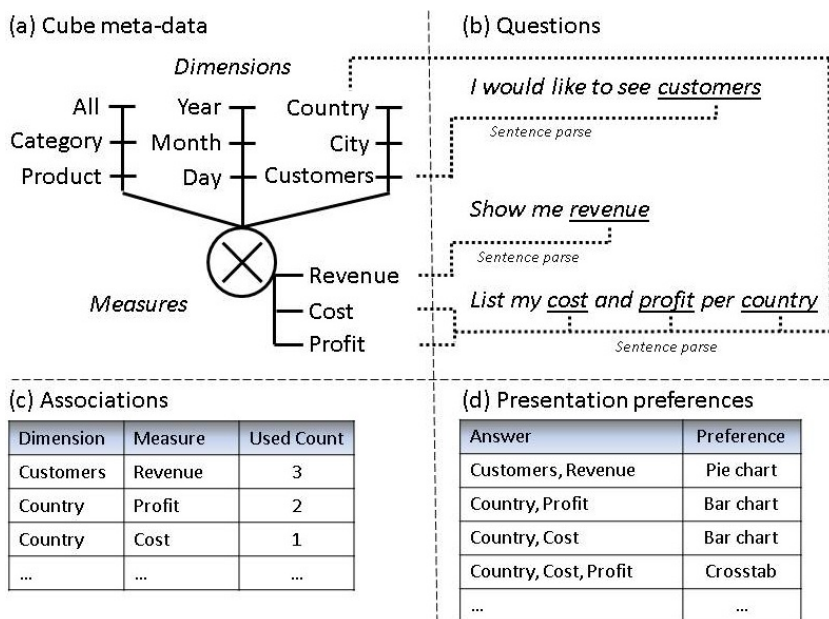


Fig. 1. The meta-morphing model

The meta-morphing model, shown in Figure 1 facilitates the following four steps:

1. A question is parsed into a set of one or more measures or dimensions.
2. If the question is incomplete (meaning that it has either only dimension(s) or measures(s)) then an *association* with the most relevant measure (if question had only a dimension) or dimension (if question had only a measure) is created.
3. A query based on the associated measures and dimensions is executed on the cube.
4. The preferred presentation of the returned data is selected given by either the *user's preferences* (earlier behavior) or if no previous “experience” exists an expert system will determine the presentation based on the size and shape of the returned data. The returned dataset is displayed to the user with the presentation properties identified.

Example: A user submits the question “I would like to see customers”.

Step 1. parses all words in the sentence representing the questions (Figure 1(b)), and these words are subsequently matched against the meta-data of the data warehouse (Figure 1(a)). If a word in the sentence is not matched in the meta-data it is simply thrown away. The output from Step 1 will be a set of dimensions and/or measures; and if the set is empty, the meta-morphing process is simply terminated. In our example, the only word that will remain from this parsing is “customers”.

Step 2. compensates for the problem of the user’s question containing only measures or dimensions. In a traditional system, asking incomplete questions like “I would like to see customers” or “Show me revenue” would at best return a list of customers (the members of the customer dimension) or the sum of revenue for all data in the data warehouse. By creating an association between measures and dimensions (Figure 1(c)), the system will learn the *individual* user’s behavior based on what he clicks, e.g., if he

clicks to select revenue and customers at the same time, then an association between revenue and customers will be created. Therefore, the answer to both questions will be a list of customers with their respective revenue. Associations are also created while the user loads any analysis, meaning that he does not need to actively pose a question including the association. This means that the user will simply feel that he is receiving information in the way he is used to. In the event that a user has never seen or asked for a given relationship, the meta-morphing process will look into which dimension or measure the user most often uses, and then create an association that is used most often, i.e., the measure or dimension from the association with the highest Used Count (see Figure 1(c)). The output of Step 2 is a combination of measures and dimensions.

Step 3. is, given the output of Step 2, a trivial query in the data warehouse retrieving “revenue” for each member on the “customers” dimension. The output of this step is a dataset as well as the dimensions and measures used to provide it.

Step 4. preferences are created for each user given the way they would normally see an answer given its dimension/measure combination, e.g., customer/revenue is usually displayed as a pie-chart (see Figure 1(d)), profit per country is usually displayed as a bar-chart, etc. Given the “experience” with the user’s preferences that are collected whenever the user sees or formats a piece of information, the dataset received from Step 3 is formatted and displayed to the user. In the event that no preferences have been collected for the user, an expert system will inspect the dataset and make a call for the best presentation object (pie, bar, geographic map, table, etc.), this expert system is based on input from a TARGIT BI domain expert. In our example, the returned revenue for each customer will be presented as a pie chart based on the data in Figure 1(d).

Using the meta-morphing model, users are able to pose incomplete questions that will still return relevant answers, while at the same time save the users a number of interactions in formatting the dataset returned since these are already known to the system. In other words, the user will be guiding the system with his intent, and the computer will provide him with the best fitting output based on the his individual preferences.

Another interesting aspect of meta-morphing is that it will allow the user to ask questions in human language as opposed to a database query language, which will allow a much more natural intuitive feel to the application that exploits the process. In particular, with regards to speech recognition, the parsing of question to meta-data in Step 1 will mean that the recognition will be enhanced simply from the fact that fewer words will be in the vocabulary, as opposed to a complete language. The combination of meta-morphing and speech is also a patented process [3].

3 Meta-Morphing in the TARGIT BI Suite

The TARGIT BI Suite is recognized by analysts as being one of the leading global business intelligence platforms with more than 286,000 users World-wide. Although no specific usability study has been conducted, the TARGIT BI Suite has been surveyed by leading industry analysts to have a unique strength in its ease of use achieved by reducing the number of interactions that a user needs to conduct in order to make decisions [5]. The so-called “few clicks” approach has been demonstrated to allow users to easily interpret the datasets displayed using automatically generated explanations [2].

In the TARGIT BI Suite, the meta-morphing process is integrated such that users have the option of activating it dynamically in three different scenarios: guided analytics called *Intelligent Analysis*, a quick drag-drop function called *TARGIT This*, and finally, an analytical link to all dashboards and reports known as *Hyper-Related OLAP*. **Intelligent Analysis** allows users to compose sentences similar to our example in the previous section by clicking on meta-data in a semi-structured environment. Once a question is posed, the process using the meta-morphing model can be activated.

TARGIT This is a drop-area to which either a dimension or a measure can be dropped, and upon “release” of the item dropped, the process using the meta-morphing model will commence with the question “I would like to analyze [measure or dimension]”.

Hyper-Related OLAP is perhaps where the most powerful results are achieved by the meta-morphing process. Hyper-Related OLAP allows the user to click any figure in the TARGIT BI Suite in order to analyze it. Since any figure presented on a business intelligence platform is a measure “surrounded” by dimensions (either as grouping or criteria), the process using the meta-morphing model can be activated by a single click at any figure, with the question “I would like to analyze [measure]”. This gives the user a starting point for analyzing any figure whenever he sees something he wants to investigate further. This functionality significantly reduces the time and interactions needed from whenever a problem is observed to when an analysis can be conducted in order to reach a decision with subsequent action. In other words, Hyper-Related OLAP directly assists the users in cycling their individual OODA loops with fewer interactions.

4 Conclusion

This paper presented the meta-morphing model and showed its practical application in an industry strength business intelligence solution. It was specifically demonstrated how the meta-morphing model will allow users to freely pose questions in human language, including in speech, and subsequently receive a presentation of the answer in accordance with their preferences. It was demonstrated how the meta-morphing model can contribute with greater ease of use by reducing the number of interactions needed in data analysis. Moreover, it was demonstrated how meta-morphing can reduce the time and interactions for users cycling an observation-orientation-decision-action loop.

Acknowledgments. This work was supported by TARGIT A/S, Daisy Innovation and the European Regional Development Fund.

References

1. Middelfart, M.: CALM: Computer Aided Leadership & Management. iUniverse (2005)
2. Middelfart, M., Pedersen, T.B.: Using Sentinel Technology in the TARGIT BI Suite. PVLDB 3(2), 1629–1632 (2010)
3. Middelfart, M.: Presentation of data using meta-morphing. United States Patent 7,779,018 (Issued August 17, 2010)
4. Middelfart, M.: Method and user interface for making a presentation of data using meta-morphing. United States Patent 7,783,628 (Issued August 24, 2010)
5. Sallam, R.L., Richardson, J., Hagerty, J., Hostmann, B.: Magic Quadrant for Business Intelligence Platforms, www.gartner.com/technology/media-products/reprints/oracle/article180/article180.html (April 28, 2011)

The Demonstration

Scenario 1: We use structured meta-morphing in the TARGIT BI Suite to compose the question “I would like to analyze Revenue” and subsequently click the “Go” button (Figure 2).

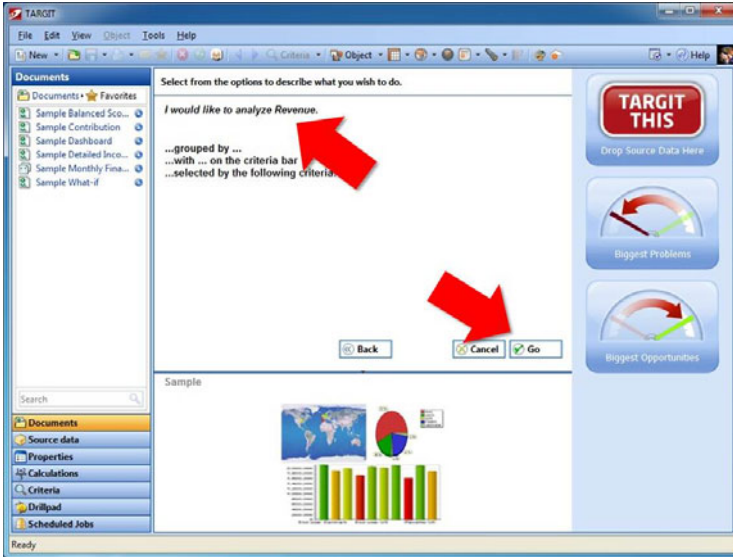


Fig. 2. Composing the question via structured meta-morphing



Fig. 3. Analysis screen resulting from structured meta-morphing

Based on experience with the user’s preferences for presenting the measure “Revenue”, a complete analysis (with the measure “Revenue” displayed over a set of dimensions: “Customer Country”, “Item”, and “Period”, including their presentation preferences: map, pie-, and bar chart) is presented (Figure 3).

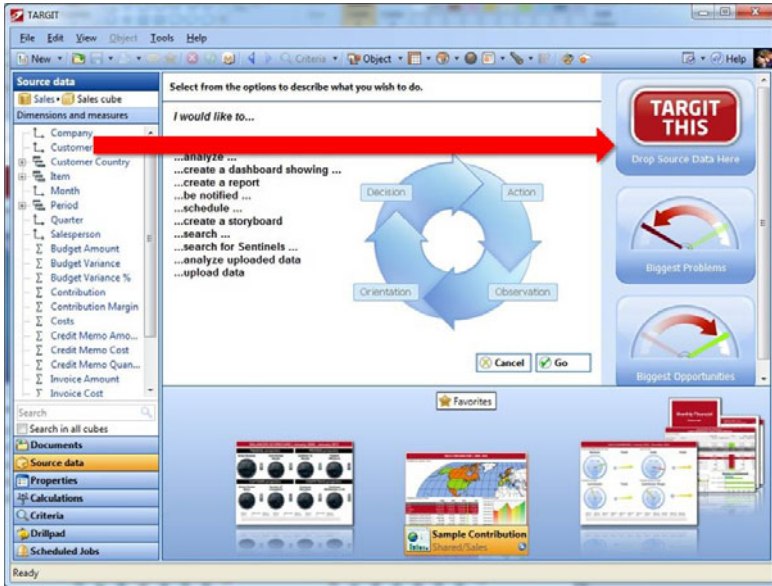


Fig. 4. Composing the question using the “TARGIT This” feature

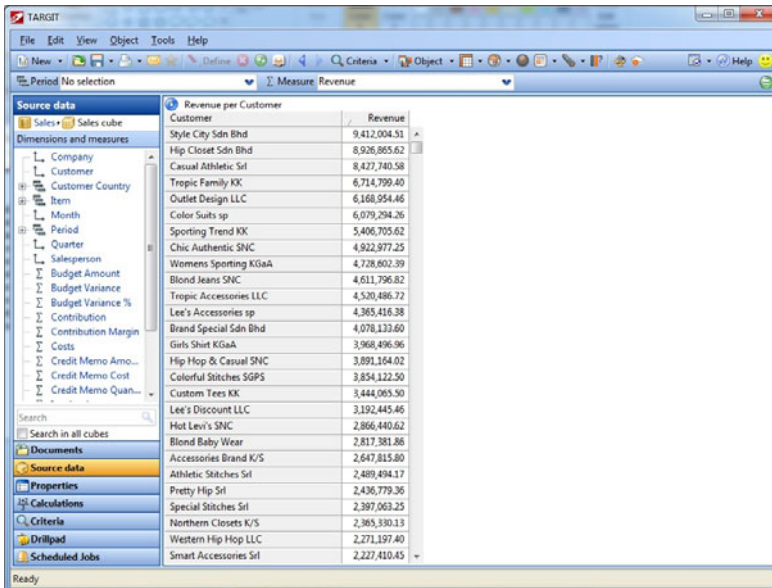


Fig. 5. Analysis screen resulting from “TARGIT This”

Scenario 2: We drag the dimension “Customer” and drop it on the “TARGIT This” drop area in the TARGIT BI Suite (Figure 4).

Based on experience with the user’s preferences the customer dimension is presented with the measure “Revenue”, and given the users presentation preferences the output is presented in a table (Figure 5). The system automatically adds a descending sorting to the table and an option for selecting on the time dimension (criteria option above the table).

Tool Support for Technology Scouting Using Online Sources

Elena Tsiporkova and Tom Tourwé

Sirris - ICT & Software Engineering Group
A. Reyerslaan 80, 1030 Brussels, Belgium
{elena.tsiporkova,tom.tourwe}@sirris.be

Abstract. This paper describes a prototype of a software tool implementing an entity resolution method for topic-centered expert identification based on bottom-up mining of online sources. The tool extracts and unifies information extracted from a variety of online sources and subsequently builds a repository of user profiles to be used for technology scouting purposes.

1 Introduction

Many firms are nowadays looking for opportunities to adopt and implement a formal, structured and focused approach for the identification and acquisition of new technology, and to develop technology based product and service innovations. This is usually referred to as technology scouting and is understood as an organised approach for identifying technological needs, gaps and opportunities, and then finding solutions outside the official borders of the enterprise. It is very often applied when: 1) a technical problem needs to be solved quickly due to some change in the competitive landscape; 2) an organisation is looking for opportunities to move into a new market with limited involvement of internal resources; 3) or specific new skills need to be acquired without increasing internal resource overhead.

The role of the technology scout will therefore include searching for opportunities and leads within a certain technological domain, evaluating leads, and creating the link between a lead and company strategy. A technology scout needs to utilize an extensive and varied network of contacts and resources, and stay on top of emerging trends and technologies.

With the rise of the social web and the advent of linked data initiatives a growing amount of data is becoming publicly available: people communicate on social networks, blogs and discussion forums, research events publish their online programmes including article abstracts and authors, websites of technological conferences and exhibitions advertise new products and technologies, governments and commercial organisations publish data, and the linked open data cloud keeps on growing. An enormous potential exists for exploiting this data by combining it and extracting intelligence from it for the purpose of technology scouting. However, there are currently no adequate tools available to support the

work of the technology scouts and although a large pool of data is available on the web, it is often gathered manually, a time intensive, tedious and error-prone process, due to the fact that the data is not centralised, is available in different formats, can be outdated or contradictory, etc.

In this paper, we describe a prototype of a software tool for topic-centric expert identification from online sources. We present the rationale and the concrete approach applied in the realisation of the tool. The main novelty of our approach lies in the fact that it is based on the *bottom-up*, *application-driven* and *incremental* creation of the repository, as opposed to the more commonly considered *top-down* approach.

2 Rationale

Identifying experts is by far not a new topic and it has been gradually gaining interest in the recent years [1,2,3,4]. However, there are certain shortcomings associated with the existing approaches *e.g.* lack of focus on realistic applications, limited to a single source, targeting too large scale, poor resolution and accuracy, high information redundancy, etc.

The problem of finding experts is very often approached top-down with respect to both application and scale, meaning that often a certain method or technology is developed without a clear idea of the potential application in mind and by default coverage comes before accuracy. Many works on the topic focus mainly on the elaboration and application of advanced information retrieval, machine learning and reasoning techniques (see [5,6,7,8,9,10]), while considering rather synthetic applications as research collaboration or enterprise expert search ([11,12,13]). Some other recent contributions [14,15] in the area of integrating web data consider a set of challenging research topics as data cleansing, profiling, and entity resolution, but suffer from the same lack of focus on concrete and realistic applications.

In terms of scale, the problem of finding experts is usually tackled by mining vast online data sources as *e.g.* Wikipedia and CiteSeer (see Jung et al. [2]), in order to gather a sufficiently large data repository containing information about persons, articles, social links, etc. This has the advantage of achieving high coverage of the experts active in a certain domain and relatively complete expert profiles in space and time. Unfortunately, such large data collections contain a substantial proportion of noisy data (contradictions, duplicates, ...) and the achieved degree of accuracy cannot be estimated in a reliable way. Accuracy is most commonly measured by precision and recall. Precision is the ratio of true positives, *i.e.* true experts in the total number of found expert candidates, while recall is the fraction of true experts found among the total number of true experts in a given domain. However, determining the total number of true experts in a given domain is not feasible. Furthermore, for applications as technology scouting the current activity and location of certain experts is more important than their career evolution within an extensive time span. In general, for applications delivering data for business decision making purposes the reliability of the information at the current time period is crucial.

We propose below an expert finding approach for the purpose of technology scouting applications. The approach implements an entity resolution method which allows reliable disambiguation of authors of scientific articles. Initially, data sources closely aligned with the topic of interest are considered in order to create the initial expert repository. This is further extended by integrating several different types of linked data sources e.g. Google Scholar and DBLP. Subsequently, to facilitate the entity resolution algorithm, LinkedIn (or other appropriate source) is used as an auxiliary data source to determine if two authors with similar names, or with the same name but different affiliations, are the same person or not.

3 Approach

The types of expert-related information identified as relevant to technology scouting applications are as follows: 1) actors in the field: leading researchers and experts, as well as companies, research institutes, universities; 2) technology-related publications: scientific and popularised publications, presentations and keynotes, press releases and technology blogs; 3) research activities: past and ongoing research projects and collaborations, organisation and participation in research events, etc. Relationships between the different data entries need also to be identified and made explicit, e.g. formal and informal expert networks, identified through joint publications, joint organisation of events, social networks such as LinkedIn and Twitter, professional and educational history.

Following the reasoning in the foregoing section, we approach the problem of identifying the main actors in a certain technological domain bottom-up by first identifying online sources to mine, targeted to the application domain in question. These serve as seeds for the further incremental growth of our expert repository. The main rationale behind the seed approach is that different expert communities use different communication channels as their primary mean for communicating and disseminating knowledge, and thus different types of sources would be relevant for finding experts on different topics.

Thus if we want to identify currently active researchers in a particular domain, as for example in software engineering, we can assume such experts regularly publish in top-quality international conferences, such as for example The International Conference on Software Engineering (<http://icse-conferences.org/>). Such conferences nowadays have a dedicated website which details their program, i.e. the set of accepted papers that will be presented, together with their authors and titles, often grouped in sessions. We consider such a website as the initial seed for our approach: we extract information about the topics presented, e.g. the titles and abstracts (if available) of presented papers and the session in which they are presented, and about who is presenting, e.g. author names, affiliation, e-mail addresses, etc.

A drawback of this approach of having a front end which is very tightly aligned to the topic of interest is that a dedicated tool needs to be developed each time a new seed source is considered for mining. This imposes certain limitations on the level of automation that can be achieved.

The initial set of experts is rather limited: one author typically only publishes one paper at a particular conference, probably collaborates with a broader set of people besides his co-authors for this particular paper, and is potentially interested in more topics and domains than the ones addressed in this paper. To extend the gathered information, we consider additional sources using the extracted information as a seed. We search for every author and co-author on Google Scholar (<http://scholar.google.com>) and the DBLP website (<http://www.informatik.uni-trier.de/~ley/db/>) to identify additional published material, resulting in a broader set of co-authors and a broader set of topics of interest. Although the completeness of the information improves, the level of accuracy decreases as more noise occurs: different authors might share a name, one author's name might be spelled differently (e.g. "Tourwé, T." versus "Tom Tourwé") in different papers, different affiliations might occur for a seemingly unique author, etc.

In order to clean the data, we again consider additional sources, such as LinkedIn or ScientificCommons (<http://en.scientificcommons.org/>) for example. People use LinkedIn to list past and present work experience, their educational background, a summary of their interests and activities, etc. In addition, information about people's professional network is available. We exploit all this information to merge and disambiguate the extracted data, e.g. we use work experience data to merge authors with different affiliations, and we separate authors with the same name by matching the list of LinkedIn connections with the list of co-authors.

The raw data contains a simple enumeration of keywords reflecting to some extent the topics addressed by a single author. These have been obtained after processing the extracted textual information for each expert (*e.g.* article title and abstract, session title, affiliation, etc.) in two steps:

- *Part-of-speech tagging*: The different words in the total text collection for each author are annotated with their specific part of speech using the Stanford part-of-speech tagger ([16]). Next to the part of speech recognition, the tagger also defines whether a noun is plural, whether a verb is conjugated, etc.
- *Keyword retention*: The annotated with part of speech text is subsequently reduced to a set of keywords by removing all the words tagged as articles, prepositions, verbs, and adverbs. Practically, only the nouns and the adjectives are retained and the final keyword set is formed according to the following simple algorithm:
 1. *adjective-noun(s) keywords*: a sequence of an adjective followed by a noun or a sequence of adjacent nouns is considered as one compound keyword *e.g.* 'supervised learning';
 2. *multiple nouns keywords*: a sequence of adjacent nouns is considered as one compound keyword *e.g.* 'mixture model';
 3. *single noun keywords*: each of the remaining nouns forms a keyword on its own.

The above process is rather crude and generates an extensive set of keywords, without any relationship between them and without guarantee that all of the

retained keywords are relevant to the topic of interest. To attain more accurate and more concise annotation of expert profiles with interrelated domain-specific keywords, a conceptual model of the domain of interest, such as a taxonomy or an ontology, can be used. In our software engineering example, this would allow to derive that the keywords “user stories” and “prioritisation” are both related to the “requirements” concept. This in turn would allow to cluster different authors on different levels of abstraction, *e.g.* around higher-level research domains, such as *requirements engineering* or around specific research topics such as *agile requirements elicitation*. Note that after such formal classification, the data can potentially be cleaned even more, as one author with two seemingly unrelated sets of topics of interest potentially should be split into two authors, or two separate authors with perfectly aligned topics that were not unified in a previous cleaning step could be merged now this additional information has become available.

Ontologies for many different domains are being developed, *e.g.* an exhaustive ontology for the software engineering domain has been developed in [17]. An alternative approach is to derive a taxonomy using additional sources, such as the Wikipedia category structure or the Freebase entity graph.

An additional advantage of the construction of a formal semantic model of the domain of interest is that this enables the use of automatic text annotation tools for analysing extensive expert-related textual information in order to enrich the expert profiles with domain-specific keywords that are not explicitly present in the original documents.

4 Demo

The tool platform proposed above is currently in an initial development stage. A concrete use case scenario will be prepared for the demonstration session in order to illustrate different functionalities *e.g.* keyword generation for user profiling, particular disambiguation features, domain modelling through taxonomy building, etc.

5 Conclusion

A software tool which serves as a proof-of-concept for topic-centered expert identification based on bottom-up mining of online sources is presented. The realised approach is still work in progress and obviously, the gathered data only covers parts of the data needed for technology scouting purposes. We consider to further extend and complemented it with information obtained from research project databases (*e.g.* the FP7 database), online patent repositories (*e.g.* US Patent Office, Google Patents) or technological roadmaps.

References

1. Balog, K., de Rijke, M.: Finding similar experts. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 821–822. ACM, New York (2007)

2. Jung, H., Lee, M., Kang, I., Lee, S., Sung, W.: Finding topic-centric identified experts based on full text analysis. In: 2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007 (2007)
3. Zhang, J., Tang, J., Li, J.: Expert finding in a social network. In: *Advances in Databases: Concepts, Systems and Applications*, pp. 1066–1069 (2010)
4. Stankovic, M., Jovanovic, J., Laublet, P.: Linked Data Metrics for Flexible Expert Search on the Open Web. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011*. LNCS, vol. 6644, pp. 108–123. Springer, Heidelberg (2011)
5. Boley, H., Paschke, A.: Expert querying and redirection with rule responder. In: 2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007 (2007)
6. Fang, H., Zhai, C.X.: Probabilistic models for expert finding. *Advances in Information Retrieval*, 418–430 (2007)
7. Zhang, J., Tang, J., Liu, L., Li, J.: A mixture model for expert finding. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) *PAKDD 2008*. LNCS (LNAI), vol. 5012, pp. 466–478. Springer, Heidelberg (2008)
8. Balog, K., Azzopardi, L., de Rijke, M.: A language modeling framework for expert finding. *Information Processing & Management* 45, 1–19 (2009)
9. Hofmann, K., Balog, K., Bogers, T., de Rijke, M.: Contextual factors for finding similar experts. *Journal of the American Society for Information Science and Technology* 61, 994–1014 (2010)
10. Tung, Y., Tseng, S., Weng, J., Lee, T., Liao, A., Tsai, W.: A rule-based CBR approach for expert finding and problem diagnosis. *Expert Systems with Applications* 37, 2427–2438 (2010)
11. Sriharee, N., Punnarut, R.: Constructing Semantic Campus for Academic Collaboration. In: 2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007, pp. 23–32 (2007)
12. Pavlov, M., Ichise, R.: Finding experts by link prediction in co-authorship networks. In: 2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007, pp. 42–55 (2007)
13. Jung, H., Lee, M., Sung, W., Park, D.: Semantic Web-Based Services for Supporting Voluntary Collaboration among Researchers Using an Information Dissemination Platform. *Data Science Journal* 6, 241–249 (2007)
14. Böhm, C., Naumann, F., et al.: Profiling linked open data with ProLOD. In: *Proceedings of the 26th IEEE International Conference on Data Engineering ICDE 2011, Workshops*, pp. 175–178 (2010)
15. Pu, K., Hassanzadeh, O., Drake, R., Miller, R.: Online annotation of text streams with structured entities. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010*, pp. 29–38 (2010)
16. Toutanova, K., Kelen, D., Manning, C.: Enriching the knowledge sources used in a maximum entropy part of speech tagger. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora EMNLP/VLC 2000*, 63–70 (2000)
17. Wongthongtham, P., Chang, E., Dillon, T., Sommerville, I.: Development of a Software Engineering Ontology for Multi-site Software Development. *IEEE Transactions on Knowledge and Data Engineering* 21, 1205–1217 (2009)

Governance Issues on Heavy Models in an Industrial Context

Sabri Skhiri¹, Marc Delbaere², Yves Bontemps², Gregoire de Hemptinne¹,
and Nam-Luc Tran¹

¹ Euranova S.A.

² SWIFT : Society for Worldwide Interbank Financial Telecommunication

Abstract. SWIFT is a member-owned cooperative providing secure messaging capabilities to the financial services industry. One critical mission of SWIFT is the standardization of the message flows between the industry players. The model-driven approach naturally came as a solution to the management of these message definitions. However, one of the most important challenges that SWIFT has been facing is the global governance of the message repository and the management of each element. Nowadays modeling tools exist but none of them enables the management of the complete life-cycle of the message models. In this paper we present the challenges that SWIFT had to face in the development of a dedicated platform.

Keywords: governance, meta-modeling, operational issues.

1 Introduction

SWIFT is the leader in the banking communication and message transmission. One of its main missions is the management of the communication standards ISO-15022 and ISO-20022 that are used between banks in order to exchange messages. Those standards provide the definition of message payloads (i.e. which data fields can or must be included in which communication flow).

One of the difficulties of managing a worldwide business standard is the continuous need to evolve the standard to cater for new business requirements. From a model management point of view, this creates a lot of new definitions that then have to be organized properly.

In 2009, SWIFT Standards undertook a major strategic study aimed at defining a 5 year roadmap for standard capabilities evolutions. They identified a set of priorities: (i) the management of the content and the reuse, (ii) the ability to support specialized standards and market practices, and (iii) the management of changes.

Very recently, SWIFT and their customers have reiterated the role of ISO-20022 as a mechanism to facilitate the industry integration at the business level. In order to realize this vision, the need of a common and machine-readable definition has been established. This definition comprised the business processes, the data dictionary, the message definitions, the market practices and the mapping

rules. All of these definitions would be managed in a controlled manner over time (versioning). It should be possible for industry players to customize the definitions in order to fit their needs (e.g. local market practices, bilateral agreements) as well as the representations of their own formats in this environment (using the expressive power of the ISO 20022 global dictionary).

A thorough model-based approach has been set up in order to enable the proper governance of the whole model and the interoperability across the actors working on the model.

2 Technical and Operational Issues

All the issues related to the management of an industry-wide repository of structured business content are related to the concept of governance. This concept involves the control of the life-cycle of each model element, the management of the dependencies between them, the versioning, the design of new elements based on existing components and many other aspects related to governance. One additional complexity in the case of the ISO 20022 models is the existence of two layers of models: a business layer that describes the business at conceptual level (process definitions, business concept definitions at the semantic level) and a message definition layer that describes all of the data objects and the actual message definitions. In this architecture, each data element present in a message definition must be linked back to a semantic element. This additional layer makes it even harder to govern the repository content. In this section we briefly describe the different challenges met by such governance framework.

- **Versioning:** we should be able to maintain a version for each model element with a granularity at the atomic attribute. Most of the modeling frameworks (such as EMF) usually use XMI (XML Metadata Interchange) for the serialization of the models. We would then have to rely on the versioning at the file level, which does not meet the requirement of our versioning granularity.
- **Link between message and semantic objects:** when the first messages are created, the links between the business components and the content of the messages are clear. This relationship must remain clear over the evolution of the repository and the messages. How can we guarantee the relational integrity between business objects and message definitions over different versions?
- **Knowledge of the market practices:** each financial environment has its own needs. In order to let the model evolve on the right way we need to know the main market practices and how they are used. From a model viewpoint, the market practices are a set of restrictions applied on a view. A view is the projection of a subset of messages from the repository. We should be able to maintain a coherent market practice along the different versions of the messages that it constraints. Going further, we should be able to evaluate the impact of a message change on all the market practices using this message and also on the complete repository. This is a typical governance use case.

- **Content management:** the documentation and the structure of the models needs to be centralized in one place in order to promote the reusability of message components and to control the repository content. This includes, among others, the documentation, the search tools, the central repository, the message model templates and the comparison tools (e.g.: find isomorphic components).
- **Auditing tools:** we should be able to audit each modification in the repository and to track back detailed information on the change such as the identity of the requester, the version, the implementor, the impacted release, etc...

3 Solutions

The foundational element of the solution is the implementation of a meta-model above the current standard model. This meta-model allows to manage and to simplify the use of the standard model: instances of the meta-model are the models of the standard, and instances of the models are the message definitions. Both are defined in an XMI file.

- **Versioning and organization:** based on the standards defined before, we have developed a graphical interface built on a meta-model which allows us to create and manage models based on the meta-model and with respect to the standards.
- **Business objects:** these objects are based on the semantic model. The link between the semantic model and the message model is always maintained thanks to the object-oriented programming methodology. Since the message model and the semantic model are defined in the same meta-model, they can be linked and work together. Once the link between message and semantic is done, it is conserved.
- **Auditing tools:** thanks to the meta-model, the objects are always linked. All the variations of the model are based on differences between messages. Every definition has links to other instances of definitions. This makes possible the comparison of objects and the creation of auditing tool. It allows to add search functionalities, impact analysis and message definitions comparison.
- **Market practices:** meta-modeling is also a good choice for the creation of market practices. These are the usages in which the message definitions are used in practice by the users. Thus, these consist in restrictions added on a message definition. This can be seen as elements that are added to the base object. This is easy to accomplish thanks to the meta-modeling. The restriction should be easy to change and to remove. We therefore keep the initial object every time, and it becomes a restricted object.
- **Content management:** in order to structure the documentation, the model allows adding documentation on each object. Furthermore, using models to organize objects and data makes things clearer for non-expert people.

The data can then be stored on a centralized way in order to promote re-usability. Modeled architecture data storage as provided by CDO is ideal in this situation.

4 Implementation

We have developed a framework answering the governance issues, based on the Eclipse Rich Client Platform (RCP) tools with the Eclipse Modeling Framework (EMF). We have created a visual editor in order to manage the models very efficiently. Mainly based on tree and editor views, this interface is intended to draw a graphical representation of the models. The documentation for each model element can be added almost everywhere and can be reviewed. We developed an authorization and authentication layer integrated with the editor. In such way, we are able to track back any operation on models. This makes it possible to formally separate the global standardization process from the definition of local market practices.

5 Illustration

In this section we chose to describe two important mechanisms of the Model editor. This description aims at giving a better understanding of the meta model and its usage.

5.1 View Model

The View model is used for grouping elements within an high level view. Originally it was designed for resolving two important challenges of the ISO 20022 Repository: (1) finding a relevant content, as its relevance may rely on several criteria of different kinds (project, business domain, publication, etc.), (2) the need to define finer-grained scopes, since the dictionary size will dramatically increase in the coming years. Therefore, the view model is a mechanism for providing a view on specific set of message components. In additionn it offers informations to describe when this view was extracted. The root element of the model exposed in the Figure 1 is the ViewSet, which is nothing else than a set of views. Basically a View has a name, a description and a list of message components. A view does not contain components, it references them. A view can be checked out from the repository. In such a case, the editor will create a local copy of each element. A change request can then be associated with this view. That is why the change model is also linked with the view model. In case of modifications when the view is checked in, the user can use EMF compare and its 3-way comparison to evaluate the changes and merge the different versions. The mechanism of Publication View is based on the same concept. A publication is the mechanism by which the base standard is published, it uses the same concepts and adds additional elements: the previous and next Publication View. These attributes enables to link the publications between them.

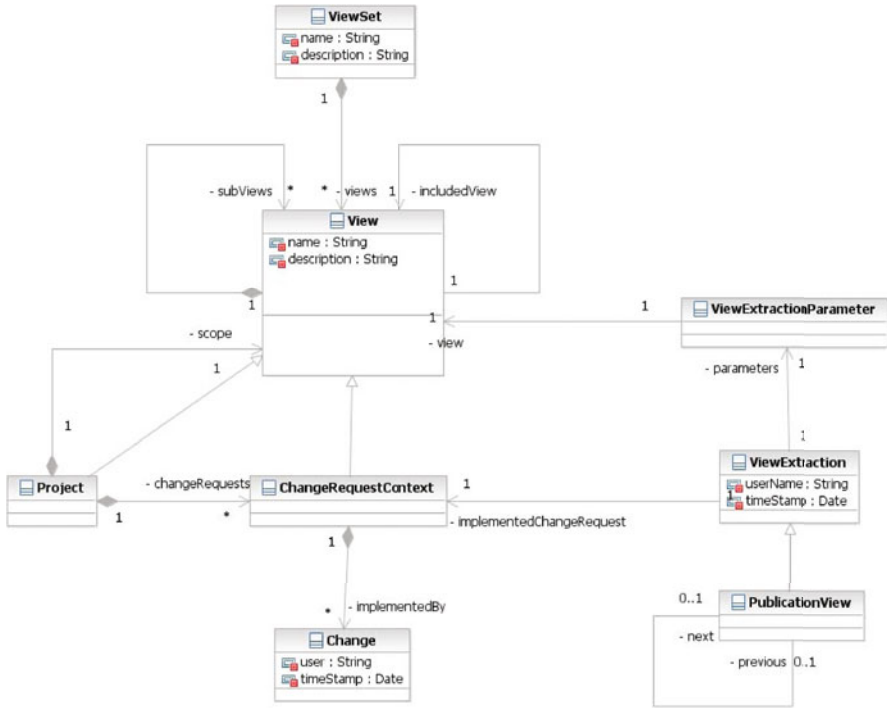


Fig. 1. The view model is used for checking out content, but also for publishing new base standard

5.2 Semantic Traces

Semantic traces aim at simplifying the understanding of the message model. The ISO20022 repository is composed of a conceptual and a logical layer. The first is used for defining the normalized business model while the second layer is used for defining message components. The message components are a view of the business components, they make up the real message definitions used on the network. The mapping to the business layer aims at giving semantic meaning. In the example of the Figure 2, the PersonIdentification is the message component while its attributes are the message elements. We have to trace the message component to a business component, in this case the Person, but also each message element. A message element can be traced to a business element of the business component traced from its container (person in this case) but also to any other business elements of any business component connected to Person through a navigation path. For instance, a navigation path can specify that the birthPlace message element represents the IsoCode of the birthPlace of a Person. By linking two parts of the model that evolve at their own pace, semantic traces can pose governance issues, namely consistency problems.

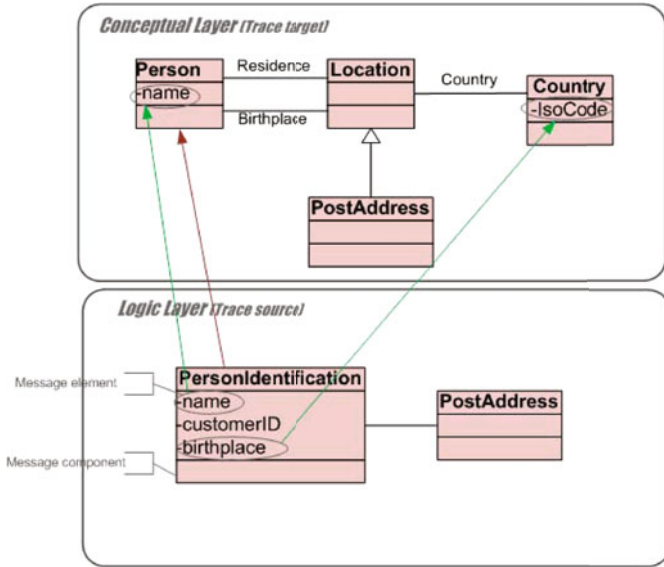


Fig. 2. The semantic trace between the business and logical layer. This trace is used for defining the impact of an element.

Thanks to the meta-modeling approach, we could define the concept of impact, eg, a trace target X (a business component has an impact on a trace source y (a message component) only if the target of y is X, and implement an impact analyzer. Now, before any change is performed on the business model, analysts can assess the semantic impact on message definitions.

6 Conclusion

The governance of models of significant size is a real challenge and no framework today can cover all its different aspects. In this case, we have developed a simplified framework covering our prior requirements. However, there is a need for research and industrialization in this area. It has clearly been shown that the modeling approach brings more cohesion and clarity in the managed data. With this problem impacting significant models, we see how we can go back to a more elegant solution that allows easier governance by adding abstraction and meta-modeling.

High Quality Technical Documentation for Large Industrial Plants Using an Enterprise Engineering and Conceptual Modeling Based Software Solution

Steven J.H. van Kervel

Formetis BV, Hemelrijk 12 C, 5281PS Boxtel, The Netherlands
steven.van.kervel@formetis.nl

Abstract. Engineering contractors are building very large installations such as nuclear power plants, airports, oil refineries etc. Two closely related artifacts, the plant and the accompanying technical documentation, need to be produced. The quality requirements, the size and the complexity of the documentation are high and any shortcoming may lead to severe problems. State of the art IT systems for engineering documentation fail to meet the requirements and address only the symptoms of problems. This paper presents the underlying scientific theories, methodology and software technology to meet the requirements of high quality technical documentation. The conceptual modeling methodology delivers high quality enterprise models. A software engine executes directly these enterprise models and is the de facto IT system. This paper is for the ER 2011 Industrial Track, describing an innovative software artifact as a solution for a specific industrial problem.

Keywords: technical documentation, engineering contractors, enterprise engineering, ontology, conceptual BPM modeling, DEMO methodology, DEMO processor, IT system engineering, document information systems.

1 Introduction

Engineering contractors are designing and building very large and complex installations such as nuclear power plants, airports, oil refineries etc. The whole plant is described, specified in detail by technical documentation. This documentation is of equal importance as the actual plant and should meet also high quality requirements. This application domain is characterized by increasing size and complexity of the plants, compliance to increasing legal and other external regulation and the increasing need to apply and control rules for governance. This increases the overall complexity of the documentation and the number of rules –business processes- to be followed in an exponential way. The risks, responsibilities, liabilities and the financial exposure increase similarly. The elements of this domain are:

- Plants to be designed, constructed, maintained and demolished after service life.
- Technical documentation to provide an appropriate and truthful representation of the plant during service life.

- The production of the documentation is performed by an 'enterprise', an engineering contractor, working according to business procedures.
- The enterprise uses an IT system for the production of technical documentation.

2 The Problems and Symptoms

Technical documentation that is not appropriate, truthful, and/or also not concise, comprehensive, consistent and coherent (the C4-ness quality criteria) causes practical problems:

- Documentation as the most important control mechanism to build the plant (a BoM is just a part of this documentation) may cause production delays.
- A technically perfect plant without proven high quality documentation may not go into production for legal reasons.
- Ambiguity and uncertainty in assignment or exclusion of responsibilities and liabilities of stakeholders, suppliers etc, may lead to complex legal procedures.
- Several symptoms of problem are mentioned in the NEMI White Paper Report, [1]: "i) Document File type, conversion and format issues, lack of standard formats. Paper documents; ii) Document version versus component version issues; iii) BoM correctness issues (up to 80% error rates), often delivered by suppliers; iv) Consistency, Completeness and Correctness problems. v) High level of 'fire-fighting' due to waiting queues;" etc.

It is claimed that these are mostly symptoms of unrecognized underlying problems.

3 Requirements for Technical Documentation

Despite the growing complexity, documentation should meet stringent requirements:

- Documentation should provide a high quality appropriate and truthful representation of the plant during the whole life cycle of the plant and after this.
- Documentation should support several active functions; validation of functional requirements and constructional requirements by the customer; support the construction of the plant; support the production of the plant and support validation of the plant against the documentation. The IT system should be able to render automatically views such as a BoM, current state of the construction, identification of bottlenecks and support for external project management tools.
- Documentation should be able to deliver automatically rendered proof of overall verification; all business procedures must have been executed and terminated.
- The overall state of the documentation can be reconstructed for any point in time.

4 State of the Art of IT Systems and Business Alignment

There is broad recognition in the literature (Standish Group 2009; Saur and Cuthbertson 2003; Tata Consultancy 2007) that the alignment of business (design and operation of an enterprise) and the supporting IT system versus the strategy fails. This is explicitly the case for all intangible document-driven services (financial services,

insurance, pharmaceuticals, engineering contractors etc) that are produced interactively with the customer. We are able to engineer high quality artifacts in many domains such as electronics, telecommunication, cars etc. if and only if we apply the underlying scientific foundations and modeling methodologies well. Engineering contractors are typically not (yet) able to meet the before-mentioned requirements. Their IT systems focus on practical solutions for symptoms of problems.

5 Root Cause of the Problems

The root cause is the lack of scientific foundations in the engineering of those enterprises that produce document-driven services interactively with the customer. Design science of Hevner [7] describes the engineering of artifacts (in this case products/services, enterprises, IT systems) and claims that the engineering sciences should also be rooted in the empirical sciences. Since enterprises are social systems operating in "the real world" we need the behavioral sciences. For any framework, methodology or technology that is not founded in the empirical sciences we are unable to validate models against reality, which is mandatory in engineering, as argued by Dietz and Hoogervorst [2]. The capability to aggregate unlimited large business process models at runtime with formally correct execution is also mandatory for this domain. Van Nuffel et al [4] propose enhancements for BPMN (Business Process Model Notation, OMG) [3] for i) a lack of high quality formal languages (lack of completeness, construct excess, construct overload, construct redundancy, ambiguity) to represent models; ii) the inability to translate models (except small trivial models) into reliable executable systems. The required enhancements for BPMN are not (yet) available [4].

6 Theory, Methodology and Scientific Foundations

The proposed approach is founded on research conducted at the Delft University of Technology and several other universities, the CIAO! Consortium [5], the Enterprise Engineering Institute [6] and proprietary company research. The design and modeling of the conceptual enterprise models of the documentation, the enterprise and the IT system are all engineering artifacts and are therefor all founded on the engineering sciences, e.g. Design Science [7] and the behavioral sciences. The applied scientific foundation is the theory of Enterprise Ontology [8]. Enterprise ontology has by now a well-founded ontological appropriateness and truthfulness claim based on a reasonable set of industrial case studies. The applied methodology is DEMO (Design and Engineering Methodology for Organizations) [8], derived from the theory of Enterprise Ontology. DEMO delivers high quality (C4-ness; coherent, consistent, concise and comprehensive) enterprise models in a formal language. The high quality enterprise models enable the construction of a formally correct software engine.

7 Technical Documentation and Related IT System Engineering

A cornerstone of the DEMO conceptual modeling approach is that: i) the technical documentation to be produced, ii) the organization or enterprise (actors, communication,

coordination of the production) that produces the documentation and iii) the supporting IT system, are closely related engineering artifacts. Example: the engineering of a bicycle factory is directly derived from the engineering of a bicycle. A supporting IT system for the bicycle factory is also an engineering artifact that is directly derived from the way a bicycle factory operates. This is common practice and successful in manufacturing. However, for document-driven enterprises this is not (yet) applied in a proper way due to a lack of applicable science (enterprise ontology) and methodology (DEMO). For this domain, engineering contractors the produced technical documentation is also controlling the production and maintenance of the engineering contractor artifact (refinery, energy plant etc.).

For each new project these three artifacts have to be designed in advance and represented in high quality DEMO conceptual enterprise models. Enterprises are complex entities, recursively composed of aggregated sub-enterprises (example: multinational-national-departmental-sub-department) Recursive modeling into finer detail delivers finer detailed enterprise models. These aggregated conceptual models of unlimited size and complexity specify simultaneous:

- The construction and the operation of the enterprise including all communication and coordination;
- The detailed production for each actor- employee in the enterprise, and
- The IT system that monitors and controls the enterprise operation.

This may seem very elaborate but these models have a high degree of reusability and the overall modeling effort is small.

The core of the software technology is a software engine, the DEMO processor.

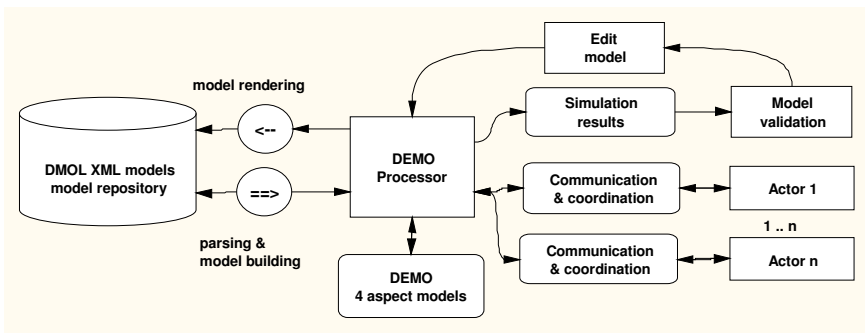


Fig. 1. DEMO model simulation and validation of an (elementary) enterprise model

The DEMO processor constructs and executes DEMO models, enables model simulation, model validation and incremental model improvement during development (fig. 1). The development process starts with the modeling stage for an enterprise (nested in enterprises etc.) that delivers the 4 graphical DEMO aspect models. Then the models are translated (not programmed) on the DEMO processor in a 1:1 process. The DEMO processor executes the enterprise model dynamically and delivers simulation results for model validation. The execution involves all

communication and coordination in a *prescriptive* way between all (human) actors (1..n) of the enterprise, similar to a workflow system.

If model validation fails the model under execution can be edited and the simulation – validation cycle is repeated. After successful validation the model is rendered as an DMOL (DEMO Modeling Language) XML file with full state information and stored in a model repository. At any time the original model, in its current state, can be parsed and rebuild for further simulation. A model representation rendered in (near) natural language enables the (re)construction of the 4 DEMO aspect models. This process is repeated until we have a complete model repository of all elementary enterprises, each of them producing an elementary production component.

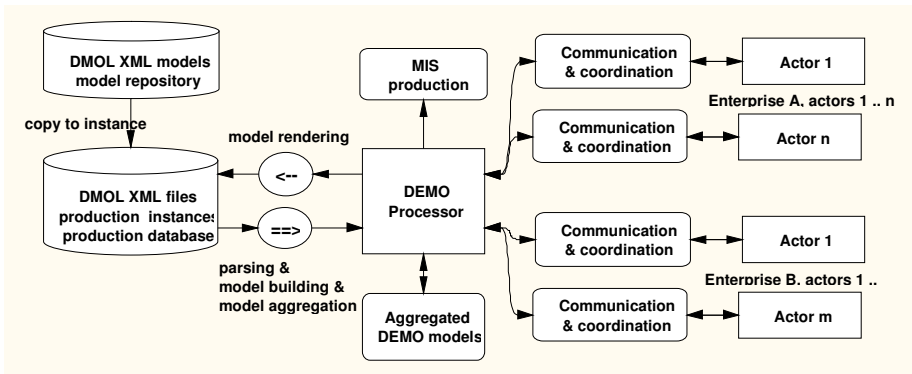


Fig. 2. DEMO processor in production environment

In production (fig 2) for each elementary production component a production model instance for that elementary enterprise is created from the model repository by a copy operation. Production instances are aggregated by the DEMO processor into a model under execution that represents the production of all aggregated enterprises A and B. Model building, aggregation and optional destruction is executed at runtime. In production the DEMO processor executing the aggregated model delivers all communication and coordination for each actor in each enterprise, as a workflow system. For each production step *descriptive* data can be delivered to a MIS (Management Information System). At any time an aggregated model can be decomposed into its aggregating DMOL file components, stored in the production database. When the production resumes the original production model reconstructs itself, with full state information and continues communication and coordination.

8 State of the Art of the DEMO Processor

DEMO and enterprise ontology are a new methodology and theory, with still a small number of industrial case studies but with convincing empirical validation [9]. The before-mentioned application development platform has been in use since 1997 in

several large professional applications, mostly financial services. The DEMO processor exists now as a prototype for proof of correctness. A professional production version is under development and will be made available i) for free as a modeling, simulation and education tool and ii) as a webservice production environment (cloud). A major bank-insurance company made the commitment to be the launching DEMO processor customer. Due to the nature of the application domain of engineering contractors, the potential benefits are assumed to be there more substantial. Interested parties are invited to discuss participation.

References

1. NEMI White Paper Report, In search of the perfect Bill of Materials (2002), <http://thor.inemi.org/webdownload/newsroom/Articles/BoMwhitepaper.pdf>
2. Dietz, J.L.G., Hoogervorst, J.A.P.: Language Design and Programming Methodology. LNCS, vol. 79. Springer, Heidelberg (1980)
3. Object Management Group Inc, 2009. Business Process Modeling Notation (BPMN) Specifications (2009), <http://www.omg.org/spec/BPMN/1.2/PDF/>
4. Nuffel van, D., Mulder, H., van Kervel, S.: Enhancing the formal foundations of BPMN using Enterprise Ontology. In: CAiSE CIAO (2009)
5. Ciao Consortium, <http://www.ciaonetwork.org>
6. DEMO Knowledge Centre; Enterprise Engineering institute, <http://www.demo.nl>
7. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1) (2004)
8. Dietz, J.L.G.: Enterprise Ontology. Springer, New York (2006)
9. Mulder, J.B.F.: Rapid Enterprise Design. PhD thesis (2008); ISBN 90-810480-1-5

Publishing Open Data and Services for the Flemish Research Information Space

Christophe Debruyne¹, Pieter De Leenheer^{2,3}, Peter Spyns⁴, Geert van Grootel⁴,
and Stijn Christiaens³

¹ STARLab, Vrije Universiteit Brussel, Brussels, Belgium

² Business Web & Media, VU University Amsterdam, Amsterdam, The Netherlands

³ Collibra nv/sa, Brussels, Belgium

⁴ Flemish Dept. of Economy, Science, and Innovation, Brussels, Belgium

Abstract. The Flemish public administration aims to integrate and publish all research information on a portal. Information is currently stored according to the CERIF standard modeled in (E)ER and aimed at extensibility. Solutions exist to easily publish data from databases in RDF, but ontologies need to be constructed to render those meaningful. In order to publish their data, the public administration and other stakeholders first need to agree on a shared understanding of what exactly is captured and stored in that format. In this paper, we show how the use of the Business Semantics Management method and tool contributed in achieving that aim.

Keywords: ontology development, methodology, social process, business semantics management, fact-orientation, natural language.

1 Introduction: An Innovation Information Portal for Flanders

For a country or region in the current knowledge economy, it is crucial to have a good overview of its science and technology base to develop an appropriate policy mix of measures to support and stimulate research and innovation. Also companies, research institutions and individual researchers can profit from the information maintained in such a portal. EWI¹ thus decided to launch the Flanders Research Information Space program (FRIS) to create a virtual research information space covering all Flemish players in the field of economy, science and innovation. The current version of this portal² contains, for instance, mash-ups of data on key entities (such as person, organization, and project; and their relationships) on a geographical map. Another aim of FRIS is to reduce the current administrative burden for universities as they are confronted with repeatedly reporting the same information in different formats to various institutions. Indeed, if all information would be centralized and accessible in a uniform way, creating services for such reports would greatly facilitate the reporting process. Before data can be centralized, this initiative faces two problems: 1) capturing the semantics of the domain in an ontology and 2) appropriately annotate or commit the heterogeneous data sources to that ontology.

¹ The Department of Economy, Science and Innovation of the Flemish Government
<http://www.ewi-vlaanderen.be/>

² <http://www.researchportal.be/>

As we will explain in Section 2, integrating all information and reducing the administrative burden faces some problems for which appropriate data governance methods and tools are needed. Such method and tool is presented in Section 3 and we end this paper with a conclusion in Section 4.

2 Problem: Heterogeneous Information Sources

Universities receiving funding from the Flemish government are asked to regularly report the same information to different organizations (local and international). As there is little alignment between those reports, universities are confronted with repeatedly sending the same information in other formats, other structures or according to different classifications not always compatible with each other³. This creates a heavy administrative burden on those knowledge institutions. Universities furthermore store their information in *autonomously developed* information systems, adding to the complexity of the problem.

As the EU also wants to track all research information in Europe, they ask all universities to report using the Common European Research Information Format (CERIF) [4], a recommendation to EU-members for the storage and exchange of current research information. While the CERIF model, created with Entity-Relationship (ER) diagrams, allows for an almost unlimited flexibility on roles and classifications used with entities, the actual approach has shown its *limitations* when it comes to *communicating* the modeled domain facts to domain experts and end users. The learning curve for the domain experts to understand the ER model and translate it back to the conceptual level is quite steep. **Fig. 1** shows some CERIF entities, their attributes and relationships.

To populate the FRIS portal with all information provided by the delivered CERIF files and other heterogeneous sources, (i) a consensus amongst the involved parties on a common conceptual model for CERIF and the different classifications is needed (taking into account the non-technical expertise of most domain experts), (ii) an easy, repeatable process for validating and integrating the data from those sources and finally (iii) using that shared understanding to publish that information as in a generic way on the Web on which third parties can develop services (commercial or not, e.g. to produce the different reports) as demonstrated by other Linked Data initiatives.

3 Approach: Business Semantics Management

In order to overcome the above-mentioned difficulties, all these classifications and models need to be actualized and homogenized on a conceptual level, first within Flanders, later with more general and international classifications. The Business Semantics Management (BSM) [2] methodology was adopted to capture the domain knowledge inside CERIF and the different classifications. BSM adopts the Semantics of Business Vocabulary and Business Rules (SBVR) [1] to capture concepts and their relationships in facts. SBVR is a *fact-oriented* modeling approach. Fact-oriented modeling is a method for analyzing and creating conceptual schemas for information

³ Different classifications are used within Flanders: IWETO discipline codes, IWETO science domains, VLIR scientific disciplines, IWETO application domains, SOOI (based on the IWI-Web of Science codes), NABS (used for budgeting) and FOS (Fields of Science), etc.

systems starting from (usually binary) relationships expressed as part of human-to-system communication. Using concepts and a language people are intended to readily understand, fact-oriented modeling helps ensuring the quality of a database application without caring about any implementation details of the database, including e.g. the grouping itself of linguistic concepts into records, relations, ... In fact-oriented approaches, every concept plays roles with other concepts, and those roles may be constrained. It is those constraints that allow the implementer of a database (or in fact an algorithm) to determine whether some linguistic concept becomes an entity or an attribute, or whether a role turns out to be an attribute relationship or not. This is different from other approaches such as (E)ER and UML, where these decisions are made at design time.

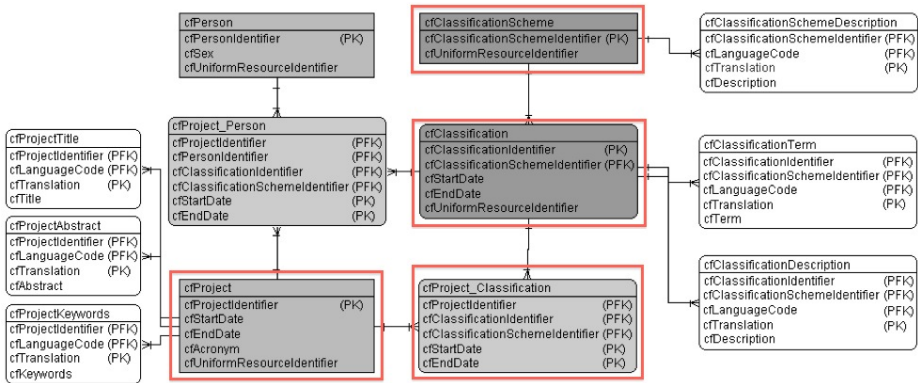


Fig. 1. The CERIF entity cfProject and its relationship with the entity cfProject_Classification (linked by the two identifiers of the linked entities). A CERIF relationship is always semantically enriched by a time-stamped classification reference. The classification record is maintained in a separate entity (cfClassification) and allows for multilingual features. Additionally, each classification record or instance requires an assignment to a classification scheme (cfClassificationSchemeIdentifier).

Business semantics management is the set of activities (depicted in Fig. 2) to bring business stakeholders together to collaboratively realize the reconciliation of their heterogeneous metadata; and consequently the application of the derived business semantics patterns to establish semantic alignment between the underlying data structures.

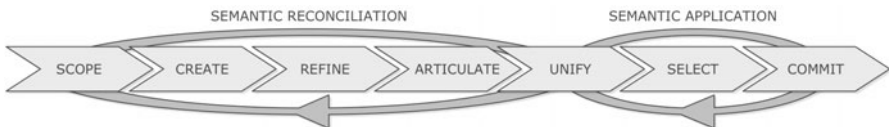


Fig. 2. Business Semantics Management overview: semantic reconciliation and application

The first cycle, *semantic reconciliation*, is supported by the Business Semantics Glossary (BSG) shown in Fig. 3. This figure shows a screenshot of the term “Project” (within the “Project” vocabulary of “CERIF” speech community that is part of the

“FRIS” semantic community). The software is currently deployed at EWI for managing business semantics of CERIF terms. A term (here “Project”) can be defined using one or more attributes such as definitions, examples, fact types, rules sets, categorization schemas (partly shown in taxonomy), and finally milestones for the lifecycle. “Project” in this case is a subtype of “Thing” and has two subtypes: ”large academic project” and “small industrial project”. Re governance: in the top-right corner is indicated which member in the community (here “Pieter De Leenheer”) carries the role of “steward”, who is ultimately accountable for this term. The status ”candidate” indicates that the term is not yet fully articulated: in this case “Project” only 37.5%. This percentage is automatically calculated based on the articulation tasks that have to be performed according to the business semantics management methodology. Tasks are related to defining attributes and are distributed among stakeholders and orchestrated using workflows.

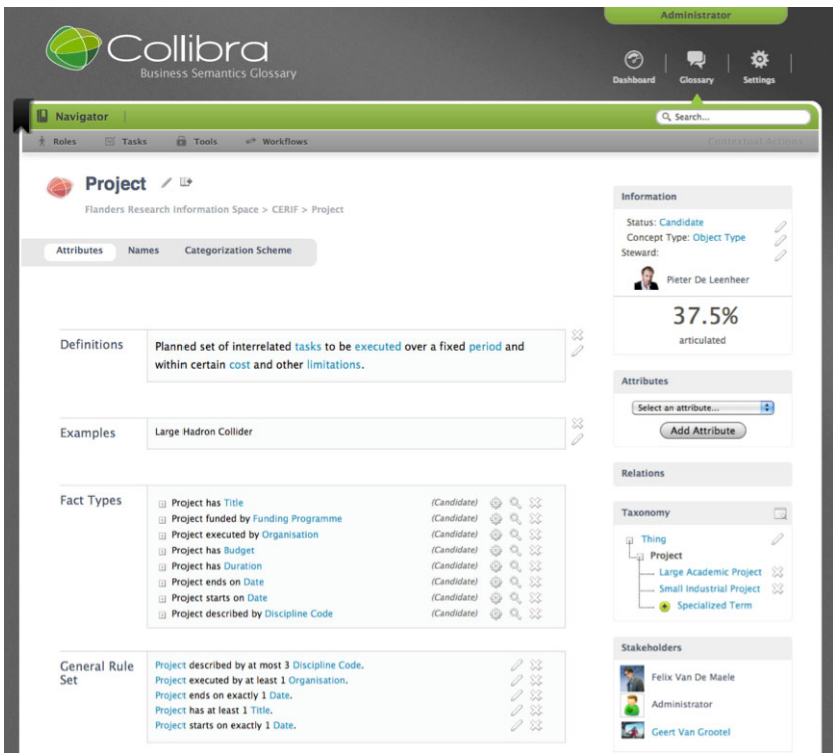


Fig. 3. Screenshot of Collibra’s BSG supporting the semantic reconciliation process of the BSM methodology by providing domain experts means to enter simple key facts in natural language, natural language definitions on facts and terms in those facts as well as constraints.

Applying BSM results in a community driven (e.g. representing the different classifications and models mentioned earlier), iteratively developed shared and agreed upon conceptual model in SVBR. This model then is automatically converted in a CERIF-based ER model and RDFS/OWL for Web publishing. **Fig. 4** shows a part of the generated OWL from the concept depicted in the previous figure. In this figure,

we see that `Project` is a `Class` and all instances of that class are also instances of entities with at least one value for the property `ProjectHasTitle`, one of the rules expressed in SBVR in **Fig. 3** (see general rule sets).

```

- <owl:Class rdf:about="http://labs.collibra.com/bsgtrunk/bin/view/Project/Project">
- <Project:ProjectHasBudget>
  <owl:Class rdf:about="http://labs.collibra.com/bsgtrunk/bin/view/Project/Budget">
  </Project:ProjectHasBudget>
  <dc:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1f5c7dcd-9942-4313-950d-d26e7bd140c6</dc:identifier>
- <rdfs:subClassOf>
  - <owl:Restriction>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    - <owl:onProperty>
      <owl:ObjectProperty rdf:about="http://labs.collibra.com/bsgtrunk/bin/view/Project/ProjectHasTitle"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>

```

Fig. 4. Screenshot of the OWL around `Project` generated by BSG. In this picture, we see that `Project` is a `Class` and all instances of that class are also instances of entities with at least one value for the property `ProjectHasTitle`.

The contents of the databases to be annotated can be published with off-the-shelf solutions such as D2R Server⁴. D2R Server generates an RDF a mapping for transforming the content of a database into RDF triples. This mapping – also described in RDF – contains a “skeleton” RDFS of classes and properties that are based on the database schema. **Fig. 5** below depicts a part of the generated mapping file around the table containing information around projects.

```

@prefix map: <file:///.../OSCB/d2r-server-0.7/map.n3#> .
@prefix vocab: <http://192.168.0.136:5432/vocab/resource/> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
...
map:CFPROJ a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "CFPROJ/@CFPROJ.CFPROJID|urlencode@" ;
  d2rq:class vocab:CFPROJ;
  d2rq:classDefinitionLabel "EWI.CFPROJ";
...

```

Fig. 5. Part of the generated mapping file by D2R server, it maps the table `CFProj` to the generated `CFPROJ` RDFS class. It uses the primary key to generate a unique ID and the class definition label is taken from the table’s name.

Even though classes and properties are generated and populated with instances, these RDF triples are not semantic as they stem from one particular information system(’s database schema). That RDFS is then aligned with the generated RDFS/OWL classes and properties generated from the BSM ontology. The commitments described in the previous section are used as a guideline to create this alignment. **Fig. 6** below shows the changes (highlighted) made on the generated mapping file with the ontology. The ontology can then be used to access the data.

⁴ <http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/>

```

@prefix ont: <file:///.../Project.rdf#> .
...
map:CFPROJ a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "CFPROJ/@CFPROJ.CFPROJID|urlencode@" ;
    d2rq:class ont:Project;
    d2rq:classDefinitionLabel "Project";
...

```

Fig. 6. Modified mapping file with the ontology exported from BSG. An extra namespace (for the exported ontology) is added and the generated classes and properties are appropriately annotated with that ontology.

4 Conclusion: Inclusion of the Method and Tool in the Portal's Architecture

This paper presented a case of applying Business Semantics Management (BSM) in a Flemish public administration for the creation of an innovation information portal. The purpose of this portal is to integrate and provide a uniform access mechanism to all research information in Flanders as RDF, allowing third parties to create services around that data (e.g. reporting) and removing some of the administrative burden of universities. Business Semantics Management method and tools helped in constructing an ontology and was well received by the users.

Publication of data in relational databases became fairly easy with solutions such as D2R server. The triples generated by such tools are rendered “meaningful” by exporting the ontology into an implementation in RDFS/OWL and use these to annotate the instances, since the ontology is the result of collaboration and meaning agreements between stakeholders representing autonomously developed information systems. Future work in that area consists of developing a flexible layer between the generated RDF triples from existing tools and the generated ontology from the Business Semantics Glossary.

References

- [1] OMG SBVR, version 1.0, <http://www.omg.org/spec/SBVR/1.0/>
- [2] 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)
- [3] Halpin, T.: *Information Modeling and Relational Databases*. Morgan Kaufmann, San Francisco (2008)
- [4] Jörg, B.: CERIF: The common European research information format model. *Data Science Journal* (9), 24–31 (2010)
- [5] Jörg, B., van Grootel, G., Jeffery, K.: Cerif2008xml - 1.1 data exchange format specification. Technical report, euroCRIS (2010)
- [6] Spyns, P., Tang, Y., Meersman, R.: An ontology engineering methodology for DOGMA. *Applied Ontology* 3(1-2), 13–39 (2008)
- [7] Spyns, P., van Grootel, G., Jörg, B., Christiaens, S.: Realising a Flemish government innovation information portal with business semantics management. In: Stempfhuber, M., Thidemann, N. (eds.) *Connecting Science with Society. The Role of Research Information in a Knowledge-Based Society*: University of Aalborg, Aalborg University Press (2010)

Author Index

- Abelló, Alberto 108
Aguilar, José Alfonso 14
Aguilera, David 323
Ait-Ameur, Yamine 98
- Bach Pedersen, Torben 364
Bakhtouchi, Abdelghani 98
Bartie, Phil 231
Battaglia, Antonino 358
Bellatreche, Ladjel 98
Bergamaschi, Sonia 328
Bhatt, Ankur 343
Bianchini, Devis 34
Billen, Roland 230, 322
Bontemps, Yves 377
Boukhebouze, Mohamed 24
Brdjanin, Drazen 292
Briand, Lionel 338
Brisaboa, Nieves R. 241
- Cabot, Jordi 332
Caniupán, Mónica 75
Chen, Tao 251
Christiaens, Stijn 389
Clasen, Cauê 332
Clementini, Eliseo 231
Corchuelo, Rafael 345
- Dau, Frithjof 45
De Antonellis, Valeria 34
Debruyne, Christophe 389
de Hemptinne, Gregoire 377
Delbaere, Marc 377
De Leenheer, Pieter 389
de Oliveira, Jose Palazzo Moreira 2
De Troyer, Olga 120
Di Tria, Francesco 86
- Egenhofer, Max J. 261
El Dammagh, Mohammed 120
Embley, David W. 183
Erbin, Lim 24
- Faulkner, Stéphane 44
Frasincar, Flavius 1
Frixione, Marcello 210
- Gailly, Frederik 163
García-Ranea, Raúl 323
Garrigós, Irene 14, 336
Geerts, Guido L. 291
Glorio, Octavio 336
Golfarelli, Matteo 358
Gómez, Cristina 323
Guerra, Francesco 328
Guizzardi, Giancarlo 161
- Hallot, Pierre 230, 322
Hernández, Inma 345
Hernández, Paul 336
Houben, Geert-Jan 1
- Jansen, Slinger 151
Jouault, Frédéric 332
Jureta, Ivan J. 44
- Kabbedijk, Jaap 151
Käkölä, Timo K. 119
Kästner, Christian 130
Khan, Ateeq 130
Kingham, Simon 231
Köppen, Veit 130
- Lechtenböcker, Jens 65
Lefons, Ezio 86
Levin, Ana M. 203
Liddle, Stephen W. 183
Lieto, Antonio 210
Linner, Konrad 55
Loebe, Frank 193
Lonsdale, Deryle W. 183
Lopes, Giseli Rabello 2
Lozano, Angela 141
Luaces, Miguel R. 241
Lukyanenko, Roman 220
- Maric, Slavko 292
Marth, Kevin 312
Masolo, Claudio 173
Mazón, Jose-Norberto 14, 65, 336
McGinnes, Simon 4
Melchiori, Michele 34
Mens, Kim 118

- Middelfart, Morten 364
 Mitsch, Stefan 271
 Moreau, Guillaume 281
 Moro, Mirella Moura 2
- Neto, Waldemar Pires Ferreira 24
 Neumayr, Bernd 55
- Olivé, Antoni 323, 349, 353
- Panesar-Walawege, Rajwinder Kaur
 338
- Parsons, Jeffrey 220
 Pastor, Oscar 161
 Poels, Geert 163
- Reinhartz-Berger, Iris 118, 302
 Reitsma, Femke 231
 Ren, Shangping 312
 Retschitzegger, Werner 271
 Ritter, Daniel 343
 Rivero, Carlos R. 345
 Rizzi, Stefano 358
 Rodríguez, M. Andrea 241
 Romero, Oscar 108
 Rossi, Matti 291
 Rota, Silvia 328
 Ruiz, David 345
- Saake, Gunter 130
 Sabetzadeh, Mehrdad 338
 Sancho, Maria-Ribera 349, 353
- Schneider, Markus 251
 Schrefl, Michael 55
 Schwinger, Wieland 271
 Sertkaya, Barış 45
 Servières, Myriam 281
 Simitis, Alkis 357
 Skhiri, Sabri 377
 Skyberg Knutsen, Torbjørn 338
 Spyns, Peter 389
 Sturm, Arnon 118
- Tangorra, Filippo 86
 Thiran, Philippe 1
 Tort, Albert 349
 Tourwé, Tom 371
 Tran, Nam-Luc 377
 Trujillo, Juan 65
 Tsporkova, Elena 371
 Tsoury, Arava 302
- Vaisman, Alejandro 75
 van der Kroon, Matthijs 203
 van Grootel, Geert 389
 van Kervel, Steven J.H. 383
 Van Mingroot, Hans 357
 Velegarakis, Yannis 328
 Villegas, Antonio 353
- Wand, Yair 161
- Zaki, Chamseddine 281
 Zimányi, Esteban 44, 230