

María José Escalona  
José Cordeiro  
Boris Shishkov (Eds.)

Communications in Computer and Information Science

303

# Software and Data Technologies

6th International Conference, ICSOFT 2011  
Seville, Spain, July 2011  
Revised Selected Papers

Editorial Board

Simone Diniz Junqueira Barbosa

*Pontifical Catholic University of Rio de Janeiro (PUC-Rio),  
Rio de Janeiro, Brazil*

Phoebe Chen

*La Trobe University, Melbourne, Australia*

Alfredo Cuzzocrea

*ICAR-CNR and University of Calabria, Italy*

Xiaoyong Du

*Renmin University of China, Beijing, China*

Joaquim Filipe

*Polytechnic Institute of Setúbal, Portugal*

Orhun Kara

*TÜBİTAK BİLGEM and Middle East Technical University, Turkey*

Tai-hoon Kim

*Konkuk University, Chung-ju, Chungbuk, Korea*

Igor Kotenko

*St. Petersburg Institute for Informatics and Automation  
of the Russian Academy of Sciences, Russia*

Dominik Ślęzak

*University of Warsaw and Infobright, Poland*

Xiaokang Yang

*Shanghai Jiao Tong University, China*

María José Escalona  
José Cordeiro Boris Shishkov (Eds.)

# Software and Data Technologies

6th International Conference, ICSOFT 2011  
Seville, Spain, July 18-21, 2011  
Revised Selected Papers

 Springer

## Volume Editors

María José Escalona  
Universidad de Sevilla  
ETS Ingeniería Informática  
Av. Reina Mercedes S/N  
41012 Sevilla, Spain  
E-mail: mjescalona@us.es

José Cordeiro  
INSTICC / IPS  
Department of Systems and Informatics  
Rua do Vale de Chaves, Estefanilha  
2910-761 Setúbal, Portugal  
E-mail: jose.cordeiro@estsetubal.ips.pt

Boris Shishkov  
IICREST  
1618 Sofia, Bulgaria  
E-mail: b.b.shishkov@iicrest.eu

ISSN 1865-0929  
ISBN 978-3-642-36176-0  
DOI 10.1007/978-3-642-36177-7  
Springer Heidelberg Dordrecht London New York

e-ISSN 1865-0937  
e-ISBN 978-3-642-36177-7

Library of Congress Control Number: 2012955613

CR Subject Classification (1998): D.2.1-3, D.2.5, D.2.9, D.3.2, I.2.4, H.2.8

© Springer-Verlag Berlin Heidelberg 2013

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

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

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

Printed on acid-free paper

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



# Preface

The present book includes extended and revised versions of a set of selected papers from the 6th International Conference on Software and Data Technologies—ICSOFT 2011. The conference was held in Seville, Spain, in collaboration with the University of Seville and the Escuela Técnica Superior de Ingeniería Informática (ETSII) at the University of Seville. Sponsored by the Institute for Systems and Technologies of Information, Control and Communication (INSTICC), ICSOFT was organized in cooperation with IICREST (Interdisciplinary Institute for Collaboration and Research on Enterprise Systems and Technology), CEPIS (Council of European Professional Informatics Societies), ATI (Asociación de Técnicos de Informática), FIDETIA (Fundación para la Investigación y el Desarrollo de las Tecnologías de la Información en Andalucía), and INES (Iniciativa Española de Software y Servicios).

The purpose of ICSOFT 2011 was to bring together researchers and practitioners interested in information technology and software development. The conference tracks were “Enterprise Software Technology,” “Software Engineering,” “Distributed Systems,” “Data Management” and “Knowledge-Based Systems”.

Software and data technologies are essential for developing any computer information system: ICSOFT’s scope encompassed a large number of research topics and applications, from programming issues to the more abstract theoretical aspects of software engineering; from databases and data-warehouses to the most complex management information systems; knowledge-base systems; distributed systems, ubiquity, data quality and many other topics.

ICSOFT 2011 received 220 paper submissions from 48 countries. To evaluate each submission, a double-blind paper evaluation method was used: each paper was reviewed by at least two internationally known experts from the ICSOFT Program Committee. Only 27 papers were selected to be published and presented as full papers (10 pages in proceedings / 30-min oral presentation). Additionally, 62 papers were accepted as short papers (6 pages / 20-min oral presentation)—for a total of 89 oral presentations—and 33 papers as posters. The full-paper acceptance ratio was thus 12.3%, while the total oral-paper acceptance ratio was 40.4%.

The quality of the papers presented stems directly from a successful and solid conference, which would not have been possible but for the dedicated effort of a complex organizing structure, from the Steering and Scientific Committees to the INSTICC team responsible for handling all secretariat and logistical details. A word of appreciation is also due to the conference keynote speakers and to the many authors and attendants who gave us the honor of helping present their ideas and hard work to the scientific community.

We hope that you will find these papers interesting and consider them a helpful reference in the future when addressing any of the research areas mentioned above.

March 2012

María José Escalona  
José Cordeiro  
Boris Shishkov

# Organization

## Conference Co-chairs

José Cordeiro Polytechnic Institute of Setúbal / INSTICC,  
Portugal  
Maria Jose Escalona University of Seville, Spain

## Program Chair

Boris Shishkov IICREST, Bulgaria

## Organizing Committee

Patrícia Alves INSTICC, Portugal  
Sérgio Brissos INSTICC, Portugal  
Helder Coelhas INSTICC, Portugal  
Vera Coelho INSTICC, Portugal  
Andreia Costa INSTICC, Portugal  
Patrícia Duarte INSTICC, Portugal  
Bruno Encarnação INSTICC, Portugal  
Liliana Medina INSTICC, Portugal  
Carla Mota INSTICC, Portugal  
Raquel Pedrosa INSTICC, Portugal  
Vitor Pedrosa INSTICC, Portugal  
Daniel Pereira INSTICC, Portugal  
Cláudia Pinto INSTICC, Portugal  
José Varela INSTICC, Portugal  
Pedro Varela INSTICC, Portugal

## Program Committee

Alain Abran, Canada Farhad Arbab, The Netherlands  
Muhammad Abulaish, India Cyrille Artho, Japan  
Hamideh Afsarmanesh, Colin Atkinson, Germany  
The Netherlands Mortaza S. Bargh, The Netherlands  
Jacky Akoka, France Bernhard Bauer, Germany  
Markus Aleksy, Germany Nouredine Belkhatir, France  
Rafa E. Al-Qutaish, UAE Fevzi Belli, Germany  
Toshiaki Aoki, Japan Jorge Bernardino, Portugal  
Keijiro Araki, Japan Marko Boškovic, Canada  
Gabriela Noemí Aranda, Argentina Lydie du Bousquet, France

VIII Organization

Mark Van Den Brand,  
The Netherlands

Lisa Brownsword, USA

Manfred Broy, Germany

Dumitru Burdescu, Romania

Cristina Cachero, Spain

Fergal Mc Caffery, Ireland

Antoni Lluís Mesquida Calafat, Spain

José Antonio Calvo-Manzano, Spain

Gerardo Canfora, Italy

Mauro Caporuscio, Italy

Cinzia Cappiello, Italy

Cagatay Catal, Turkey

Krzysztof Cetnarowicz, Poland

Kung Chen, Taiwan

Shiping Chen, Australia

Yoonsik Cheon, USA

Chia-Chu Chiang, USA

Peter Clarke, USA

Rem Collier, Ireland

Kendra Cooper, USA

Sergiu Dascalu, USA

Steven Demurjian, USA

Giovanni Denaro, Italy

María J. Domínguez-Alda, Spain

Juan C. Dueñas, Spain

Philippe Dugerdil, Switzerland

Jürgen Ebert, Germany

Fikret Ercal, USA

Maria Jose Escalona, Spain

João Faria, Portugal

Cléver Ricardo Guareis de Farias,  
Brazil

Luis Fernandez, Spain

Rita Francese, Italy

Kehan Gao, USA

Jose M. Garrido, USA

Nikolaos Georgantas, France

Paola Giannini, Italy

J. Paul Gibson, France

Itana Gimenes, Brazil

Athula Ginige, Australia

Juan Carlos Granja, Spain

Des Greer, UK

Slimane Hammoudi, France

Christian Heinlein, Germany

Markus Helfert, Ireland

Brian Henderson-Sellers, Australia

Jose Luis Arciniegas Herrera,  
Colombia

Jose R. Hilera, Spain

Jang-eui Hong, Republic of Korea

Shihong Huang, USA

Ilian Ilkov, The Netherlands

Ivan Ivanov, USA

Bharat Joshi, USA

Yong-Kee Jun, Republic of Korea

Sanpawat Kantabutra, Thailand

Dimitris Karagiannis, Austria

Foutse Khomh, Canada

Roger (Buzz) King, USA

Mieczyslaw Kokar, USA

Jun Kong, USA

Dimitri Konstantas, Switzerland

Walter Kusters, The Netherlands

Martin Kropp, Switzerland

Patricia Lago, The Netherlands

Philippe Lahire, France

Konstantin Läufer, USA

Raimondas Lencevicius, USA

Hareton Leung, China

Hua Liu, USA

David Lorenz, Israel

Zakaria Maamar, UAE

Ricardo J. Machado, Portugal

Leszek Maciaszek, Australia

David Marca, USA

Eda Marchetti, Italy

Katsuhisa Maruyama, Japan

Antonia Mas, Spain

Tommaso Mazza, Italy

Bruce McMillin, USA

Stephen Mellor, UK

Marian Cristian Mihaescu, Romania

Dimitris Mitrakos, Greece

Valérie Monfort, Tunisia

Mattia Monga, Italy

Sandro Morasca, Italy

Paolo Nesi, Italy

Jianwei Niu, USA

Rory O'Connor, Ireland  
 Pasi Ojala, Finland  
 Vincenzo Pallotta, Switzerland  
 Patrizio Pelliccione, Italy  
 Massimiliano Di Penta, Italy  
 César González Pérez, Spain  
 Pascal Poizat, France  
 Andreas Polze, Germany  
 Christoph von Praun, Germany  
 Rosario Pugliese, Italy  
 Anders Ravn, Denmark  
 Werner Retschitzegger, Austria  
 Claudio de la Riva, Spain  
 Colette Rolland, France  
 Gustavo Rossi, Argentina  
 Gunter Saake, Germany  
 Krzysztof Sacha, Poland  
 Francesca Saglietti, Germany  
 Beijun Shen, China  
 Boris Shishkov, Bulgaria  
 Yanfeng Shu, Australia  
 Marten van Sinderen, The Netherlands  
 Harvey Siy, USA

Yeong-tae Song, USA  
 Cosmin Stoica Spahiu, Romania  
 George Spanoudakis, UK  
 Peter Stanchev, USA  
 Davide Tosi, Italy  
 Sergiy Vilkomir, USA  
 Gianluigi Viscusi, Italy  
 Florin Vrejoiu, Romania  
 Christiane Gresse von Wangenheim,  
 Brazil  
 Martijn Warnier, The Netherlands  
 Ing Widya, The Netherlands  
 Dietmar Wikarski, Germany  
 Eric Wong, USA  
 Jongwook Woo, USA  
 Qing Xie, USA  
 Haiping Xu, USA  
 Tuba Yavuz-kahveci, USA  
 I-Ling Yen, USA  
 Fatiha Zaidi, France  
 Xiaokun Zhang, Canada  
 Hong Zhu, UK  
 Elena Zucca, Italy

### Auxiliary Reviewers

Narciso Albarracin, USA  
 Tom Arbuckle, Ireland  
 Carmen Bratosin, The Netherlands  
 Patricia Shiroma Brockmann,  
 Germany  
 Félix Cuadrado, Spain  
 Subhomoy Dass, USA  
 Boni García, Spain  
 Rodrigo Garcia-Carmona, Spain  
 Michiel Helvensteijn, The Netherlands  
 Joseph Kaylor, USA  
 Dae S. Kim-Park, Spain  
 Ruurd Kuiper, The Netherlands

James Mulcahy, USA  
 Rob van Nieuwpoort, The Netherlands  
 Behrooz Nobakht, The Netherlands  
 Marcos Palacios, Spain  
 Jesús Pardillo, Spain  
 Ignazio Passero, Italy  
 Jose Proenca, Belgium  
 Alexander Schneider, Germany  
 Davide Taibi, Italy  
 Saleem Vighio, Denmark  
 Anton Wijs, The Netherlands  
 Yunqi Ye, USA

### Invited Speakers

Ivan Ivanov, SUNY Empire State College, USA  
 Antonia Bertolino, Italian National Research Council – CNR, Italy  
 David Marca, University of Phoenix, USA  
 Oscar Pastor, Universidad Politécnica de Valencia, Spain

# Table of Contents

## Invited Papers

The Impact of Emerging Computing Models on Organizational Socio-technical System . . . . .	3
<i>Ivan I. Ivanov</i>	
On-the-Fly Dependable Mediation between Heterogeneous Networked Systems . . . . .	20
<i>Antonia Bertolino, Antonello Calabrò, Felicita Di Giandomenico, Nicola Nostro, Paola Inverardi, and Romina Spalazzese</i>	
SADT/IDEF0 for Augmenting UML, Agile and Usability Engineering Methods . . . . .	38
<i>David A. Marca</i>	
From Requirements to Code: A Full Model-Driven Development Perspective . . . . .	56
<i>Óscar Pastor, Marcela Ruiz, and Sergio España</i>	

## Part I: Enterprise Software Technology

Enabling Automatic Process-Aware Collaboration Support in Software Engineering Projects . . . . .	73
<i>Gregor Grambow, Roy Oberhauser, and Manfred Reichert</i>	

## Part II: Software Engineering

Hybrid Debugging of Java Programs . . . . .	91
<i>Christian Hermanns and Herbert Kuchen</i>	
Combined Constraint-Based Analysis for Efficient Software Regression Detection in Evolving Programs . . . . .	108
<i>Anh D. Le, Tho T. Quan, Nguyen T. Huynh, Phung H. Nguyen, and Nhat-Van Le</i>	
Requirements-Driven Iterative Project Planning . . . . .	121
<i>Yves Wautelet, Manuel Kolp, and Stephan Poelmans</i>	
An Approach for Model-Driven Design and Generation of Performance Test Cases with UML and MARTE . . . . .	136
<i>Antonio García-Domínguez, Inmaculada Medina-Bulo, and Mariano Marcos-Bárcena</i>	

Typing Legacy COBOL Code ..... 151  
*Alvise Spanò, Michele Bugliesi, and Agostino Cortesi*

A Repository for Integration of Software Artifacts with Dependency  
Resolution and Federation Support ..... 166  
*Rodrigo García-Carmona, Félix Cuadrado, Juan C. Dueñas, and  
Álvaro Navas*

Automated System Testing of Dynamic Web Applications ..... 181  
*Hideo Tanida, Mukul R. Prasad, Sreeranga P. Rajan, and  
Masahiro Fujita*

**Part III: Distributed Systems**

Technologies for Autonomic Dependable Services Platform:  
Achievements and Future Challenges ..... 199  
*Eila Ovaska, Liliana Dobrica, Anu Purhonen, and Marko Jaakola*

**Part IV: Data Management**

Extracting the Main Content of Web Documents Based on Character  
Encoding and a Naive Smoothing Method ..... 217  
*Hadi Mohammadzadeh, Thomas Gottron, Franz Schweiggert, and  
Gholamreza Nakhaeizadeh*

Facilitating Structuring of Information for Business Users with Hybrid  
Wikis ..... 237  
*Florian Matthes, Christian Neubert, and Alexander Steinhoff*

**Part V: Knowledge-Based Systems**

Statistical and Possibilistic Methodology for the Evaluation  
of Classification Algorithms ..... 255  
*Olgierd Hryniewicz*

What Else Can Be Extracted from Ontologies? Influence Rules ..... 270  
*Barbara Furletti and Franco Turini*

**Author Index** ..... 287

# **Invited Papers**



# The Impact of Emerging Computing Models on Organizational Socio-technical System

Ivan I. Ivanov

State University of New York - Empire State College, Long Island Center, NY 11788, U.S.A.  
Ivan.Ivanov@esc.edu

**Abstract.** Consolidated Enterprise IT solutions have proven to enhance business efficiency when significant fractions of local computing activities are migrating away from desktop PCs and departmental servers and are being integrated and packaged on the Web into “the computing cloud.” Whether referred to Grid, Utility or Cloud Computing, the idea is basically the same: instead of investing in and maintaining expensive applications and systems, users access and utilize dynamic computing structures to meet their fluctuating demands on IT resources and pay a fixed subscription or an actual usage fee. The immense economic demands in the last several years, in conjunction with the immediate reduction of upfront capital and operational costs when cloud-based services are employed, increase the speed and the scale of cloud computing adoption both horizontally -across industries-, and vertically –in organizations’ technology stacks.

In actuality, the radical changes for organizations are in rethinking and reengineering their traditional IT resources, advancing them with cloud architectures and implementing services based on dynamic computing delivery models. The changes and business transformations are underway on a large scale, from providers and customers to vendors and developers. The key issues are not only in economics and management, but essentially how emerging IT models impact organizational structure, capabilities, business processes, and consequential opportunities. This paper explores the impact of the dynamic computing models on the organizational socio-technical system and provides the author’s vision and experience in strategizing and utilizing emerging cloud-based applications and services.

**Keywords:** Cloud Computing, Virtualization, On-demand Services, Cloud-based Services, Dynamic Structures, Emerging Technologies, Socio-technical System, Systemic Effect, IT Architecture, Enterprise Architecture.

## 1 Introduction

Increasingly we have witnessed how business success and economic opportunities steadily depend on IT-enabled capabilities and IT-driven business transformations. In today’s global digital economy, the technology and business domains are colliding forcefully than ever and new business models and growing prospects emerge.

The IT and especially emerging technologies profoundly change how companies create value both within specific industries, and through industry boundaries.

Consolidated Enterprise IT solutions have proven to enhance business efficiency when significant fractions of local computing activities are migrating away from desktop PCs and departmental servers and are being integrated and packaged on the Web into “the computing cloud.” Whether referred to as Grid, Utility or Cloud Computing, the idea is basically the same: instead of investing in and maintaining expensive applications and systems, users access and efficiently utilize dynamic computing structures to meet their fluctuating demands on IT resources, and pay a fixed subscription or an actual usage fee [1].

Insightful businesses and organizations grasp the “grid,” “virtual,” or “cloud” ideas, discerning what the emerging computing services and models proffer, gauge how they can utilize them to confront business challenges to create a competitive advantage. Lynda Applegate’s textbook, *Corporate Information Strategy and Management*, illuminates this “jump-on-the-ball” approach comparing it with a start of a football game: the referee blows the whistle to start the play and, immediately, all players on both teams jump on the ball. Similarly, executives often use the “jump-on-the-ball” approach particularly during emergence of a new business phenomenon or a novel technology [2]. This approach was popular and widely implemented in the 1990s during the “dot-com” era when many blinded by the Internet phenomenon lost the ability to stick to fundamental business principals when planning new sustainable business ventures.

Although organizations today are facing increasing pressures with respect to innovations, operational excellence, and performance efficiency, the economic difficulties and cautiousness are keeping the costs for novel technology at an absolute minimum. In actuality, the radical changes for organizations are in rethinking and reengineering their traditional IT architectures to one with a creative digital business mindset. These transformations require new advanced knowledge and skills sets with emerging technologies, particularly, virtualization, cloud architectures and services based on more dynamic computing delivery models. The changes and business transformations are underway on a large scale, from the C-levels to end-users, from providers and vendors to developers and system architectures. The key issues extend beyond economics and management, and focus on how emerging IT models impact organizational strategy, capabilities, business processes, and consequential opportunities and rational value.

In the paper I will give emphasis to the impact of emerging dynamic computing models on the socio-technical system and related organizational and management considerations.

## **2 IT in the Organizational Context**

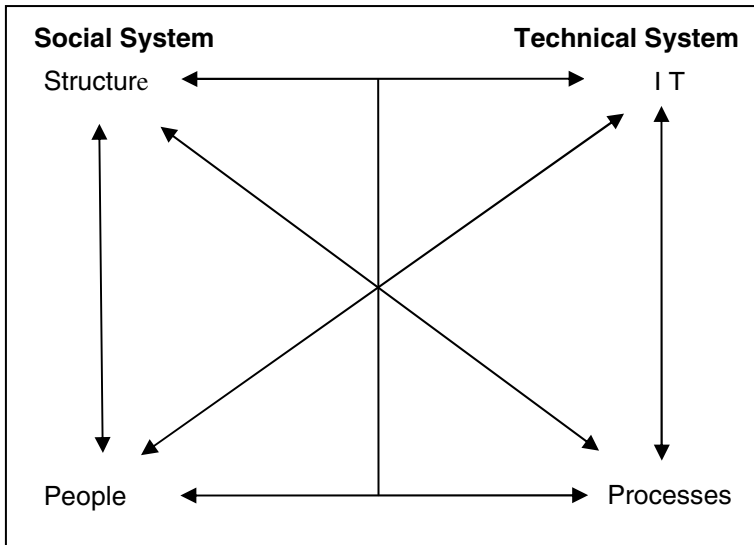
To explore the complexity of the problems and to avoid unrealistic expectations when employing new technologies and emerging models, a formal methodology of examining and evaluating IT in the organizational context can be applied. The contemporary approaches to Information Systems, and more specifically IT, encompass multidisciplinary theories and perspectives with no dominance of a single discipline or model.

## 2.1 Information Systems as Sociotechnical Systems

In the “Information Systems for Managers” text, the author Gabriele Picolli features IT as a critical component of a formal, sociotechnical information system designed to collect, process, store, and distribute information [3]. Authors, Kenneth and Jane Laudon in *Managing the Digital Firm*, define Information Systems as Sociotechnical Systems incorporating two approaches: *Technical* and *Behavioral*, with several major disciplines that contribute expertise and solutions in the study of Information systems [4].

The notion of above definitions is based on the Sociotechnical theory work developed by Tavistock Institute in London in mid-50s and 60-ties. The IT Sociotechnical approach not only visualizes the concept, but reveals the impact of new technologies and processes –the technical subsystem- on the entire work system, and the dependencies and interactions between all other facets and components of the sociotechnical system.

According to Picolli any organizational Information System can be represented as a Sociotechnical system which comprises four primary components that must be balanced and work together to deliver the information processing functionalities required by the organization to fulfill its information needs (Figure 1). The IS Sociotechnical model validates the most important components, and at the same time primary driving forces, within organizations: structure, people, process, and technology. The first two – *people* and *structure* – shape the *social subsystem*, and represent the human element of the IS. The latter two – *process* and *technology* (more specifically - I.T.) – contour the *technical subsystem* of the IS and relate to a wide range of IT resources and services intertwined with a series of steps to complete required business activities.



**Fig. 1.** Information systems primary components as a sociotechnical system [3]

In addition, the Laudons identify the disciplines that contribute respectively to the technical and the behavioral approaches. The founding disciplines of the technical approach to information systems include computer science, management science and operations research. Their knowledge areas communicate and provide physical technology and logical structures, mathematical models, management practices and optimization techniques when studying formal capabilities, performance and efficiency of the Information Systems. The technical subsystem is frequently the cause of behavior problems or issues such as design, implementation, and strategic integration of business innovations; alignment, acceptance and utilization of emerging applications and systems; changes in management policy, organizational culture, and cost control structures [4]. To understand, address, and resolve behavior concerns, multi-disciplinarily knowledge and exploration in economics, sociology, and psychology need to be applied.

The Sociotechnical system approach validates the four critical components of the Information system interdependency and proves that none of them works in isolation. They all interact, are mutually dependent, and consequently are subject to “*systemic effects*,” defined as any change in one component affecting all other components of the system. Bob Napier, former CIO of HP, was credited in 2003 with the quote: “Every business decision triggers an IT event.” Certainly the quote was valid eight years ago; it can be argued that today it is even more important. The two occurrences should not be separated: when addressing business issues like productivity, service quality, cost control, risk management, and ROI the decision-makers have to consider the appropriate corresponding modifications in the IT domain.

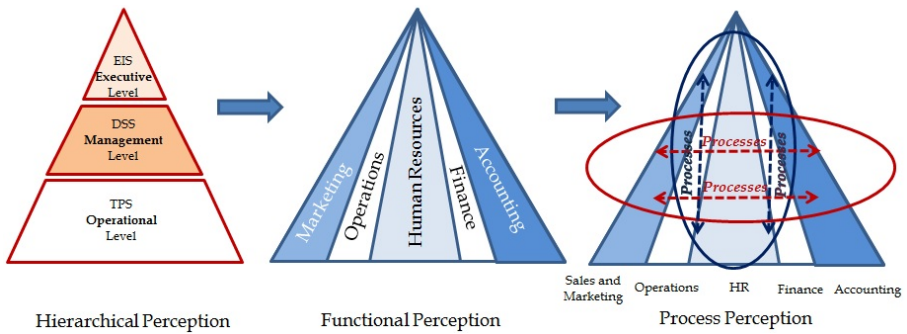
The process of changes and reciprocal adjustment of both technical and social subsystems should continue to interplay and growing closer until a mutually satisfying results are reached [4]. However, the model in reality could not be with equal subsystems’ changes. It should evolve from micro to macro level to reflect crucial influences of the external environment, including regulatory requirements, social and business trends, competitive pressures, interoperability with partnering institutions, especially when we analyze the role of the IT domain.

## 2.2 Organizational Information Systems

In his classic book, *Images of Organization*, Gareth Morgan outlines diverse perspectives of organizational processes and paradoxes: “Organizations are information systems. They are communications systems. And they are decision-making systems. If one thinks about it, every aspect of organizational functioning depends on information processing of one form or another [5].” As the saying goes “organizations are information processing systems,” and as such we need to examine thoroughly the limitations in processing capacity and in functional integration from the two key perspectives of the organizational model: *hierarchical* and *functional*.

The *hierarchical* perception defines three primary levels in an organization where specific to each level of activity and decision making event takes place [3]. At the *operational level* on the bottom of the organization’s model short-term, highly structured activities are performed and the objective is an efficient transaction

processed under a limited degree of uncertainty. The IT, commonly known as transaction processing systems, is utilized to structure and automate recurring operations assuring speed, accuracy, and precision in their execution.



**Fig. 2.** Hierarchical, functional and process perceptions

At the *managerial level* the main concerns and decision making activities are semi-structured and related to functional areas. The middle management, as a key factor at this level, executes and controls the processes based on adopted patterns and proven models. The IT systems supporting this operational model are mostly known as Decision Support Systems (DSS). They provide information and resources founded on internal operations and data analysis gleaned from the organization's *processing systems* to functional managers for tactical planning and mid-term decision-making.

The *executive level* handles all strategic planning and ad hoc circumstances, prioritizing long-term and wide-range decisions. Senior management must focus on industry trends, the competitive environment, and current organizational standing. The IT systems supporting this level known as executive information systems (EIS) are capable to collect, analyze, and synthesize organizational and external trend data.

The hierarchical perspective evolves, and following recent trends toward flattening organizational hierarchy moves to decentralization within organization creating business entities based on distinct functional areas such as Sales, Marketing, Production and Services, Finance and Accounting, Human Resources. The formed *functional perspective* alters the organization's information processing needs and a new variety of information systems, unique and homogeneous within a functional area such as CRM, SCM, HRS emerges. However, the technology support to local and enterprise-wide information processing and inter-functional areas sharing was insufficient.

Both hierarchical and functional perspectives suffer from a lack of integration among diversified systems, significant redundancy of data and resources, and operational limitations both technical and decision-making. A *process perspective* was prompted along with a *business process reengineering* (BPR) approach that employs a process view of organization's activities – Figure 2. The BPR utilizes top-down methodology to achieve internal business integration, activities rationalization, and duplications elimination across functional areas and managerial levels. Despite efficiency and cost saving achieved when employing BPR, as methodology it has

experienced difficulties and criticism as a result of neglecting the people factor, and focusing only on a single dimension of optimization and improvements.

### 2.3 IT Transformations and Competitive Advantage

To provide maximum benefits and strategic advances with novel networked technologies, information systems must be developed with a holistic understanding of the organization, its tactical goals, the competitive forces in related industries and in the surrounding environment. Ultimately, Michael Porter's five forces of the competitive position model is recently the most widely adopted approach (Figure 3). The model comprehensively exposes not only a general view of the organization with its traditional direct competitors, but also the reliance with four other forces within its market environment: new market entrants, supplier power, substitute products and technology development, and customer power [6].

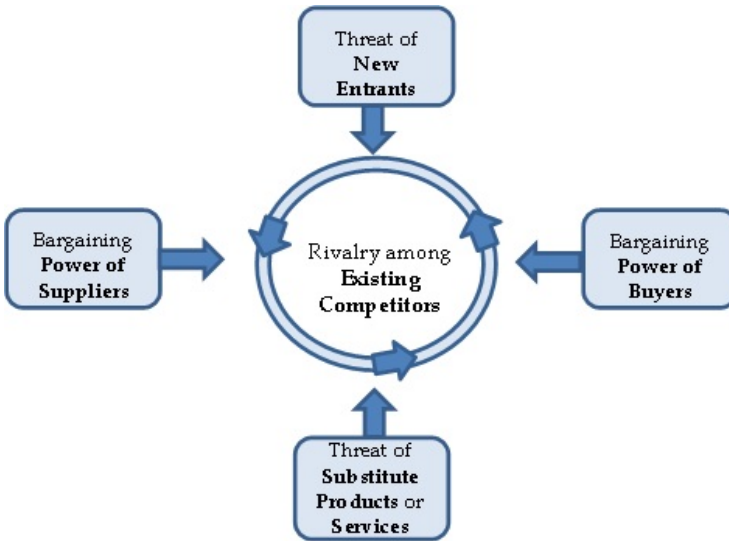


Fig. 3. Michael Porter's competitive forces model

While all of the five forces are critically important when strategic planning and managerial decisions are taken, the most associated force with emerging technologies is *substitute products* and *technology development*. The fast evolving technologies create new substitute products and services all the time. The impact of this force is not only local in reducing the company's price control and the profit margins in general, but also it is magnified because of the "networked businesses," speedy global market exposure, and consequent legislative effects.

A brilliant illustration of Porter's competitive forces model and how emerging technologies and innovative ideas could transform large business organizations is the IBM's "Business On Demand" approach launched by the CEO Sam Palmisano in mid 2000s. Employing emerging technologies still in their *hype* such as grid,

virtualization, utility and later cloud computing, the most innovative IT company aggressively expanded their products, services and operations across industries and globally, and made them available upon customers' demand, market opportunity or external threat. Strictly focus on the power and capabilities of emerging technologies, IBM sold its traditional personal computer business to Lenovo in 2005, and shifted sharply into higher-margin businesses, increasing recently its earnings per share fourfold. Continuing with this strategic transformation, IBM is building a government-funded private cloud in the Wuxi Industrial park in China. The China Cloud Computing Center in Wuxi is based on IBM's "Blue Cloud" technologies comprised of IBM CloudBurst - PaaS, IBM Tivoli Service Automation Manager – IaaS, a full range of SaaS including CRM and eCommerce solutions, open source software with capabilities to deliver Web 2.0 applications such as mashups, open collaboration, social networks, and mobile commerce. IBM has a significant research, development and business presence in China and a potential expansion of this cloud initiative to 100 more Chinese cities looks even more promising and favorable [7].

The success of IBM's business transformations and strategic cloud utilization harmonized to Porter's competitive forces model in all principal perceptions. In addition, these business cases validate that to execute a successful business strategy for rapid, right-size deployment of advanced transformations, a comprehensive analysis and realignment of capabilities within the following principal areas must be performed [2]:

- technology and infrastructure
- processes and products
- people and partners
- organizational culture, leadership and governance.

The process of repositioning the IT capabilities with listed principal areas, precisely harmonized with the sociotechnical system components, will result in enabling the IT alignment with organization's business strategy. The approach protects and maximizes the value of IT investments, and facilitates the next phase of the process – designing adventurous enterprise architecture.

## 2.4 Aligning IT with Enterprise Architecture

The process of enterprise architecture design requires a holistic view of the organization. Following such approach makes possible to explore how business processes, information flow, systems, technology and predominantly business priorities and strategies interact and contribute value to the organization.

Hence, understanding the organizational synergy in detail provides the means to define two important choices related to organizational business operations:

- How *standardized* its business should be across operational units?
- How *integrated* its business processes should be across those units?

Any organization operates in one of the four possible operating models, based on the business process selection as illustrated by Jeanne Ross from MIT Center for Information Systems Research. Which one is considered as "the right one" depends on the organization executives' strategic decision [8]:

- In the *diversification model* - low standardization and low integration - organizations operate in a decentralized mode with independent transactions, unique units with few data standards across local autonomies, most IT decisions are made within the units;
- The *coordination model* - low standardization, high integration - is used by organizations that deliver customized services and solutions by unique business units, while accumulating and providing access to integrated data across the divisions. The IT decisions should be made in consensus for designing IT infrastructure and integrated services, while IT applications decisions are processed within individual units;
- Organizations implementing the *replication model* - high standardization, low integration - typically provide high operational autonomy to their business units while requiring highly structured and standardized business processes. All decisions related to IT infrastructure, applications and services are centrally mandated;
- Organizations operating in the *unification model* - high standardization, high integration - are centralized managed with highly integrated and standardized business processes. The Enterprise IT is highly centralized and all decisions are made centrally.

For the IT domain it is most important to define and establish the organization's IT architecture underneath the enterprise architecture. A well-formulated IT architecture typically consists of *content* and *processes* and describes the technology components of the enterprise IT architecture:

- Technology strategy, governance and decisions aligned to business processes
- Information and data flow architecture
- Functional systems and applications architecture, including correlated interfaces
- Existing technology: technical infrastructure, platforms, services, and adopted standards.

An organization's imperative benefits come with a symbiosis and advancement of IT and the enterprise architecture defined by Ross as: "the organizing logic for business processes and IT infrastructure, reflecting the integration and standardization requirements of the firm's operating model." The MIT Center for IS Research describes the process of IT and enterprise architecture evolution in terms of four maturity stages [9]:

- Stage 1- *Business Silos*: complex and expensive localized IT solutions that respond to instant business needs, helping in local and functional optimization
- Stage 2- *Standardized Technology*: disciplined processes in IT service delivery and investment prioritization, achieving IT functional efficiency as to low OpEx and high reliability
- Stage 3- *Optimized Core*: defined enterprise priorities, investing in core packaged or customized integrated platforms and systems, accomplishing high operational efficiency



- Stage 4— *Business Modularity*: synchronized strategic and operational decisions with clear rules, reliable data, and business intelligence, gaining the most of IT-enabled capabilities towards assets utilization and business growth.

Evolving through the four stages is a challenging and advancing experience for organizations and their IT. As it has been said at the beginning - the radical changes impact the organizational mindsets in rethinking and reengineering their traditional IT resources. Indicators for such transformations can be seen analyzing the IT budget patterns from the MIT Center for Information Systems Research survey comparing 2007 and 2010 IT budget spending across the four stages [10].

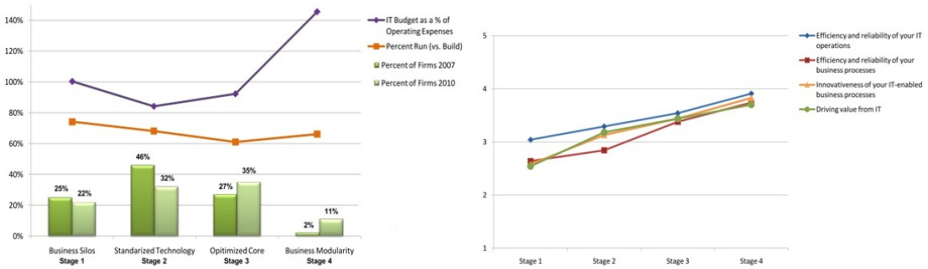


Fig. 4. Business capabilities relative to stages of architecture maturity [9]

An evident variation from the first three stages is at stage four where a substantial increase in IT budget occurs. While in stages 1 to 3 cutting IT costs is a priority and investments are mostly for eliminating inefficiencies by optimizations, in stage four organizations invest in increasing reliability and efficiency of the IT operations, seeking advances innovativeness in IT-enabled business processes and opportunities for driving value from IT (Figure 4). The decision makers at this stage are exploring and are ready to take the advantage of emerging technologies and delivery models. As one CIO is cited “we stopped thinking of the IT as a bad, and started thinking of it as what keeps the business running.”

Global business strategies place a new emphasis on growth					
Business strategies	Ranking of business strategies CIOs selected as one of their top 3 in 2011 and projected for 2014				
Ranking	2011	2010	2009	2008	2014
Increasing enterprise growth	1	*	*	*	1
Attracting and retaining new customers	2	5	4	2	3
Reducing enterprise costs	3	2	2	5	6
Creating new products or services (innovation)	4	6	8	3	4
Improving business processes	5	1	1	1	13
Implementing and updating business applications	6	*	*	*	12
Improving the technical infrastructure	7	*	*	*	7
Improving enterprise efficiency	8	*	*	*	10
Improving operations	9	*	*	*	2
Improving business continuity, risk and security	10	*	*	*	23
Expanding into new markets and geographies	11	13	10	4	5
Attracting and retaining the workforce	12	4	3	6	8
Introducing and improving business channels	15	15	*	*	9

Fig. 5. The CIOs reimagine IT [11]

The latest Gardner Executive Report “The 2011 CIO Agenda” surveying over 2000 CIOs across industries worldwide affirms the aforementioned trend of radical transformations in IT to support the growth and competitive advantage [11]. The trend comes together with sustained tight IT budgets as it is evident from the diagram, which leads to the next trend in IT reengineering - reducing costs and creating new products and services- that would happen through a process of “creative destruction.”

The figures show that these strategic alterations are with second highest priority in the executives’ 2011 and projected 2014 agendas – Figure 5. In the same survey close to half of all CIOs are confident in shifting their operations to applications, platforms, and infrastructures using cloud technologies.

### 3 Emerging Technologies Implications

As the demand of faster, efficient, and powerful computing structures intensifies, the need for more capable and dynamic computing environments increases. IT systems evolve over time. Each new generation of technology can, if utilized properly, perform faster and improve productivity at lower cost and higher quality. The economic and social motivation for specialized and consolidated IT models steadily intensifies. Any delays in understanding and adapting novelties might lead to dramatic changes or collapses. One of the many examples from the first decade of the century is with the retail book industry - it was slow in adapting to the E-commerce opportunities and got “Amazoned.”

#### 3.1 The Evolution of Emerging Dynamic Structures

Today’s cloud computing is based on foundational concepts that addressed an early need to best leverage computing resources over 40 years ago, taking the concepts of commodity *grid* computing and *virtualization* further by allowing self-services, metered usage and more automated dynamic resource and workload management practices.

*Grid* computing specifically refers to leveraging massive numbers of computers in parallel to solve particular problems, or to run specific applications. Grid computing is the underlying technology for a utility type computing model. Afterwards, with *virtualization* of systems, servers and applications, the model has expanded to a virtual platform without a specific underlying infrastructure. Virtualization is an abstract layer that allows multiple virtual machines, with heterogeneous operating systems to execute separately side-by-side on a same physical system. Virtualized services allow customers to utilize and expand their systems in many directions such as: server consolidation, storage and network virtualization, disaster recovery and business continuity protection, streamline testing and training, and secure enterprise desktops.

A splendid example of successful massive virtualization deployment is shared in the IDC Analyze the Future report from March 2011. Emerson, a diversified \$21 billion technology company, consolidated its IT infrastructure and deployed an inclusive virtualization model and efficient de-duplication backups. Based on the virtualization approach, Emerson now employs a utility chargeback model with its

internal divisions. Costs are assigned as services are consumed or equipment is used. The user can control, plan and manage their IT budgets more effectively. Consolidating all IT resources within two high-tech global datacenters, Emerson gains agility and cost efficiency, aligning the IT capabilities with its core business model [12].

Grid computing is now heading towards a convergence of utility pricing model, Web service integration, and virtualized technologies to enable a variety of cloud computing platforms for delivery and deployment.

### 3.2 The Cloud Hype

To design and deliver a future IT architecture that captures the promises and the benefits of Cloud Computing, the five core characteristics defined by National Institute for Standards and Technology must be considered:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured Service

Besides the five core features, there are favorable advances which focus on further strategies for adopting cloud-based services [13]:

- Lower cost: consolidated cloud resources operate at higher efficiencies and with greater utilization, resulting in significant cost reduction. Though cloud vendors charge premium for their services, the customers would save money by selecting the most needed options. Additionally, cloud computing deployment lets someone else manage the cloud infrastructure, while the organization will focus on managing their core activities, achieving considerable reductions in IT staffing costs
- Ease of utilizations: depending upon the type of services there would be no or minimal hardware and software requirements, upfront costs or adoption time
- Quality of Services: the higher cloud QoS compares to on-premises IT, can be obtained under the contract or SLA from the cloud vendor
- Reliability: The scale of cloud resources and their ability to provide load balancing and failover makes them highly reliable, often much more consistent than IT service in a single organization
- Simplified maintenance and upgrade: Naturally for centralized systems all patches and upgrades are easily performed and the users always have access to the latest versions timely
- Low Barrier to Entry: In particular, as upfront CapEx are dramatically reduced despite the institutional size, anyone in or because of cloud computing can gain the most of it at any time.

The largest cloud category to date and an anticipated leader in the next decade is Software as a Service. Customers use software applications, hosted by a cloud

provider and available over the Internet. According to Forrester Report from 2011, the SaaS revenue will reach in 2011 \$21.2 billion from total of \$25.5 billion from the public cloud. As a result of a strong demand from companies and organizations, Forrester predicts SaaS revenues to elevate up to \$92.8 billion by 2016, which would be 26% of the total software market [14].

Platform as a Service is the third largest cloud delivery model with a market size of \$820 million in 2011, with a predicted growth from 2012 on. PaaS is the middleware of the cloud, and customers use infrastructure and programming tools hosted by the service provider to develop their own applications. The customer does not manage or control the underlying cloud infrastructure, but has control over the deployed applications and possibly application hosting environment configurations.

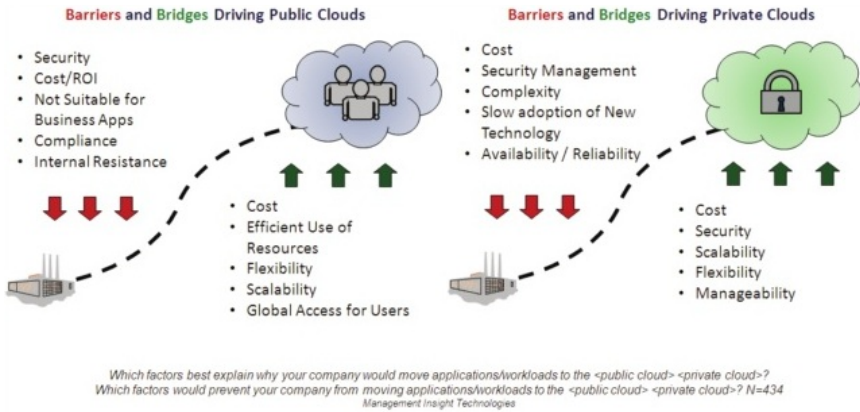
The second largest cloud category, with a \$2.9 billion market size in 2011 is Infrastructure as a Service. The IaaS provides computing power, storage, archiving, and other fundamental computing resources to an organization with a utility pricing and delivery model. The consumer does not manage or control the underlying cloud infrastructure, but does have control over operating systems, storage, deployed applications, and possibly select networking components.

For many organizations the primary question is not related to the delivery models, but directed to the purpose of the cloud and the nature of how the cloud is located: in other words - the deployment model. According to the NIST, the four cloud deployment models are: *Private* cloud, *Community* cloud; *Public* cloud; and *Hybrid* cloud.

IT professionals in organizations with a well-established and up-to-date data centers are inclined more towards private cloud solutions than to public or other varieties of off-premises cloud services. In Indiana University for example, with their approximately 1300 virtual servers and dynamic storage capabilities the private cloud architecture is a powerful and efficient data strategy [15]. By deploying internal cloud services on the newly refresh servers at their two data centers, without critical disruption of the institutional sociotechnical system, the university achieved long-term savings, immediate high performance and energy gains, substantial savings related to licensing and staff recruitment and training.

When exploring the main domains of the cloud services model, organizations need to consider not only the capabilities and the economic efficiency of the scale-oriented delivery, but the organization's *comfort with* and possible *resistance to* the idea of third party service provisioning. Cloud initiatives offer promise in the minds of executives and IT professionals alike, but the new delivery models take time to gain acceptance, and there are often technological and organizational inhibitors to adoption reflecting specifics in the socio-technical organization context.

According to a world-wide study comprised over 400 professionals published by Management Insight Technology, economic factors, and to a lesser extent, better services and increased productivity, are leading executive interests in the cloud [16]. At the same time, IT professionals, IT decision makers, and implementers are enthusiastic to work in the latest high-tech environment where better technology will lead to greater efficiency, improved services and they will be gaining current skills set (Figure 6).



**Fig. 6.** Private and public cloud implications [16]

Most executives and financial managers however, advocate for public clouds as initially more cost-efficient, flexible, and scalable with a global access solution. Strategizing the enterprise cloud solutions for many organizations might be most appropriate to build a specific hybrid model incorporating internal private cloud for the most sensitive data and critical applications and to off-premises - public or community clouds – for some less decisive systems and routine applications. Evolving the internal IT infrastructure toward a cloud-like model for supporting the most sensitive business operations at lower risk and higher security, will enhance and facilitate the integration with outside applications running into external clouds. In many organizations such an approach would decrease the risk of disruptive technology and process changes, and they will better control their information assets.

### 3.3 Emerging Dynamic Structures Impacts

To summarize the impact of emerging technologies on sociotechnical systems the following significant points should be considered:

- *People* factor - the resistance to changes
- *Process* change – from operational efficiency – to business agility and growth
- *Structure* rationalization – changing IT from a cost center to advance assets
- Strategic choice towards distinctive *digitalization*

*The first finding* reflects a common reason for high-tech failures is not the failure of the technology, but the resistance to change individual routines, and to invest additional time and efforts that may not be compensated. This phenomenon is visible from the latest survey by Management Insight Technologies – where *risk of job lost*, *IT loss of control*, *lack of skills*, and *perceived benefits*, are the critical IT staff concerns [16] – Figure 7.

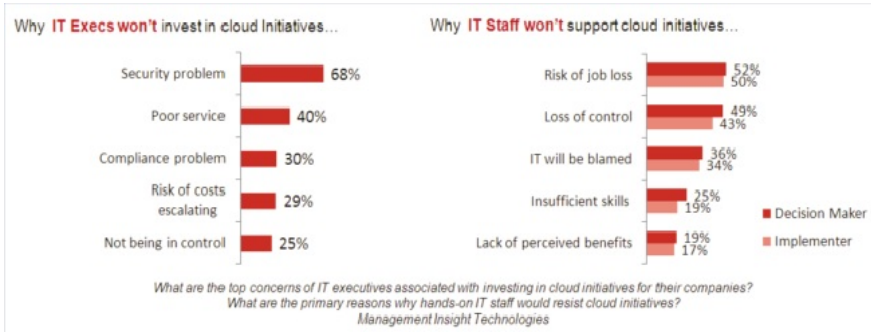


Fig. 7. People considerations to changes [16]

In conjunction with the latest economic difficulties and the reductions of the operational budgets the options for staff benefits are diminished – Figure 8. The two statistics only reinforce the importance of the people factor when novel technologies are considered.

**Median Annual Growth of IT Operational Budgets: 2006-2011**

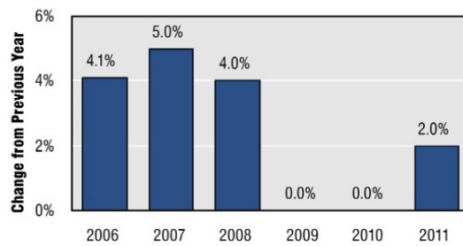


Fig. 8. Annual operational budget trend [17]

**IT Operational Budget per User: 2006-2011**

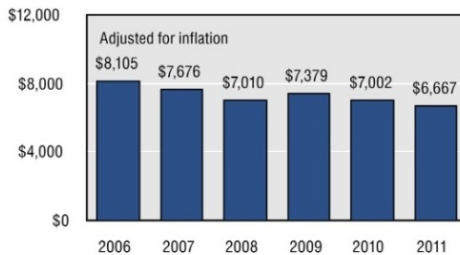


Fig. 9. IT spending per user [17]

The second finding suggests process transformations not only towards operational and service efficiency, but to support business mobility and growth with emerging technologies and new delivery models. The Computer Economy survey confirms that

since 2006 to 2011, there was a steady decline of IT spending per user, and the pressure to do more with less IT costs per user, is a motive to not only address the operational efficiency, but to explore advanced technologies and new cost saving models – Figure 9.

The third finding reinforced *Structure* rationalization to transform IT from a cost center to advanced assets and business intelligence capabilities. The Gartner research on Future Directions of the IT industry from 2011, exemplifies several emerging IT-enabled initiatives capable to increase the revenue if appropriate structural changes are employed. The IT Money Spending Model at Figure 10 depicts where the transformations are the most appropriate – from lessening the *After the Sale* IT spending to substantial increase the investments in *Before the Sale* and *The Sale* sectors. The process will require significant alternations in the enterprise structure to support emerging technologies and initiatives such as: content-aware computing, social and cloud computing, and information-enabled pattern- based strategy.



Fig. 10. IT money spending model (last 50 years) [18]

**Change from Prior Year in Median IT Operational Budget, by Sector**

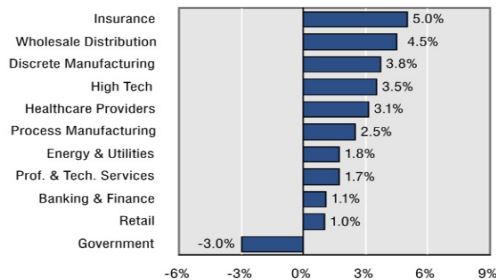


Fig. 11. Public vs. Private IT spending [17]

The fourth finding is that in a digital economy every organization needs to adopt a strategic choice towards distinctive *Digitalization*. This will maximize the contribution IT makes to core performance areas from the executives’ perspective:

business growth, business agility, and asset utilization. According to latest survey of Center for IS Research from March 2011, “firms scoring in the top quartile on digitalization impact had a 5% higher return on equity than the average firm, and 15% higher than the bottom quartile in their industry.” This fourth finding should concern more the public sector as Computer Economics Analysis illustrates at Figure 11 government organizations are still cutting their IT spending levels with 3%, while the private sector indicates 2% growth, and some industries such as Insurance, Wholesales, Manufacturing, Hi-Tech and Healthcare are between 3% to 5% growth.

There are many other organizational considerations and challenging socio-technical implications of emerging technologies in difficult economic times: asymmetric competition, speed of innovations, speed to the market, speed of institutional model’s evolution. All these specifics require more effort and analysis on how to integrate technology into the sociotechnical system to stretch the core business activities and to excel on IT-enabled capabilities. We may need to take a fresh look at Peter Drucker’s pivotal statement from 1980: *“In turbulent times, an enterprise has to be able to withstand sudden blows and avail itself of unexpected opportunities. This means that, in turbulent times, the fundamentals must be managed and managed well.”*

## References

1. Ivanov, I.: Emerging Utility and Cloud Computing Models. In: I-WEST 2009. INSTICC Press, Portugal (2009)
2. Applegate, L., Austin, R., Soule, D.: Corporate Information Strategy and Management: Text and Cases. McGraw-Hill/Irwin, Inc., New York (2009)
3. Picolli, G.: Information Systems for Managers: Text and Cases. John Wiley and Sons, Inc., Hoboken (2008)
4. Laudon, K., Laudon, J.: Management Information Systems: Managing the Digital Firm. Pearson Education, Inc., Upper Saddle River (2011)
5. Morgan, G.: Images of Organization. Sage Publications, Inc., Thousand Oaks (2006)
6. Brown, C., Dehayes, D., Hoffer, J., Martin, E., Perkins, W.: Managing Information Technology. Pearson Education, Inc., Publishing as Prentice Hall, Upper Saddle River (2011)
7. IBM Case Study: Wuxi builds engine for economic growth with cloud computing solution from IBM. IBM Systems and Technology Group, NY (October 2009)
8. Ross, J.: Forget Strategy: Focus IT on Your Operating Model, vol. V(3C). Center for Information Systems Research, Sloan School of Management, MIT, Research Briefing (2005)
9. Ross, J., Beath, C.: Maturity Still Matters: Why A Digitized Platform is Essential to Business Success, vol. XI(II). Center for Information Systems Research, Sloan School of Management, MIT, Research Briefing (2011)
10. Quaadgras, A., Weill, P., Ross, J.: The MIT CISR Value Framework: Commitments and the Practices that Support Them, vol. XI(III). Center for Information Systems Research, Sloan School of Management, MIT, Research Briefing (2011)
11. McDonald, M., Aron, D.: Gartner Executive Programs Reimaging IT: The 2011 CIO Agenda. Gartner, Inc., Stamford (2011)



12. DuBois, L.: Case Study: Emerson Consolidates Its IT Infrastructure and Deploys Virtualization and Deduplication for Efficiency and Cost Savings. IDC Analyze the Future, Framingham (2011); ID Number: 226383
13. Sosinsky, B.: Cloud Computing Bible. John Wiley and Sons, Inc., Hoboken (2010)
14. O'Neill, S.: Forrester: Public Cloud Growth to Surge, Especially SaaS. CIO Magazine (April 26, 2011), <http://www.cio.com/article/print/680673>
15. Biddick, M.: The Why and How of Private Clouds. InformationWeek (June 05, 2010), [http://www.informationweek.com/news/hardware/data\\_centers/225300316](http://www.informationweek.com/news/hardware/data_centers/225300316)
16. Black, L., Mandelbaum, J., Grover, I., Marvi, Y.: The Arrival of "Cloud Thinking". White Paper, Management Insight Technologies, U.S.A. (2010)
17. Computer Economics: IT Spending & Staffing Benchmarks. Executive Summary 2011/2012, 2082 Business Center Drive, Suite 240, Irvine, CA 92612, U.S.A. (2011)
18. McGee, K.: The 2011 Gartner Scenario: Current States and Future Directions of the IT Industry. Gartner, Inc., Stamford (2011); ID Number G00209949

# On-the-Fly Dependable Mediation between Heterogeneous Networked Systems

Antonia Bertolino<sup>1</sup>, Antonello Calabrò<sup>1</sup>, Felicita Di Giandomenico<sup>1</sup>, Nicola Nostro<sup>1</sup>, Paola Inverardi<sup>2</sup>, and Romina Spalazzese<sup>2</sup>

<sup>1</sup> Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, Pisa, Italy

<sup>2</sup> Department of Computer Science, University of L'Aquila, Coppito (AQ), Italy  
firstname.lastname@isti.cnr.it,  
{paola.inverardi,romina.spalazzese}@univaq.it

**Abstract.** The development of next generation Future Internet systems must be capable to address complexity, heterogeneity, interdependency and, especially, evolution of loosely connected networked systems. The European project CONNECT addresses the challenging and ambitious topic of ensuring eternally functioning distributed and heterogeneous systems through on-the-fly synthesis of the CONNECTORS through which they communicate. In this paper we focus on the CONNECT enablers that dynamically derive such connectors ensuring the required non-functional requirements via a framework to analyse and assess dependability and performance properties. We illustrate the adaptive approach under development integrating synthesis of CONNECTORS, stochastic model-based analysis performed at design time and run-time monitoring. The proposed framework is illustrated on a case study.

## 1 Introduction

We live in the Future Internet (FI) era, which is characterized by unprecedented levels of connectivity and evolution. Software systems are increasingly pervasive, dynamic and heterogeneous, and many -even critical- aspects of modern society rely on their continuous availability and seamless interoperability. Ensuring the successful dynamic composition among heterogeneous, independently developed Networked Systems (NSs) raises the need of novel computing paradigms, such as the revolutionary approach to *on-the-fly connection* pursued within the European FP7 Future and Emerging Technology Project CONNECT.

CONNECT follows the ambitious goal of enabling *seamless* and *dependable* interoperability among NSs in spite of technology diversity and evolution. The key idea is to compose systems by generating on-the-fly the interoperability solution necessary to assure the connection among the heterogeneous NSs both at application and at middleware level. The synthesized solution is called a CONNECTOR or also *mediating connector* or *mediator* for short; the system obtained from the composition of the NSs through the CONNECTOR is said the CONNECTED System.

Automatically synthesized CONNECTORS are concrete emergent entities that mediate the NSs discrepancies, i.e., they translate and coordinate mismatching interaction protocols, actions, and/or data models, allowing applications to interact effectively.

A synthesized CONNECTOR, if it exists, provides by construction a correct solution to functional interoperability among the NSs.

This is however not sufficient: effective interoperability also requires that such on-the-fly CONNECTED systems provide the required non-functional properties and continue to do so even in presence of evolution. The CONNECTORS are not a priori guaranteed to provide the desired non-functional properties for the CONNECTED system, thus a suitable and adaptive assessment framework is required. In this paper, we focus on the problem of ensuring the non-functional properties for CONNECTED systems.

Concerning dependability and performance properties, several challenges arise. Off-line, or pre-deployment, assessment can help to take appropriate design decisions by providing a priori feedback about how the system is expected to operate. Nevertheless, the unavoidable high chance of inaccurate/unknown model parameters might result in inadequate analysis results. Moreover, the many possible variations occurring during the system lifetime would require to foresee and analyze all the possible scenarios which could take place at run-time (e.g., to be stored in a look-up table from which to retrieve the correct analysis upon a scenario's occurrence). On the other hand, resorting to processing the measurements collected in real operation at a later stage, e.g. in periodic reviews, may be inadequate, since by the time the observations are processed the operational environment may have changed. Furthermore, in the CONNECT context the above problems are exacerbated because, as said, components are dynamically assembled to satisfy an emergent user goal. In this scenario the only part of the system under control is the synthesized CONNECTOR, whereas for the NSs only declarative or learned on-the-fly knowledge can be assumed.

To contribute to overcome the above issues, we have developed an approach which tries to combine the benefits of both pre-deployment and processing of data obtained from real executions. The proposed assessment framework combines stochastic model-based analysis [1] with continuous on-line assessment of non-functional properties through a lightweight flexible monitoring infrastructure, and applies such approach to the on-the-fly CONNECT system into a continuous loop.

In the following we initially provide the context for our approach by introducing the CONNECT architecture (Section 2). Then we introduce the case study that is used to demonstrate the applicability of the integrated analysis framework (Section 3). Mediator synthesis (Section 4), pre-deployment analysis (Section 5) and the run-time monitor (Section 6) are briefly presented, and hence their synergic usage (Section 7), through which adaptive assessment is pursued. Finally we overview related work (Section 8) and draw conclusions (Section 9).

## 2 The CONNECT Project

Our research is carried out in the context of the FP7 “ICT forever yours” European Project CONNECT<sup>1</sup>, belonging to the Future and Emerging Technologies track. As said in the introduction, the ambition of the project is to have eternally functioning systems within a dynamically evolving context.

---

<sup>1</sup> <http://connect-forever.eu>

In Figure 1 we provide an overview of the CONNECT vision and architecture. In brief, the NSs manifest the intention to connect to other NSs. The *Enablers* are networked entities that incorporate all the intelligence and logic offered by CONNECT for enabling the required connection. We show in schematic form the enablers which are currently part of the CONNECT enabling architecture:

**Discovery Enabler:** discovers the NSs, catches their requests for communication and initiates the CONNECT process. We tend to make the minimum possible assumptions on the information (called the *affordance*) that NSs must provide;

**Learning Enabler:** we use active learning algorithms to dynamically determine the interaction behaviour of a NS and produce a model in the form of a labeled transition system (LTS);

**Synthesis Enabler:** from the models of the two NSs, this enabler synthesizes a mediator component through automated behavioural matching. More details on this enabler are given in Section 4;

**Deployment Enabler:** deploys and manages the synthesized CONNECTORS;

**Monitor Enabler:** collects raw information about the CONNECTORS behaviour, filters and passes them to the enablers who requested them. The CONNECT monitoring infrastructure is further described in Section 6;

**DEPER Enabler:** this is the enabler assessing dependability and performance properties and is described in detail in Section 5;

**Security and Trust Enabler:** collaborates with the synthesis enabler to satisfy possible security and trust requirements. It also continuously determines if the requirements are maintained at run-time, by receiving monitoring data from the monitoring enabler. For reasons of space, we do not deal with this enabler in this paper.

All communication among the enablers and with the CONNECTORS happens through a message bus, which is currently implemented by a simple message-based communication model (as for instance the Java Messaging Service (JMS)).

In this paper we provide a snapshot of the functioning of CONNECT over the case study introduced in the following section. For space limitation, we focus on the interaction among Synthesis, Dependability&Performance and Monitor, which are highlighted by tick borders in Figure 1. We show first how a dependable CONNECTOR is deployed (pre-deployment analysis), and then how, via the feedback obtained through run-time monitoring of the CONNECTOR behaviour, CONNECTOR adaptation is triggered and managed. In particular, we devise a process for the CONNECTOR creation that is supported by powerful infrastructures made available by CONNECT itself. Once the Discovery Enabler discovers new devices, the CONNECT supporting infrastructure starts the computation of a CONNECTOR on the fly –if possible. Then, when the intent to communicate is manifested, the CONNECTOR -if it exists- is partially computed and is hence concretized. Note that the CONNECTOR could not exist because NSs are not compatible and then do not have a way to communicate.

### 3 Case Study

In this section, we present our running example for presenting how synthesis, analysis and monitoring work in integrated way.

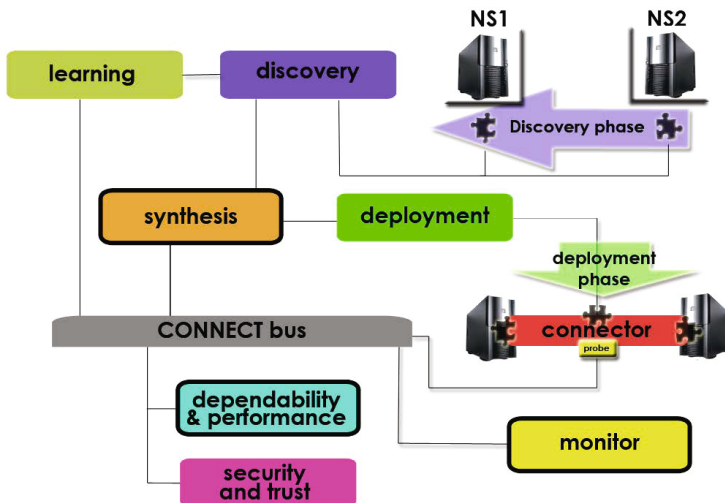


Fig. 1. The CONNECT architecture

### 3.1 Terrorist Alert Scenario

We consider the CONNECT Terrorist Alert scenario [2], depicting the critical situation that during a show in the stadium, the stadium control center spots one suspect terrorist moving around. This emergency situation makes it necessary to exchange information between policemen and security guards patrolling the surroundings of the stadium equipped with heterogeneous applications.

Each policeman can exchange confidential data with other policemen with a *Secured-FileSharing* application. Security guards, on the other hand, exchange information by using another application, denominated *EmergencyCall*. The two applications have the same aim (i.e., enable information exchange), but use different protocols as we describe in the following.

#### SecuredFileSharing

- The peer that initiates the communication denominated *coordinator* (the Policemen of our example) sends a broadcast message (*selectArea*) to selected peers operating in a specified area of interest (the Police control center of our example).
- The selected peers reply with an *areaSelected* message.
- The coordinator sends an *uploadData* message to transmit confidential data to the selected peers.
- Each selected peer automatically notifies the coordinator with an *uploadSuccess* message when the data have been successfully received or the coordinator can receive an exception.

An example of message flow between a coordinator, i.e., a Policeman, and a Police control center is depicted in Figure 2(a) while the application behaviour of another Policeman is shown in Figure 3. It is worth to notice that, for readability, in the figure

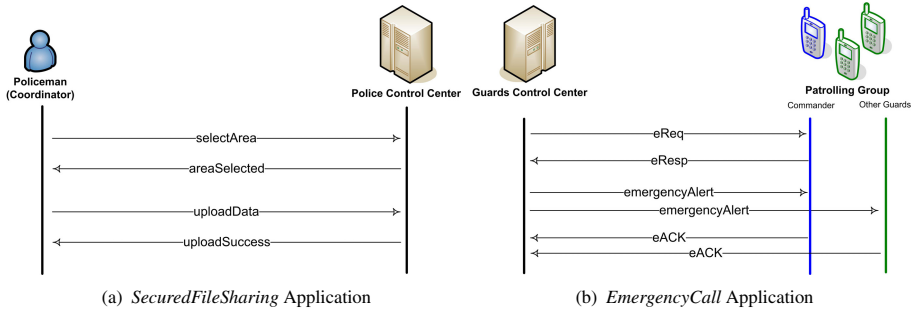


Fig. 2. Sequence Diagrams of the applications

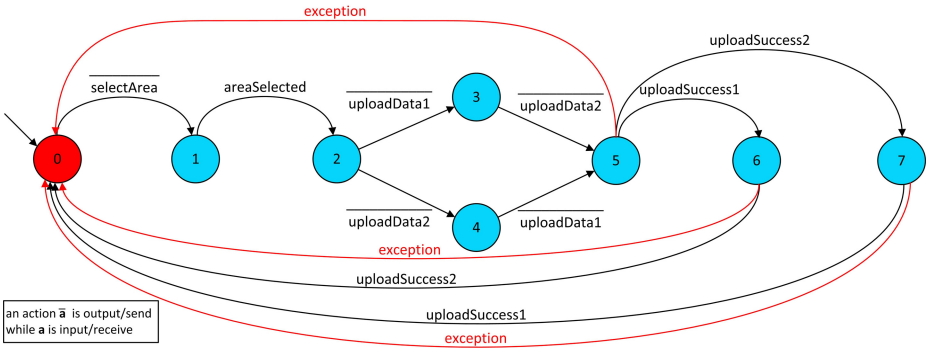


Fig. 3. LTS of the *Policeman* application

the LTS of the *Policeman* describes the sending of the broadcast alert to two selected peers while in the experiments we conducted, we used 11 selected peers.

The affordance of the *Policeman* application includes also the following *non-functional requirement*: she/he should receive the 65% of acknowledgements for the alerts sent within 30 time units, otherwise a failure is reported.

### EmergencyCall

- The Guards Control Center sends an `eReq` message to the Commanders of the Patrolling Groups operating in a given area of interest.
- The Commanders reply with an `eResp` message.
- The Guards Control Center sends an `emergencyAlert` message to all Guards of the patrolling groups; the message reports the alert details.
- Each Guard’s device automatically notifies the Guards Control Center with an `eACK` message when the data has been successfully received.

The message flow among the Guard control Center, the Commander and the Other Guards is depicted in Figure 2(b). Figure 4(a) shows the LTS of the Commander, and the LTS of the Other Guards is shown in Figure 4(b).

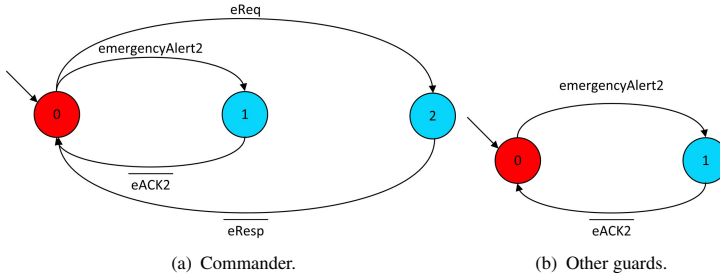


Fig. 4. LTSs of the *EmergencyCall* application

### 3.2 CONNECT in the Case Study

With reference to CONNECT architecture (see Figure II), the two NSs which need to communicate but are not a priori compatible, are the devices implementing the described applications *SecuredFileSharing* and *EmergencyCall*. Hence, to allow a Policeman and the Guards, operating in the zone where the suspect terrorist has escaped, to communicate, CONNECT proposes to automatically synthesize on-the-fly a CONNECTOR that can mediate between the two different communication protocols. Such CONNECTOR should be able to support the exchange of information, while also fulfilling possible non-functional requirements.

## 4 Automated Mediator Synthesis

Our focus is on the interoperability between heterogeneous protocols. By *interoperability* we mean the ability of protocols to *correctly communicate and coordinate* i.e., to *correctly synchronize*. In other words, two systems successfully interoperate if they correctly exchange *compatible conversations* or *compatible traces*. By *heterogeneous protocols* we mean that, although in principle they could interact since they have compatible (i.e., complementary) functionalities, protocols can be characterized by discrepancies that may undermine their ability to seamlessly interoperate (i.e., communicate and coordinate). Discrepancies include incompatible interaction protocols and/or different interfaces meaning different actions and/or data models. Examples of heterogeneous application protocols are the Policeman and Commander of the Patrolling Group of the case study.

In order to enable interoperability among heterogeneous protocols, we devised a theory for the automated synthesis of CONNECTORS [3, 4], also called mediating connectors or mediators for short. Figure 5 provides an overview of our methodology.

Our approach takes as input the descriptions of two NSs and in particular their behavioral protocols, described as Labeled Transition Systems (LTSs), together with their ontological information conceptualizing their actions through an application domain ontology. By referring to the case study, the synthesis takes as input the LTS of the Policeman, the LTS of a Commander of the Patrolling Group and of its Guards and their ontologies and follows a process made up by three phases or steps as described in the following.

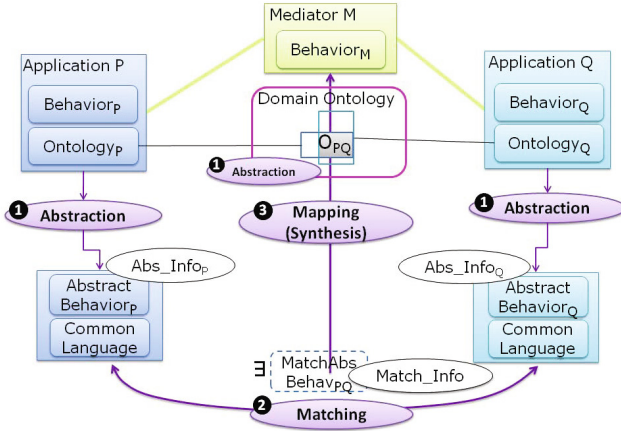


Fig. 5. An overview of the mediator synthesis approach

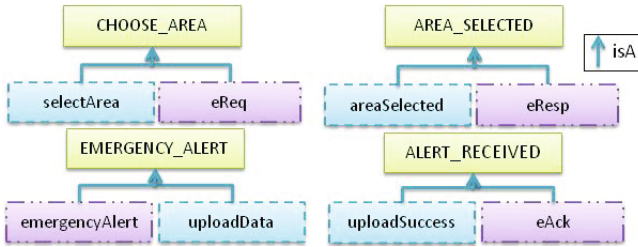
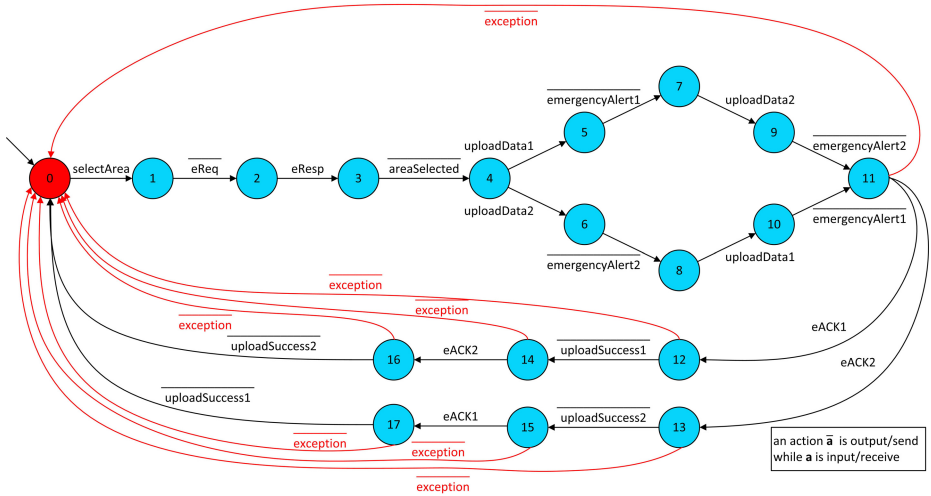


Fig. 6. Domain Ontology (upper case elements) where the ontologies of both protocols have been mapped (Dots and dashes boxes - Ontology of the Guards Control Center; dashed boxes - Ontology of the Policeman)

1. *Abstraction* that makes models comparable and, if possible, reduces their size making it easier and faster to reason on them. Reasoning on ontologies, a common language for both protocols is identified on the domain ontology and used to re-label them. The common language of our case study is illustrated by the elements with upper case names in Figure 6.
2. *Matching* that checks the NSs compatibility identifying possible mismatches. Compatible protocols can potentially interoperate despite they show some differences. That is, communication and coordination between such protocols is possible in principle since they are semantically equivalent and complementary, but cannot be achieved seamlessly because of *heterogeneity: mismatches* and/or *third parties conversations*. Examples of mismatches are: protocol languages have (i) different granularity, or (ii) different alphabets; protocols behavior have different sequences of actions with data (i.e., traces) because of (a.1) the order in which actions and data are performed by a protocol is different from the order in which the other protocol performs the complementary actions with data. Protocols behavior may have different sequences of actions also because of (a.2) interleaved actions related to *third parties conversations* i.e., with other systems, the environment. In some cases, as





**Fig. 7.** Synthesised CONNECTOR for the case study

for example (i), (ii) and (a.1), it is necessary to properly perform a manipulation of the two languages. In the case (a.2) it is necessary to abstract the third parties conversations that are not relevant to the communication. Referring to the case study, the heterogeneities we identify are what we call signature mismatches [5, 6], i.e., the two protocols use different names for semantically equivalent concepts.

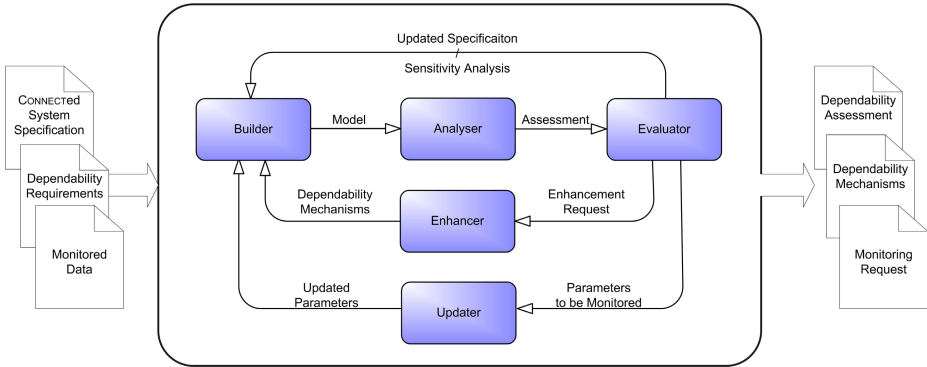
Synchronization between protocols, thus, can be achieved under mediation i.e., through a mediator that while managing such mismatches, allows protocols to effectively exchange compatible traces (sequences of actions).

3. *Mapping* or *Synthesis* that produces a mediator that mediates the found mismatches so enabling the NSs to communicate. Figure 7 illustrates the synthesized CONNECTOR between the Policeman and the Commander of the Patrolling Group applications of our case study.

- The `selectArea` message of the Policeman is translated into an `eReq` message directed to the Commander of the Patrolling Group operating in the area of interest.
- The `eResp` message of the Commander is translated into an `areaSelected` message for the Policeman.
- The `uploadData` message of the Policeman is translated into a multicast `emergencyAlert` message to all the Guards (Commander included).
- Each `eACK` message automatically sent by the Guards' devices that correctly receive the `emergencyAlert` message are translated into a `uploadSuccess` message for the Policeman.

Note that still for figure readability, the illustrated CONNECTOR is between one Policeman, a Commander of the Patrolling Group and one Guard.

A **mediator** is then a protocol that allows communication and coordination among compatible protocols by mediating their differences. It serves as the locus where



**Fig. 8.** Architecture of the Dependability&Performance (DEPER) Enabler

*semantically equivalent and complementary actions* are correctly synchronized thus enabling (a mediated) interoperability among protocols.

In summary, by reasoning about the protocols discrepancies, our theory automatically identifies and synthesizes an *emerging mediator* that solves them thus allowing protocols to interoperate. Automatically synthesized CONNECTORS are concrete emergent entities that translate and coordinate mismatching interaction protocols, actions, and/or data models, letting applications interact effectively

## 5 Pre-deployment Analysis to Support CONNECTOR Synthesis

Pre-deployment assessment is a crucial activity to drive the system design towards a realization compliant with the required quality levels. In fact, it allows for early detection of design deficiencies, so as to promptly take the appropriate recovery actions, thus significantly saving in money and time with respect to discovering such problems at later stages. As briefly outlined in the Introduction, the pre-deployment assessment in CONNECT is performed by the Dependability and Performance enabler, introduced in Figure 1 and shortly referred to as DEPER, which exploits State-Based Stochastic modelling and analysis, embedded in an automated process. Figure 8 illustrates the architecture of DEPER that, together with the prototype implementation based on the Möbius evaluation framework, is documented in [7, 8].

Very briefly, the activities of modules Builder, Analyser and Evaluator, triggered in sequence, perform the cycle of the pre-deployment analysis: from the specification of the CONNECTED system (both functional and non-functional) and of the metrics to be analysed, to checking whether the analysis results match with the metrics level, as requested by the NSs.

Evaluator informs Synthesis about the outcome of the check. In case of mismatch, it may receive back a request to evaluate if enhancements can be applied to improve the dependability or performance level of the CONNECTED system, thus calling the intervention of the Enhancer module. In the other case, the CONNECTOR's design is considered satisfactory and ready to be deployed, thus terminating the pre-deployment

analysis phase. However, because of possible inaccuracy of model parameters due to potential sources of uncertainty dictated by the dynamic and evolving context, Evaluator also instructs the Updater module about the events to be observed on-line by the Monitor enabler.

The attempts to improve dependability and performance of the CONNECTOR consist in extending the model with a dependability mechanism, selected from a library of already defined ones (see [9]), until either a successful mechanism is found, or all the mechanisms are exhausted.

The Updater module provides adaptation of the off-line analysis performed at pre-deployment time to cope with changes in, or inaccurate estimates of, model parameters, through interactions with the Monitor enabler (e.g., because of limited knowledge of the NSs characteristics acquired by Learning/Discovery enablers). It receives from Monitor a continuous flow of data for the parameters under monitoring relative to the different executions of the CONNECTOR. Accumulated data are processed through statistical inference techniques. If, for a given parameter, the statistical inference indicates a discrepancy between the on-line observed behaviour and the off-line estimated value used in the model resulting into a significant deviation of the performed analysis, a new analysis is triggered.

## 5.1 Pre-deployment Analysis in the Terrorist Alert Scenario

Taking as a reference the above described scenario, we focus in the following on the basic interactions between Synthesis and DEPER enablers, performed to exchange requests for dependability analysis of a pre-deployed CONNECTOR.

The interaction starts when Synthesis sends a JMS message to DEPER. The message contains the specification of the LTSs of the CONNECTOR and of the NSs involved.

Figure 9 depicts the dependability and performance model of the synthesized CONNECTOR built by DEPER at design time, using the SAN formalism [10]. The model is obtained through automatic transformation from the LTS specification of the NSs, that is the *SecuredFileSharing* and *EmergencyCall* in the considered scenario. The measure assessed in the evaluation is the *coverage*. Coverage represents a dependability indicator and is given by the percentage of responses the control center receives back from the guards within a certain time  $T$ .

Once the message has been received and the SAN models have been built, DEPER starts the coverage analysis through the Möbius tool [11]. The performed analysis is shown in Figure 12 of Section 7 and allows to state that the CONNECTOR fully satisfies the coverage requirement with a Timeout greater than 2 time units. Hence, a response is sent, through a JMS message, from DEPER to Synthesis to communicate that the CONNECTOR can be dependably deployed.

After the deployment of the CONNECTOR a sensitivity analysis from DEPER on the impact of model parameters on the assessment of the selected measure revealed that critical parameters to keep under observation on-line via the Monitor enabler, for the coverage measure, are occurrences of transitions  $eACK$ . Refining the pre-analysis knowledge on the values assumed for such parameters by real observations constitutes a fundamental step in enhancing the accuracy of the analysis results. In fact, should the initial forecast for these parameters deviate from what is evidenced through repeated executions,

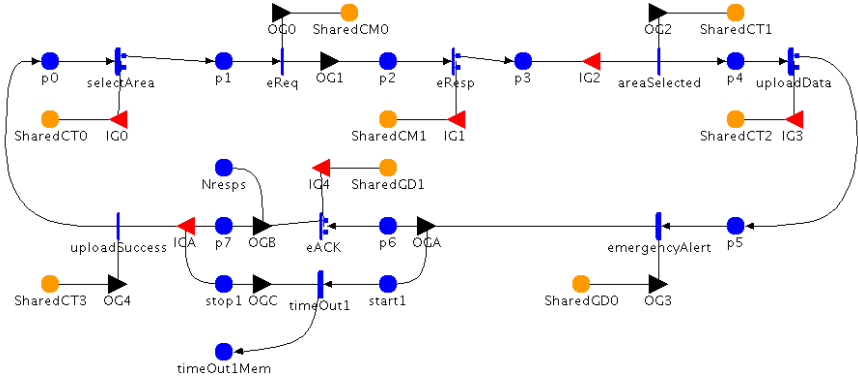


Fig. 9. SAN model of the CONNECTOR

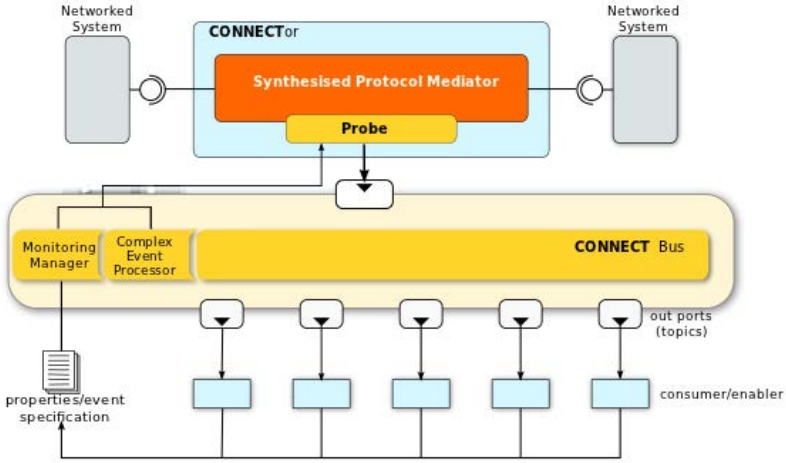


Fig. 10. GLIMPSE architecture

a new analysis round needs to be triggered to understand whether the dependability and performance requirements are still met by the CONNECTED system.

## 6 Events Observation through Monitoring

Timely and effective run-time adaptation can only be ensured by continuously observing the interactions among the NSs. To this purpose, in CONNECT we have developed a modular, flexible and lightweight monitoring infrastructure, called GLIMPSE<sup>2</sup>. GLIMPSE infrastructure, shown in Figure 10, is totally generic and can be easily applied to different contexts. To provide a better communication decoupling, we adopted a publish-subscribe communication paradigm.

<sup>2</sup> GLIMPSE is an acronym for Generic fLexible Monitoring based on a Publish-Subscribe infrastructure.

The lowest level of the monitoring is represented by the probe deployed into the CONNECTOR; this probe monitors the messages exchanged among the NSs involved into the communication, possibly applying a local filter in order to decrease the amount of messages sent on the CONNECT bus. Note that such probes are non intrusive data collectors (proxies), i.e., they have no effect on the order and timing of events in the application and do not generate overhead on the communication or on the interacting services.

The second layer of the monitoring infrastructure is represented by the information consumers, the entities interested to obtain the evaluation of a non-functional property or interested to receive notification of occurrences of events/exceptions that may occurs into the CONNECTOR.

The gathered information is provided in form of events. An event is an atomic description, a smaller part of a larger and more complex process at application level. In CONNECT, an event represents a method invocation on a remote web service: the invocation, coming from the producer to the consumer, is captured when it comes through the CONNECTOR, encapsulated into a specific object, and sent through the CONNECT bus. A Complex Event Processor (CEP) analyzes the atomic events to infer complex events matching the consumer requests, by a rule engine. In the current GLIMPSE implementation, we adopt the Drools Fusion rule language [12] that is open source and can be fully embedded in the realized Java architecture.

Finally, the Manager module is in charge to manage all the communication between the consumers and the CEP.

## 7 Continuous Run-Time Adaptation

After having performed the pre-deployment analysis phase and the deployment of the CONNECTOR, we focus here on its adaptive assessment via the interaction among Synthesis, DEPER and the Monitor enabler. Basically, the dynamicity and evolution of the targeted environment lead to potential sources of uncertainty, which undermine the accuracy of the off-line analysis. To cope with this issue, run-time monitoring is exploited to re-calibrate and enhance the dependability and performance prediction along time. As already pointed out in Section 5, the continuous run-time adaptation of the pre-deployment performed analysis is in charge to the Updater module.

DEPER and Monitor interact by using a Publish/Subscribe protocol. The interaction starts when DEPER sends a JMS message whose payload contains an XML object rule generated using ComplexEventRule classes [7].

Once the CONNECTOR is deployed, data (events) derived from real executions are sent by the probe to the CONNECT bus. The Monitor enabler gathers those events and using the CEP component, tries to infer one or more of the patterns to which the DEPER enabler is subscribed.

Upon occurrence of a relevant event the DEPER enabler is notified: the latter, in turn, performs a statistical analysis of the monitored observations and uses such information to check the accuracy of the model analysed before deployment. If the model parameters are found to be inaccurate, DEPER updates the model with the new values,

and performs a new analysis. If the new analysis evidences that the deployed CONNECTOR needs adjustments, a new synthesis-analysis cycle starts.

As an example, we consider the steps to refine the accuracy of the *failure probability* of the communication channel between the *EmergencyCall* application and the CONNECTOR. We accumulated data generated from several executions of the CONNECTOR, in scenario's configurations with a fixed number of 11 guards, which allowed to refine the value of the failure probability from 0.02, assumed during pre-deployment dependability analysis, to 0.1.

Upon updating the parameter, a new dependability analysis is performed in order to verify if the updated CONNECTOR still satisfies the requirement. Unfortunately, the analysis shows that coverage measure does not meet the requirement. The CONNECTOR needs to be enhanced. Based on the library of dependability mechanism [9], the *retry mechanism* has been automatically selected and implemented in the already developed CONNECTOR model, in order to enhance the CONNECTED system and satisfy the requirement.

The retry mechanism consists in re-sending messages that get corrupted or lost during communications, e.g., due to transient failures of communication links. This mechanism is widely adopted in communication protocols, such as TCP/IP for enabling reliable communication over unreliable channels. A typical implementation of the retry mechanism uses time-outs and acknowledgements: after transmitting a message, the sender waits for a message of the receiver that acknowledges successful communication. If the acknowledgement is not received within a certain time interval, the sender assumes that the communication was not successful, and re-transmits the message.

The SAN model of the retry mechanism is shown in Figure 11. On the sender side, the mechanism creates a message re-transmission policy for re-sending the message at most  $N = 3$  times; on the receiver side, the mechanism creates a policy for avoiding duplicated reception of messages and for sending acknowledgements. The sender stops re-transmitting the message as soon as it gets an acknowledgement that the message has been successfully received, or after  $N$  attempts. A detailed description of the mechanism model can be found in [9].

Figure 12 shows the trend of the *coverage* (on the  $y$  axis) at increasing values of Timeout (on the  $x$  axis). Also, the threshold coverage line as specified in the requirement (set to the value 0.65) is reported. The figure includes three plots, corresponding to: (i) the results of the pre-deployment analysis; (ii) the results of the analysis after the parameters influencing coverage have been updated with actual values from the run-time observations; and (iii) the results of the analysis after both the parameters influencing coverage have been updated and a retry mechanism have been implemented to enhance the CONNECTED system. It is worth noting that coverage value obtained through the pre-deployment analysis is fully satisfying the requirement with timeout value greater than 3 (time units). While the coverage value after updating the failure probability parameter never satisfies the requirement, although the value of timeout increases, which means that the estimation of coverage at pre-deployment time was too optimistic. Finally, we note that the analysis performed considering the actual value of failure probability and including the enhanced mechanism provides results on coverage that satisfies the coverage requirement.

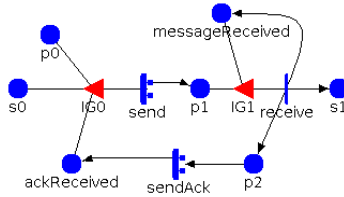


Fig. 11. Retry mechanism

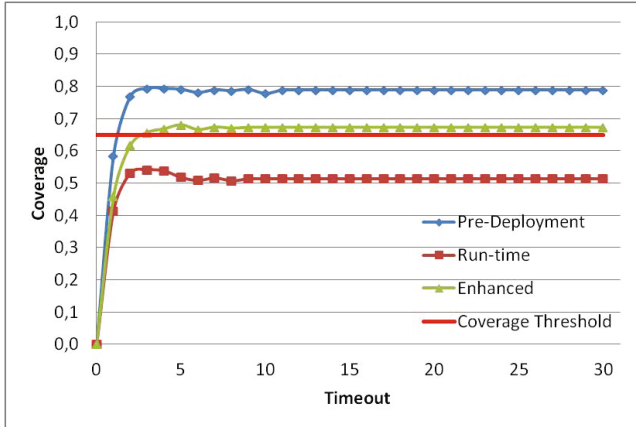


Fig. 12. Trend of Coverage as a function of Timeout

The CONNECTOR needs to be enhanced; therefore DEPER informs the Synthesis enabler about the analysis results and appropriate actions are taken by Synthesis (typically, a new CONNECTOR is synthesised).

## 8 Related Work

The automated synthesis of application-layer CONNECTORS relates to a wide number of works in the literature within different research areas.

The *Ubiquitous Computing* (UbiComp) proposed by Weiser [113] has a key principle that is to make the computer able to vanish in the background to increase their use making it in an efficient and invisible manner to users. Our CONNECTORS fit perfectly the ubiquitous vision where each NS maintains its own characteristics, being able to communicate and cooperate with the others without having any prior knowledge of them thanks to the support provided by the mediators that masks divergencies making them appear homogeneous.

*Interoperability* and *mediation* have been investigated in several contexts, among which protocol conversion [14–16], integration of heterogeneous data sources [17], software architecture [18], architectural patterns [19], design patterns [20], patterns of connectors [21, 22], Web services [23–27], and algebra to solve mismatches [28] to mention a few.

A lot of work has also been devoted to *connectors* like for example [29–33] to mention few. A work strictly related to the mediators is the seminal work by Yellin and Strom [34]. With respect to our synthesis approach, this work prevents to deal with ordering mismatches and different granularity of the languages (see [5, 6] for a detailed mismatches description). Other works related to our but posing the focus on different problems are [35] and [36].

Stochastic model-based approaches for quantitative analysis of non-functional properties have been largely developed along the last decades and documented in a huge literary production on this topic. The already cited papers [1, 37] provide a survey of the most popular ones. The choice of the most appropriate type of model to employ depends upon the complexity of the system under analysis, the specific aspects to be studied, the attributes to be evaluated, the accuracy required, and the resources available for the study. The prototype implementation of our DEPER enabler is based on Stochastic Activity Networks (SANs) [10], a variant of the Stochastic Petri Nets class.

With regard to monitoring, various approaches have been recently proposed. Similarly to GLIMPSE, also [38] presents an extended event-based middleware with complex event processing capabilities on distributed systems, adopting a publish/subscribe infrastructure, but it is mainly focused on the definition of a complex-event specification language. The aim of GLIMPSE is to give a more general and flexible monitoring infrastructure for achieving a better interpretability with many possible heterogeneous systems. Another monitoring architecture for distributed systems management is presented in [39]. Differently from GLIMPSE, this architecture employs a hierarchical and layered event filtering approach. The goal of the authors is to improve monitoring scalability and performance for large-scale distributed systems, minimizing the monitoring intrusiveness.

A prominent part of our framework is in the combined usage of pre-deployment model-based analysis and run-time observations via monitoring. Preliminary studies that attempt combining off-line with on-line analysis have already appeared in the literature. A major area on which such approaches have been based is that of autonomic computing. Among such studies, in [40], an approach is proposed for autonomic systems, which combines analytic availability models and monitoring. The analytic model provides the behavioural abstraction of components/subsystems and of their interconnections and dependencies, while statistical inference is applied on the data from real time monitoring of those components and subsystems, to assess parameter values of the system availability model. In [41], an approach is proposed to carry out run-time reliability estimation, based on a preliminary modelling phase followed by a refinement phase, where real operational data are used to overcome potential errors due to model simplifications. Our approach aims at proposing a general and powerful evaluation framework, tailored to a variety of dependability and performance metrics, to meet a wide spectrum of system requirements and adaptation needs.

## 9 Conclusions

We have introduced the ambitious vision of the CONNECT project for an eternally and dependably CONNECTed world. Of the complex CONNECT architecture under development, we have focused here on the Synthesis enabler, which derives on-the-fly a



mediator enabling the functional interoperation among heterogeneous NSs; the Dependability&Performance enabler, which applies Stochastic model-based analysis for assessing the desired non-functional properties; and the Monitor, which observes the run-time CONNECTOR behaviour. We have discussed on a case study their integrated usage to allow for adaptive analysis accounting for possible inaccurate information or potential evolution of the involved NSs. We refer to a library of adaptation patterns that DEPER suggests to Synthesis to enhance the CONNECTOR and make it compliant with the expected non-functional properties. At present, Synthesis uses such suggestion to synthesize a new CONNECTOR that can satisfy the non-functional requirements. In future, we will investigate approaches for on-the-fly adaptation of the CONNECTOR, where possible.

For reasons of space, we could not cover other important enablers in the CONNECT architecture. Further information can be obtained from the project web site.

**Acknowledgements.** This work has been partially supported by the European Project CONNECT Grant Agreement No.231167.

## References

1. Bondavalli, A., Chiaradonna, S., Giandomenico, F.D.: Model-based evaluation as a support to the design of dependable systems. In: Diab, H.B., Zomaya, A.Y. (eds.) *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, pp. 57–86. Wiley (2005)
2. CONNECT Consortium: Deliverable 6.1 – Experiment scenarios, prototypes and report – Iteration 1 (2011)
3. Inverardi, P., Issarny, V., Spalazzese, R.: A Theory of Mediators for Eternal Connectors. In: Margaria, T., Steffen, B. (eds.) *ISoLA 2010, Part II*. LNCS, vol. 6416, pp. 236–250. Springer, Heidelberg (2010)
4. Spalazzese, R., Inverardi, P., Issarny, V.: Towards a formalization of mediating connectors for on the fly interoperability. In: *Proceedings of the WICSA/ECSA 2009*, pp. 345–348 (2009)
5. Spalazzese, R., Inverardi, P.: Mediating Connector Patterns for Components Interoperability. In: Babar, M.A., Gorton, I. (eds.) *ECSA 2010*. LNCS, vol. 6285, pp. 335–343. Springer, Heidelberg (2010)
6. Spalazzese, R., Inverardi, P.: Components interoperability through mediating connector pattern. In: *WCSI 2010*, arXiv:1010.2337. *EPTCS*, vol. 37, pp. 27–41 (2010)
7. Bertolino, A., Calabró, A., Di Giandomenico, F., Nostro, N.: Dependability and Performance Assessment of Dynamic CONNECTed Systems. In: Bernardo, M., Issarny, V. (eds.) *SFM 2011*. LNCS, vol. 6659, pp. 350–392. Springer, Heidelberg (2011)
8. Masci, P., Martinucci, M., Di Giandomenico, F.: Towards automated dependability analysis of dynamically connected systems. In: *Proc. IEEE International Symposium on Autonomous Decentralized Systems*, Kobe, Japan, pp. 139–146. IEEE (June 2011)
9. Masci, P., Nostro, N., Di Giandomenico, F.: On Enabling Dependability Assurance in Heterogeneous Networks through Automated Model-Based Analysis. In: Troubitsyna, E.A. (ed.) *SERENE 2011*. LNCS, vol. 6968, pp. 78–92. Springer, Heidelberg (2011)
10. Sanders, W.H., Malhis, L.M.: Dependability evaluation using composed SAN-based reward models. *Journal of Parallel and Distributed Computing* 15, 238–254 (1992)
11. Daly, D., Deavours, D.D., Doyle, J.M., Webster, P.G., Sanders, W.H.: Möbius: An Extensible Tool for Performance and Dependability Modeling. In: Haverkort, B.R., Bohnenkamp, H.C., Smith, C.U. (eds.) *TOOLS 2000*. LNCS, vol. 1786, pp. 332–336. Springer, Heidelberg (2000)

12. Drools fusion: Complex event processor, <http://www.jboss.org/drools/drools-fusion.html>
13. Weiser, M.: Hot Topics: Ubiquitous Computing. *IEEE Computer* (1993)
14. Calvert, K.L., Lam, S.S.: Formal methods for protocol conversion. *IEEE Journal on Selected Areas in Communications* 8, 127–142 (1990)
15. Lam, S.S.: Correction to "protocol conversion". *IEEE Trans. Software Eng.* 14, 1376 (1988)
16. Okumura, K.: A formal protocol conversion method. In: *SIGCOMM*, pp. 30–37 (1986)
17. Wiederhold, G.: Mediators in the architecture of future information systems. *IEEE Computer* 25, 38–49 (1992)
18. Garlan, D., Shaw, M.: An introduction to software architecture. Technical Report CMU-CS-94-166, Carnegie Mellon University (1994)
19. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture. A System of Patterns*, vol. 1. Wiley, Chichester (1996)
20. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Resusable Object-Oriented Software*. Addison-Wesley Professional (1995)
21. Wermelinger, M., Fiadeiro, J.L.: Connectors for mobile programs. *IEEE Trans. Softw. Eng.* 24, 331–341 (1998)
22. Spitznagel, B.: *Compositional Transformation of Software Connectors*. PhD thesis, Carnegie Mellon University (2004)
23. Motahari Nezhad, H.R., Xu, G.Y., Benatallah, B.: Protocol-aware matching of web service interfaces for adapter development. In: *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*, pp. 731–740. ACM, New York (2010)
24. Cimpian, E., Mocan, A.: WSMX Process Mediation Based on Choreographies. In: *Bus-sler, C.J., Haller, A. (eds.) BPM 2005. LNCS*, vol. 3812, pp. 130–143. Springer, Heidelberg (2006)
25. Vaculín, R., Sycara, K.: Towards automatic mediation of OWL-S process models. In: *IEEE International Conference on Web Services*, pp. 1032–1039 (2007)
26. Williams, S.K., Battle, S.A., Cuadrado, J.E.: Protocol Mediation for Adaptation in Semantic Web Services. In: *Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS*, vol. 4011, pp. 635–649. Springer, Heidelberg (2006)
27. Cavallaro, L., Di Nitto, E., Pradella, M.: An Automatic Approach to Enable Replacement of Conversational Services. In: *Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS*, vol. 5900, pp. 159–174. Springer, Heidelberg (2009)
28. Dumas, M., Spork, M., Wang, K.: Adapt or Perish: Algebra and Visual Notation for Service Interface Adaptation. In: *Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS*, vol. 4102, pp. 65–80. Springer, Heidelberg (2006)
29. Spitznagel, B., Garlan, D.: A compositional formalization of connector wrappers. In: *ICSE*, pp. 374–384 (2003)
30. Fiadeiro, J.L., Lopes, A., Wermelinger, M.: Theory and practice of software architectures. Tutorial at the 16th IEEE Conference on Automated Software Engineering, San Diego, CA, USA, November 26-29 (2001)
31. Lopes, A., Wermelinger, M., Fiadeiro, J.L.: Higher-order architectural connectors. *ACM Trans. Softw. Eng. Methodol.* 12, 64–104 (2003)
32. Barbosa, M.A., Barbosa, L.S.: Specifying Software Connectors. In: *Liu, Z., Araki, K. (eds.) ICTAC 2004. LNCS*, vol. 3407, pp. 52–67. Springer, Heidelberg (2005)
33. Bruni, R., Lanese, I., Montanari, U.: A basic algebra of stateless connectors. *Theor. Comput. Sci.* 366, 98–120 (2006)
34. Yellin, D.M., Strom, R.E.: Protocol specifications and component adaptors. *ACM Trans. Program. Lang. Syst.* 19 (1997)
35. Tivoli, M., Inverardi, P.: Failure-free coordinators synthesis for component-based architectures. *Sci. Comput. Program.* 71, 181–212 (2008)

36. Canal, C., Poizat, P., Salaün, G.: Model-based adaptation of behavioral mismatching components. *IEEE Trans. Software Eng.* 34, 546–563 (2008)
37. Nicol, D.M., Sanders, W.H., Trivedi, K.S.: Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing* 1, 48–65 (2004)
38. Pietzuch, P., Shand, B., Bacon, J.: Composite event detection as a generic middleware extension. *IEEE Network* 18, 44–55 (2004)
39. Hussein, E.A.S., Abdel-wahab, H., Maly, K.: HiFi: A New Monitoring Architecture for Distributed Systems Management. In: *Proceedings of ICDCS*, pp. 171–178 (1999)
40. Mishra, K., Trivedi, K.S.: Model Based Approach for Autonomic Availability Management. In: Penkler, D., Reitenspiess, M., Tam, F. (eds.) *ISAS 2006. LNCS*, vol. 4328, pp. 1–16. Springer, Heidelberg (2006)
41. Pietrantuono, R., Russo, S., Trivedi, K.S.: Online monitoring of software system reliability. In: *Proc. EDCC 2010 - 2010 European Dependable Computing Conference*, pp. 209–218. IEEE Computer Society (2010)

# SADT/IDEF0 for Augmenting UML, Agile and Usability Engineering Methods

David A. Marca

The University of Phoenix, Online School College of Information Systems and Technology,  
3157 East Elwood Street, Phoenix, Arizona 85034, U.S.A.  
dmarca@email.phoenix.edu

**Abstract.** Many experts state that: a) specifying "all the small parts of a system" and b) correct expected system usage, can make Agile Software Development more effective. Unified Modeling Method (UML) addresses the former; Usability Engineering addresses the later. Taken together, they create a systems de-velopment framework, capable of: a) specifying functions, data, behavior and usage, b) rapid prototyping, and c) verifying system usability and correctness. All three of these methods focus first on the system, while secondarily trying to ascertain system context. Correct and complete context requires domain modeling. Structured Analysis and Design Technique (SADT/IDEF0) is a proven way to model any kind of domain. Its power and rigor come from: a) a synthesis of graphics, natural language, hierarchical decomposition, and relative context coding, b) distinguishing controls from transformations, c) function activation rules, and d) heuristics for managing model complexity. This paper explains how SADT/IDEF0 domain modeling can bring correct and complete context, to today's commonplace disciplines of the Unified Modeling Language (UML), Agile System Development, and Usability Engineering methods.

**Keywords:** Domain Modeling, General Systems Theory, UML, Agile Development, Usability Engineering, SADT, IDEF0, Domain Driven Design.

## 1 Introduction

Commercial software engineering disciplines have come a very long way since their post World War II origins. Three of the more commonplace disciplines of today are: a) Unified Modeling Language – UML [5], Agile Software Development [2], and Usability Engineering [29]. When used in combination, these methods have a strong track record for developing software for many kinds of problems and domains. Over the last ten years, a large amount of research has been done on the shortcomings of these methods, and a collection of this research is presented in this paper. Taken as a whole, this research suggests that many shortcomings arise because domain modeling is not at the core of these methods. Therefore, one way to bolster today's commonplace software development methods is to augment with a proven domain modeling method, such as SADT/IDEF0. Domain modeling is at the core of SADT/IDEF0, and when properly used, the method can produce holistic domain models that can address any level of complexity or abstraction. To explain:

### 1.1 Domain Modeling Is Not the Core of Current Methods

The first, and very important, aspect about modeling with today's commonplace methods is that UML [5], Agile Software Development [2], and Usability Engineering [29] have their origins rooted in software systems. In other words, their focus is on the software system. Thus, their principles, languages and practices were invented for creating software. While each includes a component for domain modeling, that component is not at the core of the method. For example: UML's core is software system specification, Agile's core is rapid software deployment, and UE's core is evaluation of the software system during its use. For these disciplines, domain modeling is just a first step to getting to the core work.

### 1.2 Domain Modeling Is at the Core of SADT/IDEF0

In contrast, SADT/IDEF0 is rooted in general systems theory [35]. Its focus is any kind of system. Interestingly, when it was first introduced, many in the commercial world confused it for being a method that could just describe either software systems or manufacturing processes. While it can describe these two kinds of systems, its strength is its focus on systems in general. Therefore, it has unique principles, simple language, and special practices for describing any real-world phenomenon – domain modeling is its core! When used correctly, SADT/IDEF0 can produce a set of very concise, small models, with tightly connected context and content. This paper will illuminate the often misunderstood potential of SADT/IDEF0 as a contributor to, and not a replacement for, today's software development methods (see Figure 1).

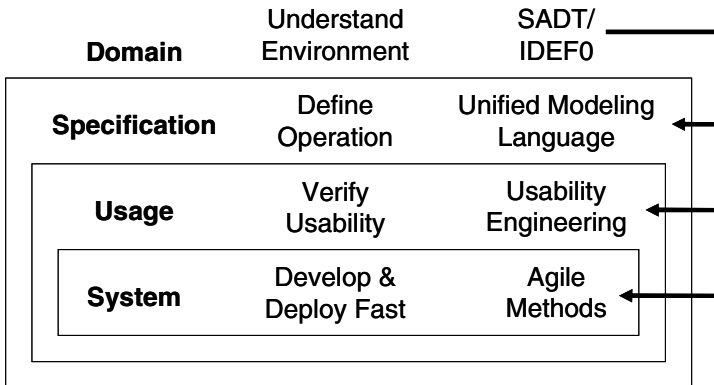


Fig. 1. Where SADT/IDEF0 augments UML, agile and usability engineering methods

### 1.3 The Use of SADT/IDEF0 Produces Holistic Domain Models

The distinguishing, unique aspect of SADT/IDEF0 is its ability to holistically describe an entire domain to any desired low level of detail, and to describe its context to any desired high level of abstraction. It is thus a story-telling discipline with very rigorous engineering syntax (i.e. boxes and arrows) and semantics (e.g. box sides for Input, Control, Output, and Mechanism – ICOM – and corresponding

implicit off-page connectors), plus heuristics for managing model complexity (e.g. hierarchic decomposition, 3 to 6 boxes per decomposition, single purpose and viewpoint per model, model “call” syntax). For example, it distinguishes controls from inputs, and advocates stopping model decomposition when the model’s purpose has been fulfilled. It is the aspect of simple, concise, complete, context-rich, holistic description that is the primary contribution of SADT/IDEF0 to the other aforementioned methods [25].

#### 1.4 SADT/IDEF0 Can Address any Level of Complexity or Abstraction

The statement often arises: “My method is effective at domain modeling, so I do not need another method.” My response: “yes and no,” and here is why: For very simple domains or systems, it is easy to “get one’s head around the problem.” Thus, no other method is needed to understand the system’s immediate context. However, for very complex problems (e.g. enterprise-wide solutions, large weapons such as a submarine, aircraft sheet metal fabrication), no single software developer can understand the whole problem. At the extreme case (e.g. well log interpretation, disaster recovery, decision making, strategy formulation), no domain expert may know or be able to articulate consistently and accurately, the entire domain. Such situations require a context map [52] to document an understanding of the domain that must then drive the solution design [54] [53]. SADT/IDEF0 has an extremely simple graphic language and a model creation technique that, from the same starting point of any particular subject, can describe: a) all details (i.e. decompose complexity), b) the context of that subject (i.e. context modeling).

## 2 Why Consider SADT/IDEF0?

Since the 1970’s, SADT/IDEF0 has been used to successfully describe a vast number and variety of domains. The reason for this success is best described by Doug Ross in his seminal paper [35]. To paraphrase: SADT/IDEF0 incorporates any other language; its scope is universal and unrestricted. It is concerned only with the orderly and well-structured decomposition of a subject. Model decomposition is sized to suit the modes of thinking and understanding of the viewpoint of the model and the intended audience of readers. Units of understanding (i.e. boxes and their data) are expressed in a way that rigorously, precisely and consistently represents domain interrelationships. Decomposition is carried out to the required degree of depth, breadth, and scope while still maintaining all of the above properties. Thus, SADT/IDEF0 increases the quantity and quality of understanding that can be beyond the limitations inherently imposed by other kinds of natural or formal languages. Some details:

### 2.1 Vast Experience in a Wide Variety of Domains

SADT/IDEF0 has over 35 years of domain modeling experience, across a vast number of problems involving systems ranging from tiny to huge, in a wide variety of industries [24]. It has been used in commerce, government and military around the world. It has been used by small or privately held companies (e.g. 1-2 person

start-ups) to some of the largest of largest organizations in the world (e.g. the U.S. Air Force), and on some of the largest initiatives (e.g. the U.S.A.F. Sheet Metal Fabrication Project [55]). This widespread success is due to its very strong set of domain modeling concepts, principles and features. A summary of these “features” is given in the Appendix of this paper, and is organized into two tables. Table 1 summarizes the box and arrow syntax and semantics, and the rules for how they interconnect. Table 2 completes Table 1 and summarizes the reference language used to identify model elements. None of the aforementioned methods can claim the array of features, the richness of graphic semantics, the number of in-context supplements, or the longevity of success across so many industries and problems *for domain modeling*.

## 2.2 Strong Conceptual Underpinnings for Modeling

Since the 1970s, experts have agreed that deep understanding of the domain is vital for successful and effective software engineering [44]. The conceptual underpinnings of SADT/IDEF0 continue to be cited as being very strong for domain modeling. Most notably: a) tightly managed multiple views at the architecture level for complex systems [19], b) support for aspect-oriented modeling by being able to modularize cross-cutting concerns [17], c) defining boundaries essential for specifying objects, system scope, human-computer interaction [40], d) hierarchical exposition of detail for very large domains without loss of context and without making errors when going to next levels of detail/abstraction [31], e) specifying strong versus weak influence that each datum has on its functions [40], and f) “calls” (i.e. just like a software subroutine call) a model from another model to maximize reuse [20].

## 2.3 SADT/IDEF0 Features Are for Domain Modeling

Back in the 1970s, we did not have the universally understood notion of “ontology” as we know it today in the software engineering field. Nonetheless, since 1977, SADT™ has had a complete ontology for domain modeling [38]. Back then, the components of the ontology were called “features.” Three of its core features are: context, model, and viewpoint [35]. With these three features, a core modeling principle was constructed: one model = one subject, described from one viewpoint [18]. This is, in effect, what we commonly call today the “system boundary.” This demarcation point allows SADT/IDEF0 to consider a domain to be the whole context within which a system operates (e.g. the enterprise for a financial system as well as the business environment around that enterprise, the submarine for a defensive weapon system, the building for a thermostat control system). Also, SADT/IDEF0 has a simple box-and-arrow graphic language with associated semantics that make it ideal for capturing domain knowledge and reviewing it with end-users [10].

## 2.4 Preservation of Context

Probably the most important aspects of a domain modeling method are: a) simple syntax within which domain-specific language can be embedded, b) powerful semantics for representing the various roles information play in a domain, c) rigorous decomposition rules to support the detailing of highly complex subjects as well as the

abstraction (to any level) of the context around any given system boundary, and d) consistent subject boundary management so that no context is lost when you move “into” the details of the subject. Together, these aspects provide a means by which context is always preserved. Figure 2 provides an example of context preservation. Notice how the data (arrow) inputs (I), controls (C), outputs (O), and mechanisms (M) that cross the functional boundary (box) are tied directly to the arrows on the diagram that details the function through the use of ICOM coding. With ICOM codes, you can never lose your way when you decompose a subject or create an abstraction that represents the context of a subject. Thus, since context preservation is crucial for domain modeling, SADT/IDEF0 has merit for augmenting the system development methods [25] such as the ones given earlier in this paper.

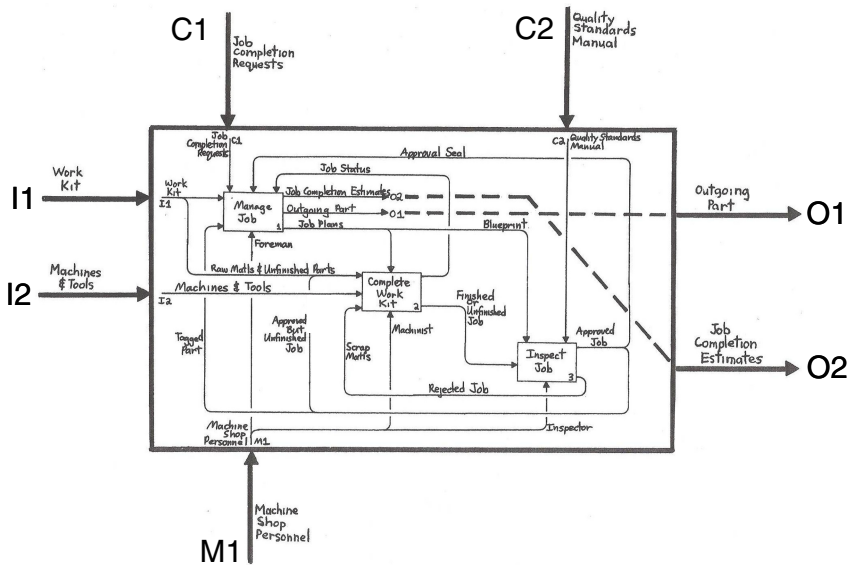


Fig. 2. An example of how SADT/IDEF0 models preserve context [24]

### 3 Augmentation Approach

The augmentation approach given in this paper is based on over 10 years of research by many practitioners and researchers. They point out particular shortcomings in the domain modeling portions of the aforementioned methods. They identified particular domain knowledge that, if it were available, could improve the results generated by the methods. So, given that particular domain knowledge required by UML, Agile and Usability Engineering methods. In short, correct, comprehensive and consistent specifications of domain knowledge are needed. Not only can SADT/IDEF0 correctly, comprehensively and consistently describe an entire domain – and not just the immediate context of a software system – it can describe that domain in rich and varied ways using carefully designed in-context supplements [38]. To explain:



### 3.1 Domain Knowledge Required by Other Methods

When practiced correctly, SADT/IDEF0 compresses a wealth of domain knowledge into a manageable set of small models. SADT/IDEF0 models and the special supplements created from the diagrams in those models, describe particular domain knowledge which the aforementioned methods depend upon: a) for UML: the system interface with its environment, decisions around manual versus automated function realization, functional scope, important objects in the domain, data dictionary, control data distinct from transactional data, overarching rules (often expressed as policies or doctrine), domain events and responses to those events (often called scenarios), and common versus special case scenarios; b) for Agile: same as for UML; and c) for Usability Engineering: the users' work, the context of that work, the tasks for accomplishing the work, the systems users need, and system usage scenarios.

### 3.2 Strong Specifications of Domain Knowledge

This paper briefly looks at some of the ontology of SADT (see Appendix), plus some additional features added after 1977, and explain how they can be used to augment the domain modeling portions of UML, Agile, and UE. Section 4 explains through figures and tables how SADT/IDEF0 models create stronger specifications of domain knowledge than the method it is augmenting. For example: a) knowledge that would have been missed by the other method, b) knowledge that would have been very hard to identify or describe by the other method, c) knowledge that needs to appear in all three methods that does not now do so, and d) how knowledge can be traced through all three methods. It is important to repeat that the domain modeling portions of the aforementioned methods are not bad; they just have shortcomings that over 10 years of practice have identified and documented (see References). SADT/IDEF0 can support the improvement recommendations in that documentation.

### 3.3 Knowledge Specification Using In-Context Supplements

The basic "unit of specification" of SADT/IDEF0 is the diagram, and a collection of diagrams comprises a model. However, SADT/IDEF0 has additional means by which domain knowledge is specified. To explain, the SADT/IDEF0 modeling process gives a person much more information than what is put on the basic diagram [24]. For example: a) terminology definitions, b) properties of functions and data, c) in-context narratives about the domain, d) particular situations (e.g. control flows, work flows) and special circumstances (e.g. mutually constraining functions) that occur in the domain, and e) rules by which functions activate and data must or must not interact with each other. Figure 3 gives an example of one basic diagram plus its supplemental pages, each identified with a letter corresponding to a-e above.

SADT/IDEF0 uses diagram supplements to capture this information, usually just after a basic diagram is approved by the domain experts that were interviewed by the systems analyst who authored the diagram. The supplements are: i) glossary page, ii) for exposition only (FEO) page, and iii) text page [38]. A glossary page defines terminology. A text page succinctly describes the operation of each box on the diagram. FEO pages contain closely related figures or pictures, or they annotate the

basic diagram with: property labels, highlighted boxes and arrows, or box activation rules. Each supplement is derived directly from only its basic diagram, and thus these specifications of domain knowledge are always inside the context of one, well-bounded subject. Thus, SADT/IDEF0 supplements are fully consistent with each other.

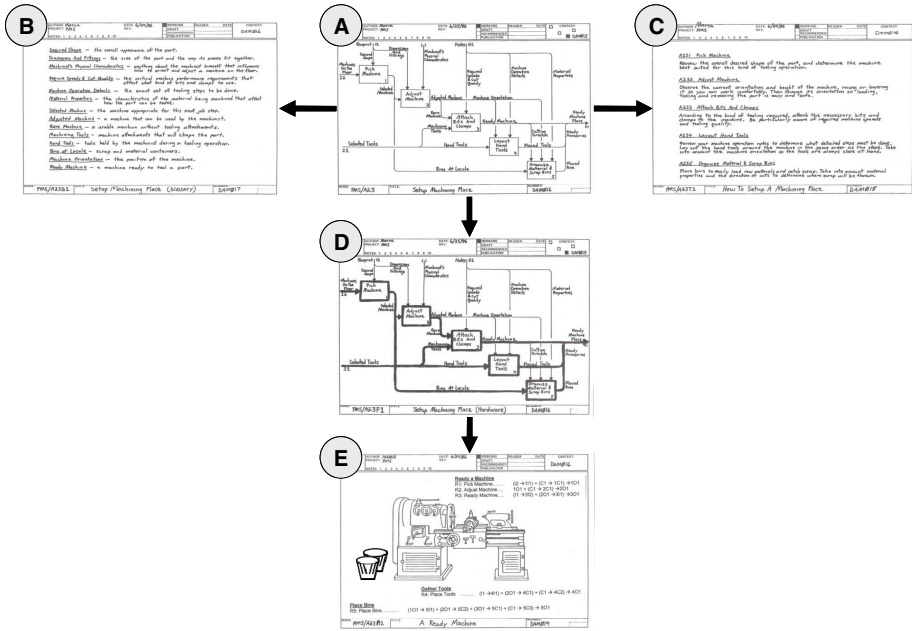


Fig. 3. A supplement set for a single SADT/IDEF0 diagram [24]

### 3.4 SADT/IDEF0 Ontology and Model Supplements Enable Augmentation

Figure 3 shows that the supplements developed directly from a single diagram comprise a rich specification of one bounded subject in the domain. Return to Table 1 and 2 in the Appendix, and identify all the ontology elements that go into these supplements: you will see the depth of SADT/IDEF0 for representing domain knowledge. As Section 2.4 says, the ontology is power enough for describing any system to any level of detail and any level of abstraction without losing context. Thus:

Hypothesis 1: A set of SADT/IDEF0 diagrams and supplements that correctly and completely describe the domain in which a software system will operate, has content that is essential for augmenting the UML, Agile, and Usability Engineering methods.

Hypothesis 2: The content of those SADT/IDEF0 models and diagram supplements can be extracted and organized so that it can become useful input to the UML, Agile, and Usability Engineering methods, and without altering those methods.

## 4 Augmentations for UML, Agile and Usability Engineering

The proposed augmentation approach centers on comprehensive, correct and consistent specifications of domain knowledge. When used properly, SADT/IDEF0 can create such specification of an entire domain, not just the immediate context, for a software system. And it can describe that domain in rich and varied ways using in-context supplements which contain the: language, beliefs, assumptions, human organization, human work, work tasks and tools, and system usage expectations, that are vital to the successful application of UML, Agile, and Usability Engineering methods. This section summarizes shortcomings and corresponding improvement recommendations, based on over 10 years of experience with the aforementioned methods. The combination of shortcomings, recommendations, and the representational power of SADT/IDEF0 diagrams and supplements led to this approach.

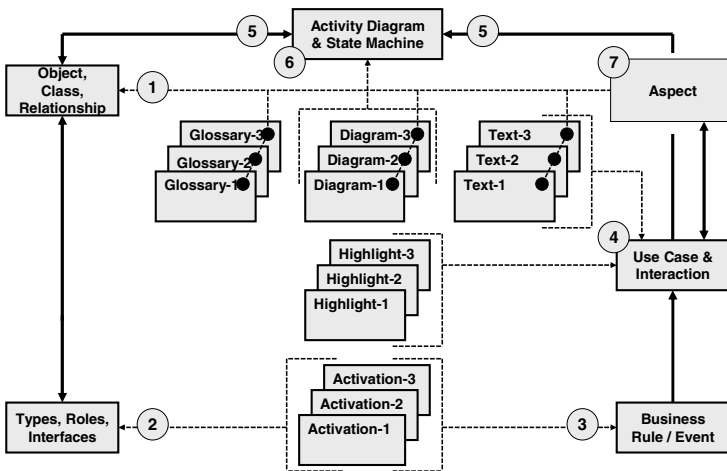
### 4.1 Benefits to UML

UML Shortcomings. Experts have consistently noted that object-oriented code design methods are better at specifying software than they are at modeling domains [16]. For domain modeling, UML considers the domain to be the entities that touch the software system [27], and that is what UML “domain model” specify. The only other outward-facing UML model, the “business model,” specifies how the software system will be used [56] [57]. These models can define a software system’s boundary, provided they are complete and accurate. But assuring completeness and accuracy without context is risky. Since modeling languages optimized for software systems are less effective at modeling the software system’s environment, augmentations have been proposed to attach more domain knowledge to UML software specifications.

SADT/IDEF0 Feature	Benefits to UML
Activation Rule	In-context specification of business rule or decision-making rule.
Annotation -- Graphic (Diagram Highlights)	In-context system use case specifications are created by telling a story based on just the highlighted boxes and arrows.
Annotation -- Text (Diagram Notes)	A well-written paragraph for each box can turn into formal descriptions of the domain.
Context Diagram, Context Model	In-context general background knowledge: a) to any highest level of abstraction, b) to any lowest level of detail.
Control Versus Input	Separation of concerns: an accurate & complete model of the control system independent from an accurate & complete model of the transaction system.
Coupling/Cohesion (assessment)	Apply these concepts to a completed model to assess pathologies in the domain. For example: “we always did it that way” becomes immediately apparent.
Data Dictionary (i.e. “glossary”)	In-context domain terminology, from which an ontology for the domain can be created,
Decomposition -- Stopping Heuristic	When decomposition stops when a function is all manual or all automated, then you have defined the human/system interface points.
Feedback Loop (output-input, output-control)	Useful for understanding: a) domain pathologies, b) interaction scenarios, c) architectural constraints.
Model Tie (i.e. “model call”) Encoding	In-context formal description of “aspect,” permits faster identification of cross-system common functionality.
Small, Multiple Models	Identification of key objects in the domain. Specify object functions independent of object modes/states.
Why, What, How (i.e. levels of abstraction)	Separation of concerns: distinct models for why (rationale), what (function), and how (mechanism) = modular understanding of context at different levels of abstraction.

**Fig. 4.** How UML can benefit from SADT/IDEF0 domain modeling

**UML Augmentation.** The augmentations that suggest strengthening UML's ability to define a software system's environment advise doing domain modeling using some other language or tool, and then linking captured knowledge to UML software specifications. For example: a) domain ontology database [6], b) general background knowledge base with reasoning logic [43], c) in-context identification, specification and validation of business rules [16] [47], and decision-making rules [58], d) how and why people do the work that they do [23], and e) formal descriptions of the domain [7]. Taken together, these augmentations suggest: a) that SADT/IDEF0 models of a domain contain knowledge that can benefit UML specifications, and b) efficacy can be achieved if domain modeling is a activity distinct from software specification. Figure 5 shows how SADT/IDEF0 diagrams and supplements can augment UML.



**Fig. 5.** The Step-by-step use of SADT/IDEF0 diagram and supplement content to augment the development of UML specifications

## 4.2 Benefits to Agile

**Agile Shortcomings.** One component of the Agile Manifesto advocates working software over comprehensive documentation [2]. Not surprisingly, traditional domain modeling methods have not heretofore been recommended for augmenting Agile software development efforts. However, "small method" augmentations have been recommended since Agile was first purported. These suggestions carefully distinguish "comprehensive" from "essential" documentation. Yes, comprehensive documentation can, when taken to the extreme, merely add time and cost to projects without adding value to the software system. But taken to the other extreme, a lack of documentation altogether often creates gaps in verified understanding between users and software developers, and leaves no rationale behind for those who maintain or wish to reuse the resulting software system. Clearly, a middle ground of specification (i.e. for domain, analysis and design) would seem to benefit all parties, so long as those specifications are efficient and effective [3]. Figure 6 summarizes the benefits.

SADT/IDEF0 Feature	Benefits to Agile Software Development
Activation Rule	Complete hierarchy of rule cause-and-effect: Highest-level rule activation causes lower-level rule activations (traceability).
Annotation -- Graphic (Diagram Highlights)	In-context system use case specifications are created by telling a story based on just the highlighted boxes and arrows.
Annotation -- Text (Diagram Notes)	In-context informal descriptions of software activations (include in prototype wrapper documentation).
Context Diagram, Context Model	In-context general background knowledge: a) to any highest level of abstraction, b) to any lowest level of detail.
Control Versus Input	Understand how to make, and then keep fixed, major object architecture decisions.
Coupling/Cohesion (assessment)	Apply these concepts to a completed model to assess pathologies in the domain. For example: "we always did it that way" becomes immediately apparent.
Data Dictionary (i.e. "glossary")	Quickly understand the user's language, and the context for language usage.
Decomposition -- Stopping Heuristic	When decomposition stops when a function is all manual or all automated, then you have defined the human/system interface points.
Feedback Loop (output-input, output-control)	Document domain knowledge using self-reflection to uncover and assess tacit knowledge and fundamental assumptions.
Model Tie (i.e. "model call") Encoding	In-context formal description of "aspect," permits faster identification of cross-system common functionality.
Small, Multiple Models	Identification of key objects in the domain. Specify object functions independent of object modes/states.
Why, What, How (i.e. levels of abstraction)	Separation of concerns: distinct models for why (rationale), what (function), and how (mechanism) = modular understanding of context at different levels of abstraction.

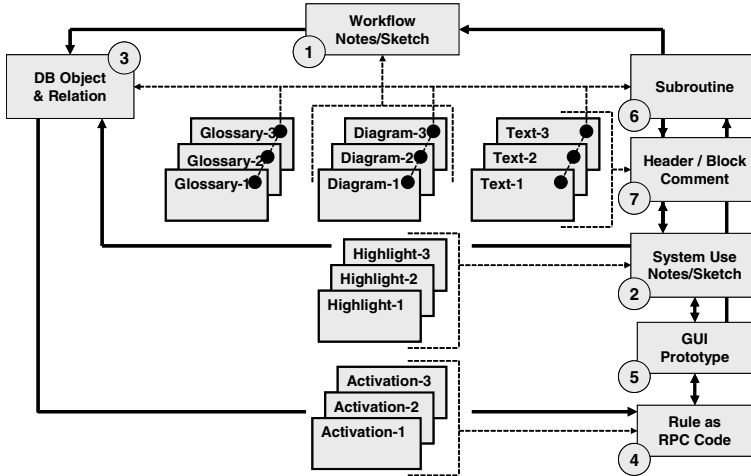
**Fig. 6.** How agile can benefit from SADT/IDEF0 domain modeling

**Agile Augmentation.** Past recommendations have suggested augmenting the informal artifacts of Agile, and advocate for practices that focus on the domain to explain why: a) people need the system, b) will use the system in particular ways, c) they expect to see certain menus, displays, interactions, and functionality, and d) they are investing their time in the software development project. For example: a) documenting domain knowledge using JAD [14], self-reflection [14], and Wikis [33] [11], b) making and keeping fixed major object architecture decisions which enable parallel development by many Agile teams in support of very large projects [34] [3], c) documenting system design knowledge with informal specifications [37], and informal tools [8], d) making explicit tacit design assumptions with Total Quality methods [12] and self-reflection [36], and e) publishing (including vital documentation) competing prototypes to the wider community for evaluation and selection a best solution for reuse [50]. Taken together, these recommendations point to an interesting line of augmentation (Figure 3) by using traditional modeling methods such as SADT/IDEF0. Figure 7 shows how SADT/IDEF0 diagrams and supplements can be used to augment Agile.

### 4.3 Benefits to Usability Engineering

**Usability Engineering Shortcomings.** Practitioners and researchers have already shown: a) how Usability Engineering can be combined with Agile [15] [59], b) that particular combinations can enable effective design space exploration [32], c) and that the prototypes from those explorations can be systematically evaluated and augmented to create best-in-class production software [49]. However, such outcomes rely on augmenting the traditional usability engineering methods with very good knowledge acquisition methods and very good modeling tools [44] [41]. To explain, Usability Engineering has had a tradition of employing the concepts and methods of participatory design [42] to obtain optimal understanding of a domain and especially the tacit knowledge of domain inhabitants. So, traditional Usability Engineering methods have employed ethnographic techniques, which have traditionally relied on

hand-written field notebooks and not on formal models. But, adding formal modeling to ethnographic practices can add value [25]. Also, with the advent of Computer Aided Software Engineering (CASE) tools, the creation and review of formal models can happen much more quickly than in the days of purely manual drawing, copying, distributing copies, the recording of feedback, and so on.



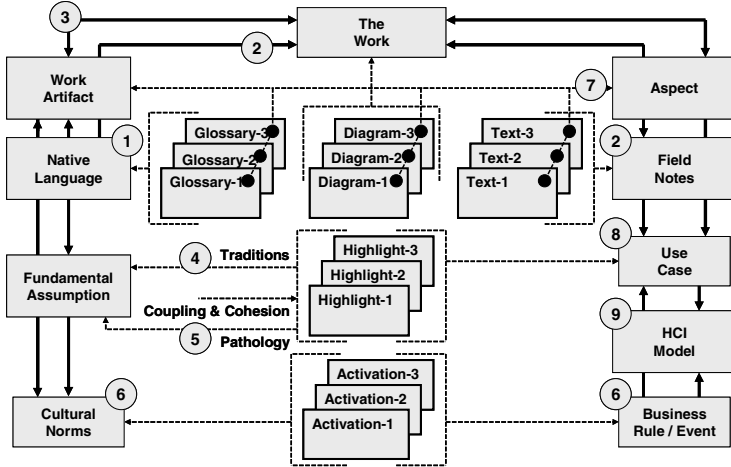
**Fig. 7.** The Step-by-step use of SADT/IDEF0 diagram and supplement content to augment the artifacts of agile software development

SADT/IDEF0 Feature	Benefits to Usability Engineering
Activation Rule	In-context interaction possibilities (patterns) and their rationale, plus associated potential implications (claims).
Annotation -- Graphic (Diagram Highlights)	In-context specification of work tasks. Context provides background and rationale for the users' work.
Annotation -- Text (Diagram Notes)	Text for all manual boxes becomes an in-context description of people's work.
Context Diagram, Context Model	Formalize and limit "context," noting how relevant information differs from context to context.
Control Versus Input	Distinguish which user generated artifacts are simply material for the next step in processing from those artifacts that govern subsequent workflow steps.
Coupling/Cohesion (assessment)	Apply these concepts to a completed model to assess pathologies in the domain. For example: "we always did it that way" becomes immediately apparent.
Data Dictionary (i.e. "glossary")	Quickly understand the user's language, and the context for language usage.
Decomposition -- Stopping Heuristic	When decomposition stops when a function is all manual or all automated, then you have defined the human/system interface points.
Feedback Loop (output-input, output-control)	Use to create test cases to evaluate software prototypes (in-context cases).
Model Tie (i.e. "model call") Encoding	Create patterns by unifying the often scattered aspects (i.e. usage behaviors) by constructing themes (i.e. relationship rules among aspects).
Small, Multiple Models	Create a context-based, generalized navigation "space" model, and then use it to create a UI presentation model.
Why, What, How (i.e. levels of abstraction)	Distinguish local dynamics from global dynamics from contextual dynamics .

**Fig. 8.** How usability engineering can benefit from SADT/IDEF0 domain modeling

Usability Engineering Augmentation. Many augmentations to Usability Engineering have been suggested, and most have been centered on incorporating ethnographic concepts and field work. Some of the most noteworthy augmentations are: a) models that distinguish local dynamics from global dynamics from contextual dynamics [45], b) a context-based, generalized navigation space model that is used that model to

create a UI presentation model [22], c) formalized context that shows how information differs from context-to-context [28], d) UI design trade-offs via patterns – in-context problem-solution pairs – and claims – implications of design decisions [1] e) a claims library that enables UI design reuse [60], e) patterns that unify the highly scattered aspects of usage behavior via a set of themes that define relationship rules for aspects [4], and f) domain models that have syntax and semantics that enable consistency across architectural, design, structural, behavioral models [13] [26].



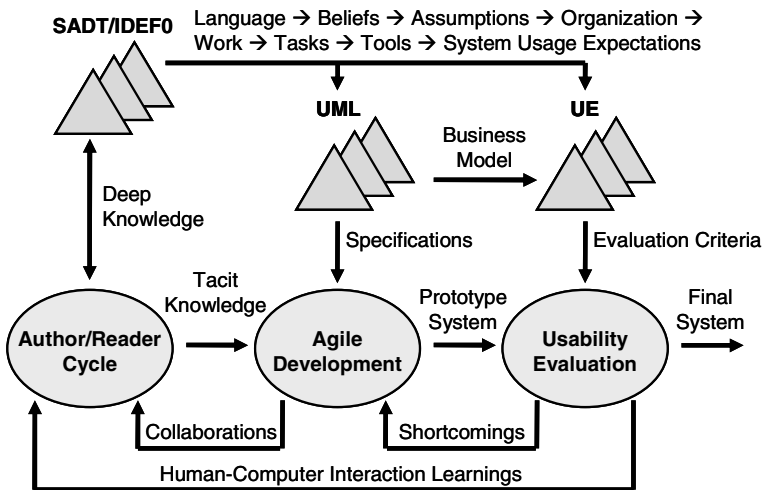
**Fig. 9.** The Step-by-step use of SADT/IDEF0 diagram and supplement content to augment usability engineering

## 5 Summary, Conclusions and Future Work

This paper has taken an approach to providing benefits to UML, Agile, and Usability Engineering methods by using SADT/IDEF0: a) for domain modeling, and b) in particular ways based on over 10 years of experience with these methods by a variety of practitioners and researchers. These experiences were selected based on their: a) advocating specific augmentations to the aforementioned methods, and b) showing how those augmentations could benefit: i) the software development process advocated by the method, ii) any software or non-software prototypes generated by the method, and iii) the reuse and maintenance of the final specifications generated by the method. The recommended shortcomings and corresponding improvement recommendations were used to develop the proposed augmentation approach.

The approach centers on comprehensive, correct and consistent specifications of domain knowledge. When used properly, SADT/IDEF0 can create such specifications of an entire domain, not just the software system's immediate context. And it can describe that domain in rich and varied ways using in-context supplements which contain the: language, beliefs, assumptions, human organization, human work, work tasks and tools, and system usage expectations, vital to the successful application of UML, Agile, and Usability Engineering methods. Figure 10 summarizes the role SADT/IDEF0 plays in the augmentation process. It also shows how the SADT/IDEF0

Author Reader Cycle [24]) can augment Agile by providing domain experts time to think about the knowledge already given to software developers to ensure facts are consistent and correct with the current common understanding.



**Fig. 10.** Augmenting UML, agile and usability engineering with SADT/IDEF0 models and the author/reader review cycle

The combination of over 10 years of experience by practitioners and researchers, their recommendations for improving upon the shortcomings they discovered, and the ability of the SADT/IDEF0 to support those recommendations, led the author to conclude that there is also merit for further elaboration and demonstration of the approach's viability by extending a commercial SADT/IDEF0 tool. Specifically, such a CASE tool could be extended by: a) enhancing its existing ontology of SADT/IDEF0, b) integrating that ontology with UML tools, c) creating an interface to a domain knowledge reasoning system and a formal specification system, and d) building a component for the automatic generation of a deep human-system interaction model that includes patterns and claims [1]. A proposal for future work is underway.

**Acknowledgements.** The author wishes to first and foremost acknowledge the late Douglas T. Ross for his contributions to the fields of industrial engineering and computer science, and for his inventions of APT, PLEX and SADT™. In alphabetical order, acknowledgements go to: Michael Connor, Melvin Dickover, Patricia Duran, Clarence Feldman, Al Irvine, Clement McGowan, Robert Munck, Chuck Patrick, Kenneth Schoman, Michelle Stowe, and the late Daniel Thornhill – the people whose early work with SADT™, and later with its public domain version IDEF0, greatly furthered: its ontology, the correct understanding of its principles, and how to correctly practice its methods. Lastly, recognition is given to the insightful researchers and practitioners cited in this paper for experiences with the UML, Agile and Usability Engineering methods, and their work in clarifying shortcomings and related practical issues surrounding the domain modeling embedded in those software development methods.



## References

1. Abraham, G., Atwood, M.: Patterns or claims: do they help in communicating design advice? In: Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group (2009)
2. Abrahamsson, P., et al.: Agile Software Development Methods: Review and Analysis. VTT Publications, Number 478, Kaitovayla (2002)
3. Alleman, G.B.: Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell about It. In: Wells, D., Williams, L. (eds.) XP 2002. LNCS, vol. 2418, pp. 70–88. Springer, Heidelberg (2002)
4. Baniassad, E., Clarke, S.: Theme: An Approach for Aspect-Oriented Analysis and Design. In: Proceedings of the 26th International Conference on Software Engineering (2004)
5. Booch, G., et al.: The Unified Modeling Language User Guide. Addison-Wesley, Boston (1999)
6. Brockmans, S., Haase, P., Hitzler, P., Studer, R.: A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 303–316. Springer, Heidelberg (2006)
7. Bryant, B., et al.: From Natural Language Requirements to Executable Models of Software Components. In: Proceedings of the Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation (2003)
8. Bryant, S., et al.: Pair programming and the re-appropriation of individual tools for collaborative software development. In: Proceedings of the Conference on Cooperative Systems Design (2006)
9. Calvary, G., Thevenin, D.: A Unifying Reference Framework for the Development of Plastic User Interfaces. In: Little, M.R., Nigay, L. (eds.) EHCI 2001. LNCS, vol. 2254, pp. 173–192. Springer, Heidelberg (2001)
10. Congram, C., Epelman, M.: How to describe your service: An invitation to Structured Analysis and Design Technique. *International Journal of Service Industry Management* 6(2) (1995)
11. Decker, B., et al.: A framework for Agile reuse in software engineering using Wiki Technology. In: Proceedings of the Knowledge Management for Distributed Agile Processes Workshop (2005)
12. Dingsøy, T., Hanssen, G.K.: Extending Agile Methods: Postmortem Reviews as Extended Feedback. In: Henninger, S., Maurer, F. (eds.) LSO 2003. LNCS, vol. 2640, pp. 4–12. Springer, Heidelberg (2003)
13. Eged, A., Medvidovic, N.: A Formal Approach to Heterogeneous Software Modeling. In: Maibaum, T. (ed.) FASE 2000. LNCS, vol. 1783, pp. 178–192. Springer, Heidelberg (2000)
14. Friedrich, W., van der Poll, J.: Towards a Method to Elicit Tacit Domain Knowledge from Users. *Interdisciplinary Journal of Information, Knowledge, and Management* 2 (2007)
15. Granollers, T., et al.: Usability Engineering Process Model. Integration with Software Engineering. In: Proceedings of the HCI International Conference 2003 (2003)
16. Halpin, T.: Augmenting UML with fact-orientation. In: Proceedings of the 34th Annual Hawaii International Conference on Systems Sciences (2001)
17. Hilliard, R.: Using Aspects in Architectural Description. In: Moreira, A., Grundy, J. (eds.) Early Aspects 2007 Workshop. LNCS, vol. 4765, pp. 139–154. Springer, Heidelberg (2007)
18. Hilliard, R.: Aspects, Concerns, Subjects, Views, ... In: OOPSLA 1999 Workshop on Multi-Dimensional Separation of Concerns in Object-Oriented Systems (1999)
19. Hilliard, R.: Views and Viewpoints in Software Systems Architecture. In: First Working IFIP Conference on Software Architecture (WICSA 1) (1999)
20. Hilliard, R., et al.: The architectural metaphor as a foundation for systems engineering. In: Proceedings of the 6th Annual International Symposium of the International Council on Systems Engineering (1996)

21. Iachello, G., Abowd, G.: From privacy methods to a privacy toolbox: Evaluation shows that heuristics are complementary. *ACM Transactions on Computer-Human Interaction* 15(2) (2008)
22. Koch, N., et al.: The Authoring Process of the UML-based Web Engineering Approach. In: *Proceedings of the 1st International Workshop on Web-oriented Software Technology* (2001)
23. Larsen, G.: Designing component-based frameworks using patterns in the UML. *CACM* 42(10) (1999)
24. Marca, D.: *IDEF0 and SADT: A Modeler's Guide*, 3rd edn. OpenProcess, Inc., Boston (2006)
25. Marca, D.: Augmenting SADT to develop computer support for cooperative work. In: *Proceedings of the 13th International Conference on Software Engineering* (1991)
26. Medvidovic, N., et al.: Round-Trip Software Engineering Using UML: From Architecture to Design and Back. In: *Proceedings of the 7th European Conference on Software Engineering* (1999)
27. Menard, R.: Domain modeling: Leveraging the heart of RUP for straight through processing. *IBM Developer Works* (2003), <http://www.ibm.com/developerworks/rational/library/2234.html> (retrieved on March 17, 2011)
28. Najar, S., et al.: Semantic representation of context models: a framework for analyzing and understanding. In: *Proceedings of the 1st Workshop on Context, Information and Ontologies, CIAO 2009* (2009)
29. Nielsen, J.: *Usability Engineering*. Academic Press, London (1993)
30. Normantas, K., Vasilecas, O., Sosunovas, S.: Augmenting UML with decision table technique. In: *International Conference on Computer Systems and Technologies: CompSys-Tech 2009* (2009)
31. Ng, J., et al.: The development of an enterprise resources planning system using a hierarchical design pyramid. *Journal of Intelligent Manufacturing* 9(5) (1996)
32. Paelke, V., Nebe, K.: Integrating Agile methods for mixed reality design space exploration. In: *Proceedings of the 7th ACM Conference on Designing Interactive Systems, DIS 2008* (2008)
33. Rech, J., et al.: Riki: A System for Knowledge Transfer and Reuse in Software Engineering Projects. In: Lytras, M., Naeve, A. (eds.) *Open Source for Knowledge and Learning Management: Strategies Beyond Tools*. IGI Global Publishers (2007)
34. Reifer, D., et al.: Scaling Agile Methods. *IEEE Software* (July/August 2003)
35. Ross, D.: Structured Analysis (SA): A Language for Communicating Ideas. *IEEE Transactions on Software Engineering* 3(1) (1977)
36. Salo, O., Kolehmainen, K., Kyllönen, P., Löthman, J., Salmijärvi, S., Abrahamsson, P.: Self-Adaptability of Agile Software Processes: A Case Study on Post-iteration Workshops. In: Eckstein, J., Baumeister, H. (eds.) *XP 2004*. LNCS, vol. 3092, pp. 184–193. Springer, Heidelberg (2004)
37. Scacchi, W.: Is Open Source Software Development Faster, Better, and Cheaper than Software Engineering? In: *Proceedings of the 2nd ICSE Workshop on Open Source Software Engineering* (2002)
38. Schoman, K., Ross, D.: Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering* 3(1) (1977)
39. Seffah, A., et al.: HCI, Usability and Software Engineering Integration: Present & Future. In: *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle*. HCI Series, vol. 8 (2005)
40. Siltala, M.: Modeling Contracting Procedure and the Concept of the Service Portfolio for Finnish Municipalities using SADT. *Nordic Journal of Surveying and Real Estate Research* 1 (2009)

41. Sitou, W., Spanfelner, B.: Towards requirements engineering for context adaptive systems. In: 31st Annual International Computer Software and Applications Conference, COMP-SAC 2007, vol. 2 (2007)
42. Spradley, J.: Participant Observation. Holt, Rinehart and Winston, London (1980)
43. Süß, J., Leicher, A.: Augmenting Domain Specific UML Models with RDF. In: Proceedings of the 3rd Workshop in Software Model Engineering, Lisbon (2004)
44. Sutcliffe, A.: Applying small group theory to analysis and design of CSCW systems. In: Proceedings of the Workshop on Human and Social Factors of Software Engineering, HSSE 2005 (2005)
45. Sutcliffe, A.: On the effective use and reuse of HCI knowledge. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7(2) (2000)
46. Sutcliffe, A.: The Domain Theory for Requirements Engineering. *IEEE Transactions on Software Engineering* 24(3) (1998)
47. Skersys, T., Gudas, S.: The Enhancement of Class Model Development Using Business Rules. In: Bozanis, P., Houstis, E.N. (eds.) PCI 2005. LNCS, vol. 3746, pp. 480–490. Springer, Heidelberg (2005)
48. Wahid, S.: Investigating design knowledge reuse for interface development. In: Proceedings of the 6th Conference on Designing Interactive Systems, DIS 2006 (2006)
49. Verlinden, J., Horva, I.: Analyzing opportunities for using interactive augmented prototyping in design practice. In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing. Cambridge University Press (2009)
50. Lethbridge, T.C., Laganière, R.: Object-Oriented Software Engineering: Practical Software Development Using UML and Java. McGraw-Hill, London (2001)
51. Winckler, M., et al.: Tasks and scenario-based evaluation of information visualization techniques. In: Proceedings of the 3rd Annual Conference on Task Models and Diagrams, TAMODIA 2004 (2004)
52. Novak, J., Cañas, A.: The Theory Underlying Concept Maps and How to Construct Them, Technical Report IHMC CmapTools 2006-01 Rev 01-2008, Florida Institute for Human and Machine Cognition (2008)
53. Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J., Neema, S.: Developing applications using model-driven design environments. *IEEE Computer* 39(2) (2006)
54. Hruby, P.: Ontology-based domain-driven design. In: OOPSLA Workshop on Best Practices for Model-Driven Software Development, San Diego, CA, USA (2005)
55. Wikipedia: Integrated Computer-Aided Manufacturing (2011), [http://en.wikipedia.org/wiki/Integrated\\_Computer-Aided\\_Manufacturing](http://en.wikipedia.org/wiki/Integrated_Computer-Aided_Manufacturing) (retrieved March 20, 2011)
56. Coste, P., et al.: Multilanguage Design of Heterogeneous Systems. In: CODES 1999 (1999)
57. Stuikeys, V., Damasevicius, R.: Relationship Model of Abstractions Used for Developing Domain Generators. *Informatica* 13(1) (2001)
58. Vasilecas, O., Normantas, K.: Decision table based approach for business rules modelling in UML/OCL. In: Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop (2010)
59. Seffah, A., et al.: HCI, Usability and Software Engineering Integration: Present & Future. In: Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle. HCI Series, vol. 8 (2005)
60. Brel, C., Renevier-Gonin, P., Occello, A., Déry-Pinna, A.-M., Faron-Zucker, C., Riveill, M.: Application Composition Driven By UI Composition. In: Bernhaupt, R., Forbrig, P., Gulliksen, J., Lárusdóttir, M. (eds.) HCSE 2010. LNCS, vol. 6409, pp. 198–205. Springer, Heidelberg (2010)

# Appendix

## 1 “Features” of the SADT Ontology

Table 1. SADT “features” published in 1977 by Douglas Ross [35]

	PURPOSE	CONCEPT		MECHANISM	NOTATION
Context	1 BOUND CONTEXT	INSIDE/OUTSIDE		SA BOX	
Arrow	2 RELATE/CONNECT	FROM/TO		SA ARROW	
Transform	3 SHOW TRANSFORMATION	INPUT-OUTPUT		SA INTERFACE	
Control	4 SHOW CIRCUMSTANCE	CONTROL		SA INTERFACE	
Means	5 SHOW MEANS	SUPPORT		SA MECHANISM	
Verbs	6 NAME APTLY	ACTIVITY HAPPENINGS	DATA THINGS	SA NAMES	ACTIVITY [VERB]      DATA [NOUN]
Nouns	7 LABEL APTLY	THINGS	HAPPENINGS	SA LABELS	[NOUN] →      → [VERB]
Path	8 SHOW NECESSITY	I-O	C-D	PATH	
Dominance	9 SHOW DOMINANCE	C	I	CONSTRAINT	
Relevance	10 SHOW RELEVANCE	ICO	ICO	ALL INTERFACES	
Omissions	11 OMIT OBVIOUS	C-O	I-O	OMITTED ARROW	
Branches Joins	12 BE EXPLICIT WITHOUT CLUTTER	PIPELINES, CONDUITS, WIRES		BRANCH	
	13			JOIN	
	14 BE CONCISE AND CLEAR	CABLES, MULTI-WIRES		BUNDLE	
	15			SPREAD	
OR AND	16 SHOW EXCLUSIVES	EXPLICIT ALTERNATIVES		OR BRANCH	
	17			OR JOIN	
Boundary Parent ICOM	18 SHOW INTERFACES TO PARENT DIAGRAM	PARENT CHILD ARROWS PENETRATE		SA BOUNDARY ARROWS (ON CHILD)	
	19 SHOW EXPLICIT PARENT CONNECTION	NUMBER CONVENTION FOR PARENT, WRITE ICON CODE ON CHILD BOUNDARY ARROWS			
	20 SHOW UNIQUE DECOMPOSITION	DETAIL REFERENCE EXPRESSION (DRE)		C-NUMBER OR PAGE NUMBER OF DETAIL DIAGRAM	
Calls	21 SHOW SHARED OR VARIABLE DECOMPOSITION	DRE WITH (MODEL NAME)		SA CALL ON SUPPORT	

Table 1. (continued)

Feedback  
 Pipeline  
 Tunnel  
 To/From All  
 Note  
 Footnote  
 Meta-Note  
 Squiggle  
 Sequence  
 Node  
 Model  
 Interface  
 To-From  
 Reference  
 Dominance  
 Description  
 Highlights  
 Glossary  
 Index

	PURPOSE	CONCEPT	MECHANISM	NOTATION
22	SHOW COOPERATION	INTERCHANGE OF SHARED RESPONSIBILITY	SA 2-WAY ARROWS	
23	SUPPRESS INTERCHANGE DETAILS	ALLOW 2-WAY WITHIN 1-WAY PIPELINES	2-WAY TO 1-WAY BUTTING ARROWS	
24	SUPPRESS "PASS-THROUGH" CLUTTER	ALLOW ARROWS TO GO OUTSIDE DIAGRAMS	SA "TUNNELING" (WITH REFERENCES)	
25	SUPPRESS NEEDED-ARROW CLUTTER	ALLOW TAGGED JUMPS WITHIN DIAGRAM	TO ALL or FROM ALL	
26	SHOW NEEDED ANNOTATION	ALLOW WORDS IN DIAGRAM	SA NOTE	NOTE:
27	OVERCOME CRAMPED SPACE	ALLOW REMOTE LOCATION OF WORDS IN DIAGRAM	SA FOOTNOTE	
28	SHOW COMMENTS ABOUT DIAGRAM	ALLOW WORDS ON (NOT IN) DIAGRAM	SA META-NOTE	
29	ENSURE PROPER ASSOCIATION OF WORDS	TIE WORDS TO INTENDED SUBJECT	SA "SQUIGGLE"	
30	UNIQUE SHEET REFERENCE	CHRONOLOGICAL CREATION	SA C-NUMBER	AUTHOR INITIATIVE INTEGER
31	UNIQUE BOX REFERENCE	PATH DOWN TREE FROM BOX NUMBERS	SA NODE NUMBER (BOX NUMBERS)	P, D, OR M < PARENT # < BOX #
32	SAME FOR MULTI-MODELS	PRECEDENCE BY MODEL NAME	SA MODEL NAME	MODEL NAME/NODE#
33	UNIQUE INTERFACE REFERENCE	ICOM WITH BOX NUMBER	SA BOX ICOM	BOX # - ICOM CODE
34	UNIQUE ARROW REFERENCE	FROM - TO	PAIR OF BOX ICOMs	BOX ICOM <sub>1</sub> BOX ICOM <sub>2</sub>
35	SHOW CONTEXT REFERENCE	SPECIFY A REFERENCE POINT	SA REF. EXP. "DOT"	A122.411 "WHICH SEE"
36	ASSIST CORRECT INTERPRETATION	SHOW DOMINANCE GEOMETRICALLY (ASSIST PARSE)	STAIRCASE LAYOUT	
37	ASSIST UNDERSTANDING	PROSE SUMMARY OF MESSAGE	SA TEXT	NODE# : T < INTEGER
38	HIGHLIGHT FEATURES	SPECIAL EFFECTS FOR EXPOSITION ONLY	SA FEOS	NODE# : F < INTEGER
39	DEFINE TERMS	GLOSSARY WITH WORDS & PICTURES	SA GLOSSARY	MODEL NAME G < INTEGER
40	ORGANIZE PAGES	PROVIDE TABLE OF CONTENTS	SA NODE INDEX	NODE# ORDER

# From Requirements to Code: A Full Model-Driven Development Perspective\*

Óscar Pastor, Marcela Ruiz, and Sergio España

Centro de Investigación ProS, Universitat Politècnica de València, València, Spain  
{opastor, lruiz, sergio.espana}@pros.upv.es

**Abstract.** Models play a paramount role in model-driven development (MDD): several modelling layers allow defining views of the system under construction at different abstraction levels, and model transformations facilitate the transition from one layer to the other. However, how to effectively integrate requirements engineering within model-driven development is still an open research challenge. This paper shows a full MDD approach that covers from requirements engineering to automatic software code generation. This has been achieved by the integration of two methods: Communication Analysis (a communication-oriented requirements engineering method [1]) and the OO-Method (a model-driven object-oriented software development method [2]). For this purpose, we have proposed a systematic technique for deriving conceptual models from from business process and requirements models; it allows deriving class diagrams, state-transition diagrams and specifications of class service behaviour. The approach has been evaluated by means of an ontological evaluation, lab demos and controlled experiments; we are currently planning apply it under conditions of practice in an action research endeavour.

**Keywords:** Information Systems, Requirements Model, Business Process Model, Model Transformation, Class Diagram, Communication Analysis, OO-method.

## 1 Introduction

The level of abstraction of software engineering methods has been lifted over the years, in order to tackle with the ever increasing complexity of software development projects: from assembler code, to structured programming languages, object-oriented programming languages, aspect-orientation, service-orientation... A new paradigm referred to as model-driven development (MDD) has recently placed all the emphasis on the role of models, to the point of changing the motto “the code is the model” to “the model is the code” [3]. “Software has the rare property that it allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods”<sup>1</sup>. Ideally, the computerised information

---

\* Research supported by projects GVA ORCA (PROMETEO/2009/015), MICINN PROS Req (TIN2010-19130-C02-02), Santiago Grisolia grant (GRISOLIA/2011/005) and co-financed with ERDF.

<sup>1</sup> Bran Selic and John Hogg have expressed this idea in a number of presentations (e.g. Selic keynote at Workshop on Critical Systems Development with UML, San Francisco, CA, USA, October 2003).

system is specified in an abstract model and then, by means of model transformations, subsequent models are obtained until the source code is automatically generated. The advent of the Model-Driven Architecture (MDA) [4] as a development paradigm and the Unified Modeling Language (UML) as a *de facto* standard notation have paved the way for viable MDD proposals (e.g. Extreme Non-Programming [5], Conceptual Schema-Centric Development [6]).

However, despite the importance of requirements engineering as a key success factor for software development projects, there is a lack of MDD methods that cover the full development lifecycle from requirements engineering to code generation. Most MDD methods range from conceptual modelling to code generation and do not address requirements. There are definitely many open research challenges in model-driven requirements engineering [7]. Furthermore, the code generation capabilities of most MDD methods are currently limited to create-read-update-delete (CRUD) operations and more complex reactions need to be programmed manually.

In this article, we describe an attempt to bridge the gap from requirements to fully functional software code in an MDD way. We have integrated two existing methods: Communication Analysis (a communication-oriented business process modelling and requirements engineering method) [1] and the OO-Method (an object-oriented model-driven development framework with automatic code generation capabilities<sup>2</sup>) [2]. The contributions of this article are the following:

- An overview of an MDD method that covers the software development lifecycle from requirements engineering to code generation is provided, explaining how it was the result of the integration of two methods.
- A detailed example of how a set of requirements evolves from their elicitation to the generated code is shown.

The paper is structured as follows. Section 2 presents the OO-Method framework, a complete MDD environment with specific methods for requirements engineering and for conceptual modelling, which moreover provides for model compilation features. Section 3 describes the overall OO-Method framework, focusing on how one of the requirements engineering methods was integrated into the framework; it also provides an outline of the corresponding model transformation. Section 4 exemplifies how the requirements evolve from their elicitation to the software code generation. Section 5 presents conclusions and future work.

## 2 The OO-Method Framework

The OO-method was first proposed in the academy [8], on top of a formal object-oriented algebra named OASIS. Soon a spin-off company named CARE Technologies was created, with the aim of creating the tool to support the OO-method. The result of this endeavour was the Integranova Model Execution system. The suite of tools includes Integranova Modeler, a computer-aided software engineering (CASE) tool that allows specifying the Conceptual Model, and a model compiler. Both the OO-Method and the Integranova technology are used in practice to develop enterprise information systems.

---

<sup>2</sup> Integranova Model Execution System

<http://www.care-t.com> Accessed 12-2011.

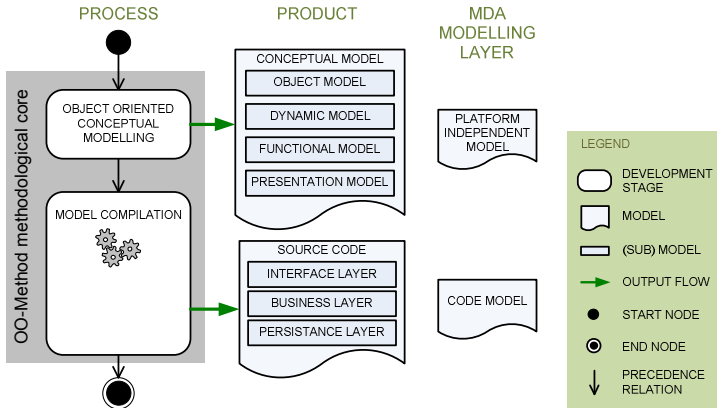


Fig. 1. Methodological core of the OO-method

## 2.1 The Conceptual Modelling Core

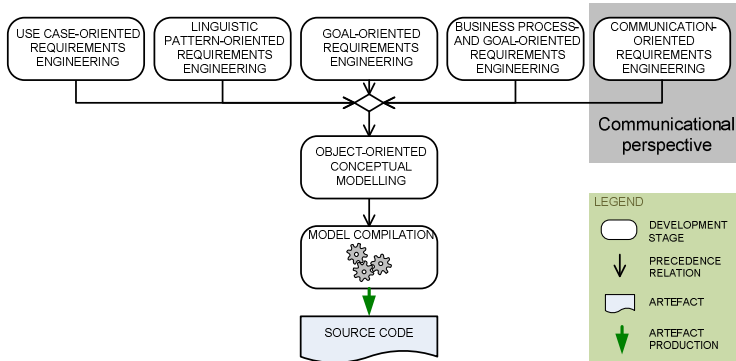
The OO-method methodological core is the *object-oriented conceptual modelling* stage, the result of which is the *Conceptual Model*, an object-oriented model that describes the computerised information sub-system disregarding the implementation platform. This way, the OO-method Conceptual Model corresponds to the Platform Independent Model (PIM) layer of the Model Driven Architecture. It is comprised of four interrelated models:

- The *Object Model* allows specifying the static aspect of the system in the form of a UML-compliant class diagram. This model allows describing the following elements of the computerised information sub-system (among others): the business objects in terms of classes of objects, structural relations among classes of objects, agents of the system (an abstraction of the users of the software system) and relations among agents and class services.
- The *Dynamic Model* specifies the possible sequences of method invocations that can occur in the life of an object; it is expressed as a State-Transition Diagram.
- The *Functional Model* specifies the effect that the method invocations have in the state of the objects; it is expressed as generic pseudo-code specifications (which is independent of any programming language).
- The *Presentation Model* offers an abstract description of the computerised information sub-system interface. This model is structured in three abstract pattern levels.

After the conceptual modelling stage, a model compiler takes as input the conceptual model and a set of compilation parameters (e.g. the selected database management system and programming language) and it generates the source code of a software application that is functionally equivalent to the conceptual model (i.e. the application fulfils the specifications determined by the model). The automatically generated software application is fully functional and it is organised in a three-layer architecture: interface, business logic and persistence.



Several proposals build upon the methodological core of the OO-Method. Some of these proposals extend the method with modelling techniques aimed at a specific type of software system (e.g. OOWS [9] for web applications). Also, some extensions addressing different requirements engineering orientations have been proposed (see the available catalogue in Fig. 2). We outline them in the following.



**Fig. 2.** Several approaches to requirements engineering in the OO-method

The use case-oriented proposal defines a requirements model that describes *what* the computerised information sub-system has to do, and a requirements analysis process that offers methodological guidance to derive a conceptual model [10]. First, a Requirements Model is created; it consists of a Mission Statement, a Function Refinement Tree and a Use Case Diagram. Then, the Requirements Analysis Process is a strategy to describe concrete details such as the composition of the computerised information sub-system; it comprises a set of Sequence Diagrams and Function Decomposition Tables.

With regards to the linguistic-pattern-based approach, a natural step after defining a requirements engineering approach based on use cases is the definition of a restricted natural language for the textual templates [11]. First, the analyst creates a Use Case Model and Linguistic Model of Use Cases (a set of normalised use-case specification templates). A set of transformation patterns is applied to the Linguistic Model of Use Cases so as to derive an initial version of the OO-Method Object Model, as well as an Interaction Model (which is a UML Sequence Diagram that is not part of the core Conceptual Model).

A goal-oriented approach proposed by [12] proposes creating business models based on the  $i^*$  framework. First, a Business Model is created, which consists of a Goal-Refinement Tree, a Strategic Dependency Model and a Strategic Rationale Model. These models are later used to derive a functional requirements model based on Use Cases and Scenarios.

A recent business process- and goal-oriented approach proposes several models aimed at describing the organisation and its work practice [13]. The first step is to perform organisational modelling (i.e. a glossary, a domain data model, a set of business process diagrams, etc.). Then the problems and needs of the organisational system are analysed by means of a goals/strategies diagrams and operationalisation tables. Next, the analyst performs the specification of system requirements using

Extended Task Descriptions. Finally the OO-Method Object Model and part of the Dynamic Model can be derived by means of manual transformation guidelines.

This article focuses in the communication-oriented approach since it has been fully integrated with the conceptual modelling stage and tool support has been provided, both for creating and transforming models.

## 2.2 Communication Analysis

Since information systems are a support to organisational communication [14], a communicational approach to information systems analysis is necessary. Communication Analysis is a requirements engineering method that analyses the communicative interactions between the information system and its environment; it was, therefore, a good candidate for completing the catalogue of requirements engineering approaches within the OO-Method framework.

The methodological core of Communication Analysis is the *information system analysis* stage, the result of which is an *analysis specification*, a communication-oriented documentation that describes the information system disregarding its possible computerisation. This way, the analysis specification produced by Communication Analysis corresponds to the CIM layer of the Model Driven Architecture.

Communication Analysis offers a requirements structure and several modelling techniques for business process modelling and requirements specification. The *Communicative Event Diagram* is intended to describe business processes from a communicational perspective. A *communicative event* is a set of actions related to information (acquisition, storage, processing, retrieval, and/or distribution), that are carried out in a complete and uninterrupted way, on the occasion of an external stimulus. Business process model modularity is guided by unity criteria [15]; there are evidences that the application of these criteria improve the quality of models [16]. The *Event Specification Template* allows structuring the requirements associated to a communicative event. Among other requirements, it contains a description of the new meaningful information that is conveyed to the information system in the event. This is specified by means of *Message Structures*, a modelling technique that is based on structured text. Previous work [17] presents the grammar of Message Structures and provides guidelines for their application during analysis and design (they are used differently in each development stage). To create the message structures, the analyst interviews the users and analyses the available business forms. They merely describe messages and are, therefore, an analysis artefact. The structure of message fields lies vertically and field properties can be arranged horizontally; e.g. information acquisition operation, field domain, an example value provided by users, etc.

At the time we undertook the integration of Communication Analysis into the OO-Method, a strong theoretical foundation and several specifications of the requirements engineering method were available [18]. There was experience with the method in action. By means of technology transfer projects, the method had been adopted by several companies: (a) the Valencia Port Authority, (b) the Infrastructure and Transport Ministry of the Valencian Regional Government, (c) and Anecoop S. Coop. However, requirements models were mainly specified using word processors and general-purpose diagramming tools. Also, no attempts to integrate Communication Analysis in an MDD framework had been made. The final software implementation was either carried out within the organisation or it was outsourced.

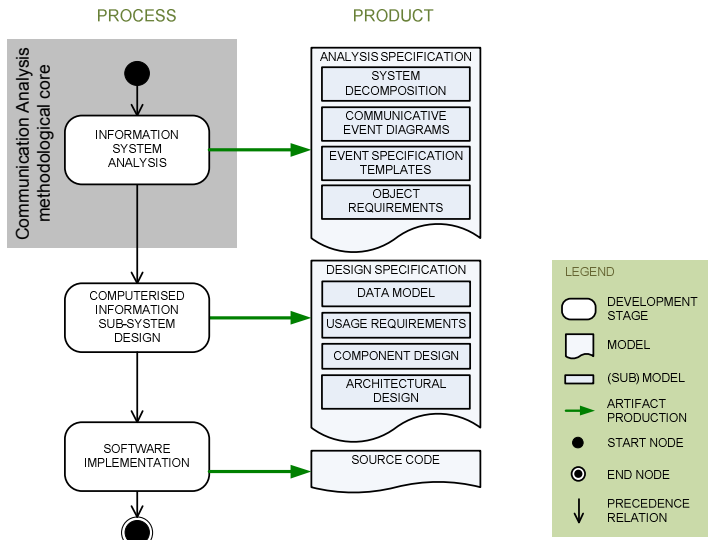


Fig. 3. Methodological core of communication analysis

### 3 Integration

The first steps were aimed at improving the method specification, providing more rigorous definitions for the underlying concepts [1] and designing the artefacts needed for a successful integration into an MDD framework, such as a method metamodel. Also, an Eclipse-based tool was implemented in order to support the creation of Communication Analysis requirements models [19]. Then the concepts of both methods were aligned in order to provide a sound theoretical basis and to envision how the integration should be performed. As a result, a flow of activities has been defined (see Fig. 4). The activities from Communication Analysis that correspond to the information system analysis stage have been preserved, but those related to design have been substituted by the OO-Method conceptual modelling stage.

Furthermore, a derivation technique aimed at obtaining a first version of the conceptual model from a requirements model has been proposed. Two kinds of derivation are provided: namely, a manual derivation intended to be performed by an analyst, and model transformation that automates the process as much as possible. Depending on the development project context (e.g. expertise of the team, organisational culture concerning MDD), one of the alternatives is to be chosen.

The manual derivation technique consists of a set of guidelines that allow to systematically reason the elements of the conceptual model, including the Object Model, the Dynamic Model and part of the Functional Model. The Presentation Model is, for the moment, out of the scope since many improvements are being made on human-computer interaction [20-22] and the derivation should take these into account once they are consolidated.

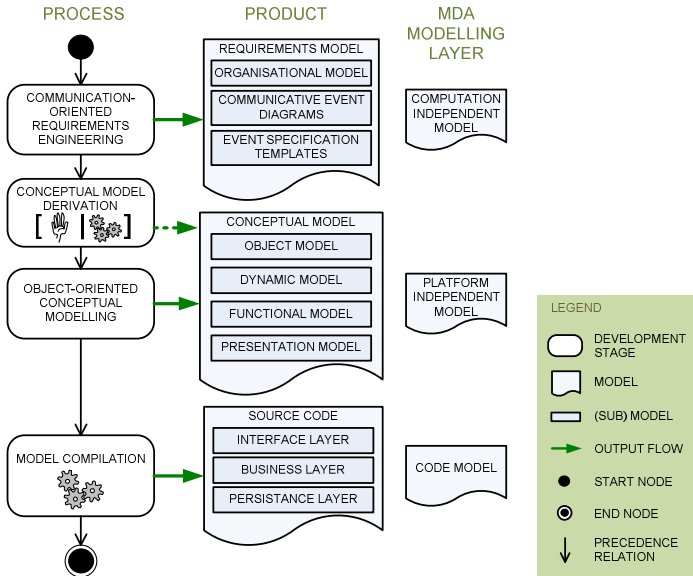


Fig. 4. Activities of the integrated method

The derivation of the Object Model consists of three main steps. Firstly, the scope of the derivation is defined; that is, the analyst may want to derive the conceptual model for the whole requirements model or just for some part of it (e.g. only the sales management business process). This is done by marking which communicative events are intended to be supported by the conceptual model. In the former case, all events are marked by default. In the latter case, the marked communicative events are added to a diagram, including the precedences between them; then, the diagram is extended by including any other events that are precedent to the marked ones. Secondly, the communicative events in the transformation-scope diagram are sorted according to their precedences so as to later process them in order. These two steps prevent inconsistencies in the conceptual model such as referencing an inexistent class. The third step implies processing each communicative event to create its class-diagram view; that is, producing the portion of the class diagram that corresponds to the communicative event. For this purpose, the message structure is processed, and other requirements in the event specification templates are taken into account (e.g. restrictions). By incrementally integrating all the class-diagram views, a complete object model is obtained. Not only the classes and relationships are derived, but also most of their properties; for instance, attribute properties, relationship cardinalities, class services and their arguments. See [23] for more details.

The derivation of the Dynamic Model consists of creating a state-transition diagram for each class that is affected by several communicative events, so as to constrain how the users can trigger the software functions over a given business object. The Dynamic Model can be obtained mainly by processing the Communicative Event Diagram. The main derivation guideline is the following: communicative events are converted into transitions and precedence relationships are converted into states. However, business process gateways lead to more complex

state-transition diagrams. Also, additional transitions can be added to the state transition diagram; for instance, transitions that correspond to atomic edition and destruction services (e.g. edit and destroy, respectively), as well as transitions that correspond to atomic services that take part in a transaction (these transitions appear as a result of processing the event specification templates). See [24] for more details.

Last but not least, the derivation of the Functional Model completes the conceptual model by specifying the reaction of the class services. Valuation rules provide the meaning to atomic services, whereas transaction formulas allow defining more complex behaviours.

## 4 The Life of Requirements

The proposed MDD framework allows to transform requirements to elements of the conceptual model and, from there, to pieces of the final software application. Traceability is enhanced by creating many trace links during the development process, most of them automatically as a result of model transformations. Although the current state of the framework provides for post-RS traceability (e.g. the classes that are affected by a business process), pre-RS traceability can still be recorded and maintained manually (e.g. the representative user that formulated a specific requirement during an interview). In the following, we illustrate the life of a set of requirements during a development project that applies the proposed MDD framework. The example is based on the SuperStationery Co. case, a lab demo that is reported in full detail in [26]. Keep in mind that, in the following, we provide a simulation that illustrates the application of the method.

SuperStationery Co. is a fictional intermediary company that provides office material to its clients. To place an order, most clients phone the Sales Department, where they are attended by a salesman. Then the client requests products that are to be sent to one or many destinations (a client company can have decentralised offices). The salesman takes note of the order. Then the Sales Manager assigns the order to one of the many suppliers that work with the company. The supplier can either accept or reject the order. In case of acceptance several arrangements are made before the order is shipped.

John is a salesman from the company who has participated as representative user in the development project. Henry is the senior analyst who has been in charge of the development. He conducted several joint application development sessions. The outcome of the sessions was written into proceedings documents, but we provide the transcription of part of an interview.

- John: Yes, each order is assigned to a supplier. Well, actually, I do not do that myself... It is Mr Lowen [the Sales Manager] who does this. I can show you the info that he provides me. There! These fields...
- (John points at several fields of the order form, see Fig. 5)
- Henry: Are all orders assigned to suppliers?
- John: Yes.
- Henry: Can an order be assigned to several supplier. For instance, in case it is big.

- John: No, that is not how we work. A supplier always has to serve all the items referred in the order. But the assignation is tentative; the suppliers actually commit once they know the full details.
- Henry: So, they can refuse to serve it? [...]

ORDER					
Order number: 10352		Request date: 31-08-2009			
Payment type: <input checked="" type="checkbox"/> Cash <input type="checkbox"/> Credit <input type="checkbox"/> Cheque		Planned delivery date: 05-09-2009			
Client					
VAT number: 56746163-R					
Name: John Papiro Jr.					
Telephone: 030 81 48 31					
Supplier					
Code: OFFIRAP					
Name: Office Rapid Ltd.					
Address: Brandenburgen street, 46, 2983 Millhaven					
Destination: Blvd. Blue mountain, 35-14A, 2363 Toontown					
Person in charge: Brayden Hitchcock					
#	Code	Product name	Price	Q	Amount
1	ST39455	Rounded scissors (cebra) box-100	25,40 €	35	889,00 €
2	ST65399	Staples cooper 26-22 blister 500	5,60 €	60	336,00 €
3	CA479-9	Stereofoam cups box-50 (pack 120)	18,75 €	10	187,50 €
					1412,50 €
Destination: Greenhouse street, 23, 2989 Millhaven					
Person in charge: Luke Padbury					
#	Code	Product name	Price	Q	Amount
1	ST65399	Staples cooper 26-22 blister 500	5,60 €	30	444,50 €
2	CA746-3	Sugar lumps 1kg	2,30 €	3	6,90 €
					451,40 €
<b>Total</b>					1863,90 €

Fig. 5. Fields that correspond to the assignment of an order to a supplier

Fig. 6 presents part of the communicative event diagram of the *Sales management* business process. It addresses the communicative event mentioned by John; that is SALE 2. *Sales manager assigns supplier*. Fig. 7 presents three message structures related to SALE 2. The first one is incorrect because it does not capture the real essence of the message, beyond the actual pieces of data that the current business form contains. For instance, it does not acknowledge the fact that the data about the supplier is already known by the company. The second message structure is correct but we consider it inappropriate at analysis time; it acknowledges the fact that the name and the address of a supplier are derived information but this is indeed design information (it reflects the current technological support; e.g., if a web service was finally implemented to support the assignation, probably it would not contain the fields Name and Address to avoid inconsistencies). The third message structure indeed follows the Communication Analysis guidelines.

Other requirements are specified within the event specification templates. Once the requirements model is complete enough, the analyst proceeds to derive an initial version of the conceptual model. This can be done manually (by applying a set of manual derivation guidelines) or automatically (by executing the ATL transformation rules). We now provide an example of the rationale behind the model transformation strategy, disregarding whether it is automated or not.

The communicative event SALE 2 affects the same business object as communicative event SALE 1; namely, the client order. Note that the reference field **Order** in the message structure of SALE 2 indicates the business object being modified.

This way, the class that corresponds to this business object is extended; namely, CLIENTORDER is affected by SALE 2. The data fields in the message structure of SALE 2 lead to adding new attributes to this class, whereas the reference fields in the message structure lead to adding new structural relationships between class CLIENTORDER and other classes that already exist in the class diagram under construction.

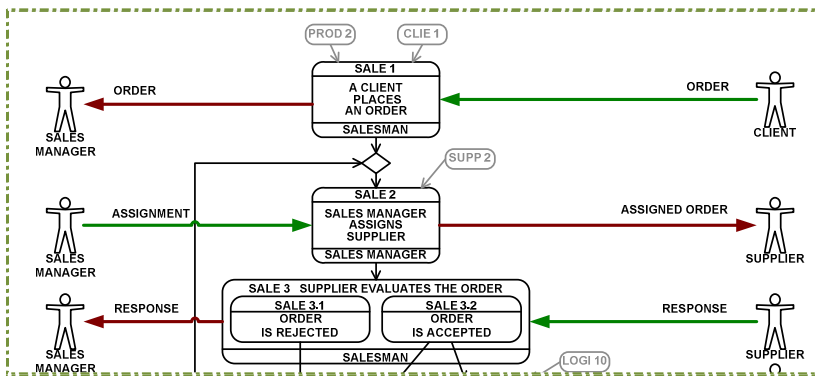


Fig. 6. Part of the communicative event diagram of the Sales management business process

FIELD	OP	DOMAIN	EXAMPLE VALUE
<b>ASSIGNMENT =</b> < Assignment date + <b>SUPPLIER =</b> < Code + Name + Address >	i	date	01-09-2009
<b>ASSIGNMENT =</b> < Order + Assignment date + <b>SUPPLIER(Code)=</b> < Name + Address >	i	Client order	10352
<b>ASSIGNMENT =</b> < Order + Assignment date + Supplier >	i	Client order	10352
<b>ASSIGNMENT =</b> < Order + Assignment date + Supplier >	i	date	01-09-2009
<b>ASSIGNMENT =</b> < Order + Assignment date + Supplier >	i	Supplier	OFFIRAP, Office Rapid Ltd.

**LEGEND**

*TYPES OF FIELDS*

- DATA FIELDS HAVE A BASIC DOMAIN (e.g. Assignment date)
- REFERENCE FIELDS HAVE A BUSINESS OBJECT TYPE AS A DOMAIN (e.g. Supplier)

For more details see [17]

**GRAMMATICAL CONSTRUCTS**

< + > AGGREGATION SUBSTRUCTURE

{ } ITERATION SUBSTRUCTURE

[ ] SPECIALISATION SUBSTRUCTURE

Fig. 7. Several message structures related to communicative event SALE 1

With regards to data fields, the field Assignment date leads to adding an attribute named assignment\_date to the class CLIENTORDER. Fig. 8 depicts this derivation and specifies the details of the new attribute. All attributes that are added to a class as a

result of a class extension have the following properties: these attributes are not part of the identification function, the attribute type is Variable, they are not requested upon creation, and they allow nulls. The data type is derived from the domain of the field, according to a set of heuristics (this case is trivial, the data type of assignment\_date is Date because the field domain of Assignment date is *date*).

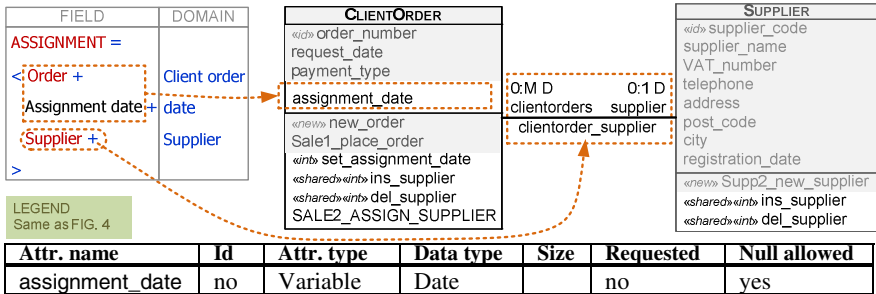


Fig. 8. Class diagram view of SALE 2 and specification of the new attribute

With regards to reference fields, the field *Supplier* references a business object that was processed in the communicative event SUPP 2. Therefore, a structural relationship is defined between the class CLIENTORDER and the class SUPPLIER. The cardinality is defined as 0:1 in the side of SUPPLIER because the orders are not assigned to suppliers when they are placed, but it occurs in a later moment in time. For the same reason, it is dynamic.

With respect to the services, several are added. A service is added to the class in order to introduce the values of this attribute; it is named set\_assignment\_date. Two shared services are included in both classes due to the cardinality of the structural relationship, and the fact that it is dynamic; namely an insertion shared service named ins\_supplier and a deletion shared service named del\_supplier (shared events manage links between instances). A transaction named S2\_ASSIGN\_SUPPLIER is defined in order to execute atomically the services set\_assignment\_date and ins\_supplier.

By incrementally integrating the class diagram views, the complete Object Model is obtained. Then the Dynamic Model and the Functional Model are derived as well.

All in all, it is possible to maintain pre-RS traceability. For instance, to trace the communicative event SALE 2 to its source: the representative user John, who formulated them during an interview. The fields in the message structure ASSIGNMENT can be traced back to both the user John and to the fields of the current paper form where they were formerly supported. It is also possible to maintain post-RS traceability. There exist trace links between elements of the requirements model and the conceptual model. For instance, during the derivation of the conceptual model, SALE 2 has generated its own class diagram view; this view includes attributes of the classes CLIENTORDER and SUPPLIER. More specifically, there are trace links from the message structure fields to the attributes of the classes (e.g. the data field Assignment date has led to the derivation of CLIENTORDER.assignment\_date) and to the structural relationships (e.g.the reference field *Supplier* has led to the derivation of the relationship clientorder\_supplier). Also, there are trace links between the elements of the



conceptual model and the software components of the generated application. For instance, the attribute `CLIENTORDER.assignment_date` is presented in the interface, as shown in Fig. 9.

As promoted by the MDD paradigm, user requirements evolve during the development process; they are transformed to elements of the subsequent models until they end up being projected onto the final software product, keeping traces of their evolution.

The screenshot shows a window titled "ClientOrder" with a search bar containing "10352". Below the search bar is a table of order details with columns "Name" and "Value".

Name	Value
Order number	10352
Request date	8/31/2009
Payment type	Cash
Planned delivery date	
VAT number	56746163
Client name	John Papiro Jr.
Client telephone	030 91 48 31
Assignment date	09/01/2009
Supplier code	1
Supplier name	Office Rapid Ltd.
Supplier address	Brandenburguen street, 46
Total	1,587.4

Below the details table is a section for "Destination" with tabs for "Client", "Supplier", "TruckDriver", and "InsurancePolicy". The "Client" tab is active, showing a table with columns "Address" and "Person in charge".

Address	Person in charge
Blvd. Blue mountain, 35-14	Brayden Hitchcock
Greenhouse street, 23	Luke Padbury

At the bottom of the window is a table with columns "#", "Code", "Product name", "Price", "Q", and "Amount".

#	Code	Product name	Price	Q	Amount
1	39455	Rounded scissors (cebra) b...	25.4	35	88
2	65399	Staples cooper 26-22 blister...	5.6	60	33
3	4799	Stereofam cups box-50 (pac...	18.75	10	187

The window has a "Close" button at the bottom right.

Fig. 9. Screenshot of the automatically generated application

## 5 Some Open Challenges and Future Work

There are many open challenges in the area of MDD, especially in industry adopted MDD methods. We now enumerate some areas where our proposal can be improved.

First of all, we acknowledge that practitioners are usually reluctant to use non-standard notations. We therefore plan adopt the Business Process Modeling Notation (BPMN) to support Communication Analysis business process modelling. The BPMN Choreography Diagram is the first candidate to represent communicative event diagrams. Three challenges stem from this decision. First, a careful investigation needs to be carried out to adopt the notation while preserving the concepts and criteria of the method. Second, the derivation technique needs to be adapted to deal with the new notation. Third, a proper tool support needs to be provided, in order to facilitate validation, promote adoption, etc. With regards to the latter challenge, there are plenty

of business process management suites in the wild these days; among them, Oryx<sup>3</sup> stands as a clear candidate because of its open, academic nature; also Modelio<sup>4</sup>, since it has recently moved to open source.

Furthermore, we plan to investigate how other analytical perspectives (e.g. goal or value orientation) may extend our approach and become useful under certain project circumstances. In Communication Analysis, business processes are means to fulfil the business goals; however, no methodological support is provided to elicit or model goals. This research line is still active due to its many open challenges [28].

Moreover, ontological analyses, lab demos and controlled experiments have been carried out, but further validations of the derivation technique are planned. This includes an action research application of the derivation technique in a pilot development project, probably carried out in the context of GEM Biosoft<sup>5</sup> (a spin-off of the Research Center on Software Production Methods). Several controlled experiments have already been carried out and their data is being analysed; others are being designed. There is a long way ahead, but we are confident that the scientific community will work together to fulfill the vision of the MDD paradigm: providing full model-driven support to all the activities of a software project.

**Acknowledgements.** We are indebted to Arturo González, for his wise advices on taking the most out of Communication Analysis. We also thank CARE Technologies for their support in using the Integranova MDD suite, including its model compiler.

## References

1. España, S., González, A., Pastor, Ó.: Communication Analysis: A Requirements Engineering Method for Information Systems. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 530–545. Springer, Heidelberg (2009)
2. Pastor, O., Molina, J.C.: Model-Driven Architecture in practice: a software production environment based on conceptual modeling, 302 p. Springer, New York (2007)
3. Embley, D.W., Liddle, S.W., Pastor, O.: Conceptual-model programming: a manifesto. In: Embley, D.W., Thalheim, B. (eds.) Handbook of Conceptual Modeling, pp. 3–16. Springer (2011)
4. OMG. MDA Guide Version 1.0.1 (2003), <http://www.omg.org/docs/omg/03-06-01.pdf> (cited 2008 12-2010)
5. Morgan, T.: Business rules and information systems - Aligning IT with business goals. Addison-Wesley (2002)
6. Olivé, À.: Conceptual Schema-Centric Development: A Grand Challenge for Information Systems Research. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 1–15. Springer, Heidelberg (2005)
7. Loniewski, G., Insfran, E., Abrahão, S.: A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part II. LNCS, vol. 6395, pp. 213–227. Springer, Heidelberg (2010)

---

<sup>3</sup> <http://oryx-project.org>

<sup>4</sup> <http://www.modeliosoft.com>

<sup>5</sup> <http://www.gembiosoft.com>

8. Pastor, Ó., Gómez, J., Insfrán, E., Pelechano, V.: The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. *Information Systems* 26(7), 507–534 (2001)
9. Fons, J., Pelechano, V., Albert, M., Pastor, Ó.: Development of Web Applications from Web Enhanced Conceptual Schemas. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) *ER 2003*. LNCS, vol. 2813, pp. 232–245. Springer, Heidelberg (2003)
10. Insfrán, E., Pastor, Ó., Wieringa, R.: Requirements engineering-based conceptual modelling. *Requirements Engineering* 7(2), 61–72 (2002)
11. Díaz, I., Sánchez, J., Matteo, A.: Conceptual Modeling Based on Transformation Linguistic Patterns. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) *ER 2005*. LNCS, vol. 3716, pp. 192–208. Springer, Heidelberg (2005)
12. Estrada, H., Martínez, A., Pastor, Ó.: Goal-Based Business Modeling Oriented towards Late Requirements Generation. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) *ER 2003*. LNCS, vol. 2813, pp. 277–290. Springer, Heidelberg (2003)
13. de la Vara, J.L., Sánchez, J., Pastor, Ó.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: Bellahsene, Z., Léonard, M. (eds.) *CAISE 2008*. LNCS, vol. 5074, pp. 213–227. Springer, Heidelberg (2008)
14. Langefors, B.: *Theoretical analysis of information systems*, 4th edn. Studentlitteratur, Lund (1977)
15. González, A., España, S., Pastor, Ó.: Unity criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem. In: *Third International Conference on Research Challenges in Information Science, RCIS 2009*, Fes, Morocco, pp. 173–182. IEEE (2009)
16. España, S., Condori-Fernández, N., González, A., Pastor, Ó.: Evaluating the completeness and granularity of functional requirements specifications: a controlled experiment. In: *17th IEEE International Requirements Engineering Conference, RE 2009*, Atlanta, Georgia, USA, pp. 161–170. IEEE (2009)
17. González, A., Ruiz, M., España, S., Pastor, Ó.: Message Structures: a modelling technique for information systems analysis and design. In: Lencastre, M., Estrada, H. (eds.) *14th Workshop on Requirements Engineering, WER 2011*, Rio de Janeiro, Brazil (2011), extended version in English and Spanish available at <http://arxiv.org/abs/1101.5341>
18. González, A.: *Algunas consideraciones sobre el uso de la abstracción en el análisis de los sistemas de información de gestión* (PhD thesis) Some considerations on the use of abstraction in management information systems analysis (in Spanish), in Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (2004)
19. Ruiz, M., España, S., Gonzalez, A., Pastor, O.: Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos. In: Avila-García, O., et al. (eds.) *VII Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM 2010)*, Jornadas de Ingeniería de Software y Bases de Datos (JISBD), Valencia, España, pp. 70–77 (2010)
20. Valverde, F., Pastor, O.: Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) *WISE 2009*. LNCS, vol. 5802, pp. 131–144. Springer, Heidelberg (2009)
21. Panach, J.I., España, S., Moreno, A.M., Pastor, Ó.: Dealing with Usability in Model Transformation Technologies. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008*. LNCS, vol. 5231, pp. 498–511. Springer, Heidelberg (2008)

22. Aquino, N., Vanderdonckt, J., Pastor, O.: Transformation templates: adding flexibility to model-driven engineering of user interfaces. In: Shin, S.Y., et al. (eds.) 25th ACM Symposium on Applied Computing, SAC 2010, Sierre, Switzerland, pp. 1195–1202. ACM (2010)
23. González, A., España, S., Ruiz, M., Pastor, Ó.: Systematic Derivation of Class Diagrams from Communication-Oriented Business Process Models. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 246–260. Springer, Heidelberg (2011)
24. España, S., Ruiz, M., Pastor, Ó., González, A.: Systematic derivation of state machines from communication-oriented business process models. In: IEEE Fifth International Conference on Research Challenges in Information Science, RCIS 2011, Guadeloupe - French West Indies, France. IEEE (2011)
25. Gotel, O.C.Z., Finkelstein, C.W.: An analysis of the requirements traceability problem. In: 1st International Conference on Requirements Engineering (1994)
26. España, S., González, A., Pastor, Ó., Ruiz, M.: Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo.2011, Technical report ProS-TR-2011-01, ProS Research Centre, Universitat Politècnica de València, Spain (2011), <http://arxiv.org/abs/1101.0105>
27. OMG. Business Process Modeling Notation (BPMN) version 2.0 (2011), <http://www.omg.org/spec/BPMN/2.0/> (cited 2011 04-2011)
28. Cardoso, E., Almeida, J.P.A., Guizzardi, R.S.S., Guizzardi, G.: A method for eliciting goals for business process models based on non-functional requirements catalogues. *International Journal of Information System Modeling and Design* 2(2), 1–18 (2011)

**Part I**  
**Enterprise Software Technology**

# Enabling Automatic Process-Aware Collaboration Support in Software Engineering Projects

Gregor Grambow<sup>1</sup>, Roy Oberhauser<sup>1</sup>, and Manfred Reichert<sup>2</sup>

<sup>1</sup> Computer Science Dept., Aalen University, Aalen, Germany

<sup>2</sup> Institute for Databases and Information Systems, Ulm University, Ulm, Germany  
{gregor.grambow, roy.oberhauser}@htw-aalen.de,  
manfred.reichert@uni-ulm.de

**Abstract.** Software Engineering (SE) remains an immature discipline and SE projects continue to be challenging due to their dynamic nature. One problematic aspect is the coordination of and collaboration among the many individuals working in such projects. Numerous efforts to establish software engineering environments (SEEs) to address this aspect have been made. However, since SE projects depend on individuals and their intentions, their collaboration is still performed manually to a large degree. Manual tasks are subject to human error in omission or commission that can result in communication breakdowns which are compounded within multi-project environments. This paper describes a synergistic approach that extends a process-aware information system with contextual awareness and integrates this in a SEE. This enables the system to support the users with active and passive information and support collaboration. Context information is presented to the users, providing them with process navigability information relating to their current activities. Additionally, automated information distribution improves awareness about the actions of others. Finally, this approach enables the automatic initiation and governance of follow-up activities caused by changes implied by other activities.

**Keywords:** Computer-supported Cooperative Work, Process-centered Software Engineering Environments, Process-aware Information Systems, Context-awareness, Semantic Web Applications.

## 1 Introduction

Recently, a trend towards greater automation and process-centricity can be observed in various industries for achieving predictable quality and efficiency [1]. Typically, process automation is applied in domains with foreknown and predictable activity sequences such as production, business, and logistics. In the software development domain, low-level operational workflows involving collaborations typically aberrate sufficiently to make process automation especially challenging.

To enhance the automated coordination capabilities in software engineering environments (SEEs), various challenges must be addressed. Software development is project-oriented and lacks the typical production stage with repeatable activities or

interactions. Process-Centered Software Engineering Environments (PCSEEs) [2] support such projects with both tooling and processes, yet these must be tailored to the unique and diverse project and product needs (e.g., quality levels, team size, etc.). While common software engineering (SE) process models (e.g., VM-XT [3] or Open Unified Process [4]) have proven to be beneficial, they are typically manually implemented (especially in small-to-medium enterprises), often remain coarse in their granularity, are documented to an often general level, and rely on humans to follow and map actual low-level concrete actions and events to the appropriate higher-level process (*process navigability*).

In this paper, the following definition of process and workflow will be used: *Process Management* deals with the explicit identification, implementation, and governance of processes incorporating organizational or business aspects. *Workflow management*, in turn, deals with the automation of business processes or parts thereof. Consequently, a workflow is the technical implementation of a process (or part thereof).

A lack of automatic process guidance and support in an SEE can result in a disparity between the specified and the executed process, and lead to unpredictable process and product quality. Furthermore, uncoordinated activities may occur, affecting process efficiency. From the process perspective, activities and workflows can be roughly separated in two categories: *Intrinsic* activities are planned and executed as part of the SE process model (e.g., VM-XT [3] or Open Unified Process [4]). *Extrinsic* activities, in turn, are executed outside the reference process model and are thus unplanned and difficult to trace or support. For an example of *extrinsic* vs. *intrinsic* workflows, we refer to Fig. 1. The figure shows a source code modification activity (*intrinsic*) that causes necessary modifications on other artifacts. These modification activities are not part of the process (*extrinsic*).

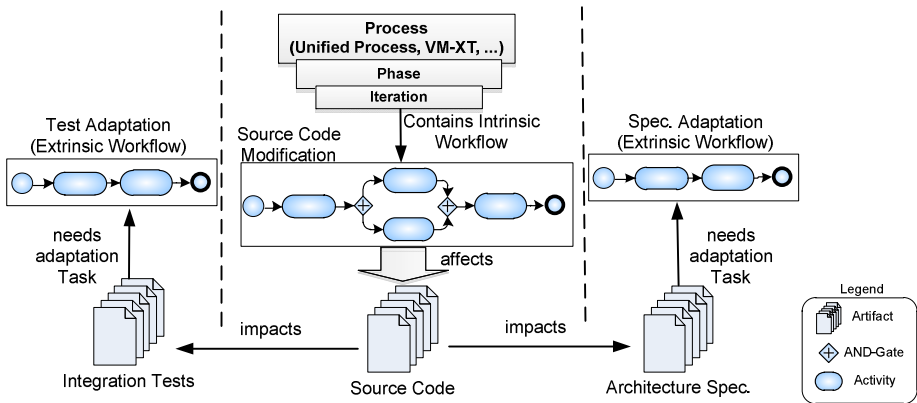


Fig. 1. *Intrinsic and Extrinsic workflows*

Our previous work has described a holistic framework that applies semantic technologies to SE lifecycles [5] and integrates context-awareness and PAIS (Process-Aware Information System) technology [6] to provide SE process support. [7] dealt

with the explicit modeling and execution support for *extrinsic* activities utilized for the automated treatment of specialized issues in SE projects (e.g., bug fixing or refactoring). [8] investigated consistency in the modeling of processes and workflows in SE to unite abstractly specified processes as well as the concretely and automatically supported workflows. Finally, automatic integration of quality aspects into processes was investigated in [9][10][11].

To comprehensively support the SE process, various other aspects should also be considered: As the SE process largely depends on individuals and their collaboration, the concrete triggering and orchestration of collaboration activities is desirable. To enable configurable collaboration support, various activity dependencies should be supported. For instance, direct follow-up actions may be necessary while in other cases notification to other team members may suffice. Extrinsic follow-up activities should be connected to the appropriate *intrinsic* activities that caused them to support traceability and integration into the SE process. In support of user contextual-awareness, automated guidance should not only be provided for the activities in one workflow (horizontal connections between the activities), but also vertically, making the hierarchical connections between processes and workflows explicit.

This paper presents an approach for collaboration support featuring different capabilities of active and passive information provision to users in an SE project. Furthermore, the connection of *intrinsic* and *extrinsic* activities is addressed, featuring a context-based reasoning process to automatically derive consequences of activities (e.g., impacts on other artifacts) and to govern follow-up activities. Additionally, the connection between abstract processes and concrete workflows is emphasized, providing this information to the user to support *navigability* and process awareness. The following three points sum up the contribution of this paper:

- Individuals working in multi project environments are supported by the automatic provision of extended activity information and process navigability information.
- Automatic information distribution is enabled to inform individuals about various events in a project including the actions of others.
- Automatic initiation and governance of related follow-up activities required by certain actions is provided.

The structure of the paper is as follows: the problems addressed are illustrated in the next section, followed in Section 3 with a description of our solution approach. Section 4 shows the application of our approach to the illustrated problems. Section 5 addresses the issue of the additional effort required. Section 6 then discusses related work, followed by the conclusion.

## 2 Problem Scenario

The issues being addressed will be illustrated using typical situations in a software company: various projects are executed in parallel by different teams of different sizes. People often have to switch between different projects, and within each project, larger numbers of people are working on the same artifacts. Without additional



coordination effort things can easily be forgotten. Activities mostly imply changes to artifacts, and thus not only relations between *intrinsic* and *extrinsic* activities exist, but there is also a continuously changing artifact base. These facts result in the three problems illustrated below:

**Problem A. Project Switching.** One issue reported by developers is related to frequent project switching. A person doing this in such a multi-team / multi-project environment has to manually gather context information after a switch to work effectively: Which assignment has to be processed for which project? Which are potential milestones and deadlines? What is the state of the currently processed assignment? What are upcoming activities to be completed?

**Problem B. Change Notification.** When cooperatively working on the same artifact base, activities and the accompanying changes to artifacts often remain unnoticed by other people. For example, if two teams (e.g. a development team and a test team) are working on the same source code artifacts, they might want to be informed about changes to the artifacts. Such information is often transferred manually and is therefore prone to forgetfulness.

**Problem C. Follow-up Action Implications.** Also when cooperatively working on the same artifact base, artifact changes often imply certain follow-up actions that are hitherto coordinated manually. This is typically dependent on the artifacts, their relations, and the type of change (e.g., interfaces concern the architect, implementation changes concern the testers, GUI changes concern the user manual author). Fig. 1 depicts a scenario detailing this: It concerns a source code artifact that is part of an interface component: since the file belongs to an interface component, the applied changes possibly not only affect the file's unit tests, but also other artifacts such as the architecture specification or integration tests. These additional activities are usually neither covered by the SE process nor governed by workflows; manual coordination can lead to impacts being overlooked and result in inconsistencies, e.g., between the source code and the tests or specifications. The fact that these activities belong to different project areas with often also different responsible persons makes this even more difficult. Even if not forgotten, follow-up actions could benefit from automated governance and support. Furthermore, it can be difficult to determine which stakeholder should be informed about which change and when, especially considering the dynamic and diverse nature of the artifact-to-stakeholder relationship and various information needs.

### 3 Automatic Coordination Support

This section starts with a brief introduction of the framework we continue to develop for supporting the SE process. In particular we want to make clear what capabilities this approach can draw on. For further technical details on its realization, we refer to [11]. The essence of our solution approach is the combination of an adaptive PAIS with semantic technology. A *Process Management* module is used to model both

*intrinsic* and *extrinsic* workflows in an integrated way, while additional information about hierarchical dependencies and the context are stored and processed in a semantic-based context management module. To acquire information about the environment, low-level events occurring during SE tool usage (e.g., saving a file or changing code) are extracted and combined to derive higher-level activities such as creating a unit test.

The realization of the solution approach is the Context-aware Software Engineering Environment Event-driven framework (CoSEEEK). It is comprised of modules in a service-based architecture: The *Process Management* module orchestrates SE activities for all project participants. Adaptive PAISs support the coordination of activities according to a pre-specified process model as well as dynamic process changes (e.g., to add, delete, or move activities) in order to cope with unforeseen situations [13][14][15][16]. To enable *Context Management*, semantic technology was chosen due to its many advantages [17], especially a vocabulary including logic statements about the modeled entities and relations as well as a taxonomy for these entities. Furthermore, well-structured ontologies also enhance interoperability between different applications and agents, fostering knowledge sharing and reuse as well as enabling automated consistency checking. The *Context Management* component makes heavy use of semantic technology, utilizing an OWL-DL [18] ontology as well as SWRL [19] for semantic rules processing and SPARQL [20] for semantic querying. Programmatic access to the ontology is supported by the Jena framework [21]. Automatic reasoning capabilities as well as the execution of SWRL rules [22] (while guaranteeing that their execution does not lead to violations of description logic statements) are enabled by Pellet [23].

*Event Extraction* primarily utilizes sensors for collecting contextual state changes in external elements via events and data associated with various SE tools. Therefore, the sensor framework Hackystat [24] is applied. These low-level atomic events and data are aggregated in the *Event Processing* module, which uses complex event processing (CEP) [25] to create high-level events with contextual semantic value.

The combination of these modules enables CoSEEEK to automatically manage ad-hoc dependencies of certain activities in an either active or passive information distribution fashion to provide coordination support.

### 3.1 Active Coordination Support

Active coordination support enables the system to automatically assign follow-up activities to responsible persons or teams. To realize this, the system must be aware of the *intrinsic* activities and workflows that may cause the need for coordination. These workflows, which are based on the users' planned activities (called *Assignments* here, e.g., develop some feature) and which are part of the SE process, are created within CoSEEEK or imported from external process management tools (e.g., MicroTool inStep) in use by an organization. In this paper, OpenUP is used as SE process model. *Assignments* concerning software development are executed by the 'Develop Solution Increment' workflow in that model and imply certain activities like 'Implement Solution' or 'Implement Tests' for the user. The detection of required follow-up activities is realized featuring a three-phased approach:

**1. Determine Projects Areas Being Affected by an Activity:** The first step is configurable and can take into account various facts to determine which *areas* of a project are affected. For the third problem in Section 2 such a configuration can be ‘*Search for affected areas in case of technical issues if an activity implies a change to an artifact and the artifact is a source code artifact belonging to an interface component*’.

**2. Determine the Concrete Target Being Affected within the Area:** The second step takes the selected *areas* and the target of the applied activity as input. This target can be a concrete artifact as in the given scenario or a more abstract *section* of the project as, e.g., a module. The concrete target is then determined via relations of the different *sections*. An example for this can be implementation and testing: the testing (structural or retesting) of an artifact relates to its implementation. In the given example, the relation does not need to be in place for the concretely processed component, but can be also found if one exists elsewhere in the hierarchy (e.g., the module the concrete artifact belongs to). If there is no direct relation from the processed source code artifact, the system looks for other components the file belongs to (e.g., the module).

**3. Determine the Information Recipient Being Responsible for the Chosen Target:** Once the target of the information distribution or follow-up action is determined, the responsible persons or teams have to be discovered. For example, if the target of the follow-up action is a source code file with no direct responsible party defined, the overlying *sections* are taken into account, e.g., the encapsulating module. If a team is responsible, the information is referred to the designated contact of that team for further distribution.

To enable such automated information distribution, a system must be aware of various facts of the project. Furthermore, to realize automated detection of follow-up actions, different concepts have to be present in the system in order to enable awareness of them:

- (1) The project has to be hierarchically split up into components like areas or modules.
- (2) Connections of relating components must be established; e.g., the fact that testing a module relies on implementing that module.
- (3) Information that can be used to clarify under which circumstances one area affects another must be present.
- (4) Different components must be classified; e.g., a package in the source code that realizes the interface of a component.

To support this, the CoSEEEK *Context Management* component contains representations of various project facts. To support awareness and to enrich workflow execution with context information, as shown in Fig. 2, workflows enacted within the *Process Management* module are annotated by concepts in the *Context Management* module. A workflow is mapped by a *Work Unit Container* and an activity is mapped by a *Work Unit*. These are in turn extended by *Assignments* and *Assignment Activities*,

which explicitly represent the content of the work the user has to perform for the project. Different areas of a project (like ‘Implementation’ or ‘Testing’) are explicitly modeled by the *Area* concept (1), while further separation of the project into logical components is done by the *Project Component* (2). The latter is an abstract building block for structuring a project, which has various subclasses.

Fig. 2 shows two of the subclasses of the *Building Block: Artifact*, which is used for various types of processed artifacts (like documents or source code files), and the *Section* that is used for concrete structuring purposes (e.g., used to map a source code package). An *Assignment Activity* being executed by a *Person* processes a certain *Project Component*. A *Project Component*, in turn, has a responsible *Role* taken by a *Resource* that is a *Team* or a *Person*. To enable the configuration of various possible impacts of an activity within the system, different concepts are used: The *Potential Impact* captures potential impacts between *Areas*, like ‘When a technical change happens to a component in Area a, this has an impact on Area b’. *Project Components* of different *Areas* can be related to each other, like ‘Testing of Module x relates to the implementation of Module x’. Many of the concepts also have asserted subclasses for further classifying them. These subclasses of which two are shown in Fig. 2 (3) are dependent on certain conditions. For example, if a *Section* is connected to problems that were detected by the system (e.g., code problems indicated by static analysis tools), the integrated reasoner automatically infers that it belongs to concept *Risk Section*.

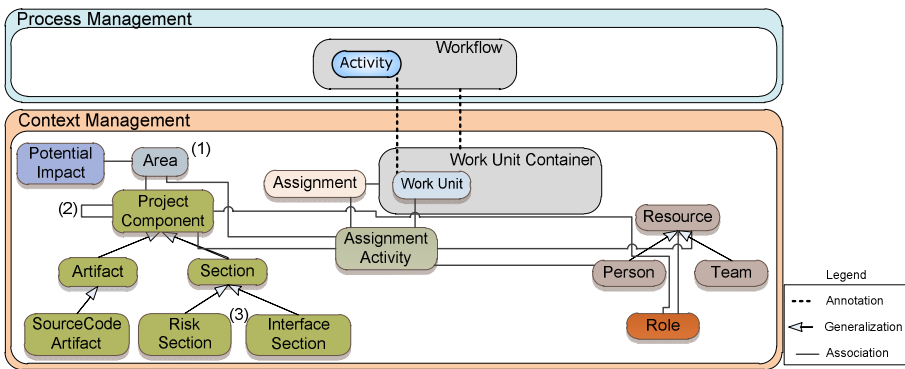


Fig. 2. Concepts enabling active coordination support

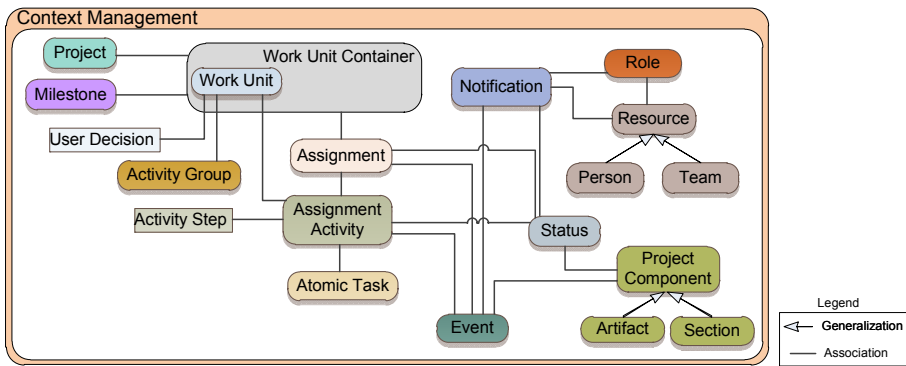
### 3.2 Passive Coordination Support

Passive coordination support comprises the provisioning of *process navigability information* and *automatic change notifications* for users.

*Navigability information* support is enabled since the workflows governing the users’ activities are mapped by concepts in the *Context Management* module. Thus additional information becomes available to the user that can be useful, e.g., when switching between the activities of different projects. The additionally modeled activity information is illustrated in Fig. 3 and explained below. Additional

information comprises the current user *Assignment*: the *Assignment Activity*, *Activity Steps*, the current *Task*, and the *Activity Group* to which the current *Activity* belongs. These concepts can be useful for capturing exactly what the user is doing at the moment as well as for additional support information coming from the process. An example for all additional information presented here is provided in Section 4.

In the *Context Management* module, a concept exists mapping internal variables used for workflow governance to so-called *User Decisions*. That way the user can decide how the workflow is actually executed, incorporating information of the current situation that cannot be known a priori. That way, the user not only has a more semantic and usable influence on the workflow, but also knows what lies ahead. As the more abstract process regions are connected to the operational workflows, the user can also directly receive information about them. This includes e.g., information about the current *Project* or its *Milestones*, which are also modeled in the *Context Management* component.



**Fig. 3.** Concepts enabling passive coordination support

*Automatic change notification* is the second passive coordination ability provided by the system. To support users in their collaboration and to counteract forgetfulness, automatic notifications can be beneficial in the first case for two situations in SE projects: When events happen that relate to activities or artifacts and when status changes occur according to the latter. Therefore, several concepts in the *Context Management* component are involved as shown in Fig. 3.

To be able to easily add notification support for the aforementioned example in Section 2, explicit concepts for *Event* and *Status* are utilized. Primarily, *Events* relate to events that occur in the context of a SE project and that are automatically detected by the *Event Management* module. The *Status* concept has been introduced to explicitly model the status of various other concepts such as *Assignments* or *Artifacts*. In an SE project, various artifacts exist with different relations belonging to different areas of the project. Examples include requirements specifications or source code artifacts. To be able to explicitly describe this in the *Context Management* module, the *Project Component* is used as abstract building block for structuring of a project.

Specializations of this concept are the aforementioned *Artifacts* and *Sections*. As example consider a source code structure where the *sections* depict the source code packages. User management in the *Context Management* component includes

concepts for roles, persons, and teams. A *Role* can be used as a placeholder for an activity when it is not yet known who should execute the activity. They can also be used in relation to *Project Components* to express, e.g., that a *Person* is responsible for a certain source code package. *Persons* and *Teams* are abstracted to a *Resource* concept to enable the assignment of activities to *Teams* as well as single *Persons*. Utilizing all of the aforementioned concepts, it is easily possible with the *Notification* concept to configure user notifications relating to various events and status changes in a SE project. Two types of notifications are supported: General notifications that are abstractly pre-defined, e.g., a notification for a role in a process that has not yet started. This notification is distributed to the person executing the role when the process is running. The second type is user-related notifications that can be added by the users themselves, as when a user wants to be kept up to date on the status of a certain *Artifact*.

## 4 Application Example

For validating our solution, the problems from Section 2 are used. Prior work investigated the practicality of technical aspects such as performance with regard to CoSEEEK realization elements [7], [9].

For the problem example (A) of a user switching between different projects, the solution illustrates the usability of additional process navigability information. In one project, she deals with requirements elicitation and executes the ‘Identify and Refine Requirements’ workflow from the OpenUP process [4]. In the other project, she develops software executing the ‘Develop Solution Increment’ workflow. Fig. 4 shows diverse supplementary information on the *Activities* as it is specified in the OpenUP process. There are supportive *Activity Steps* (as e.g., “Gather Information”), a so called discipline for the *Activity* (e.g., “Requirements”, also provided by the OpenUP process), the current processed task (e.g., “Coding”) and the specific *User Assignment* (as ‘Develop Feature X’). Additionally, the specific project (e.g., ‘Project A’) and its milestones according to the OpenUP process (e.g., ‘Initial Operational Capability’) are also included. In the ‘Develop Solution Increment’ workflow there are many decisions for potential loops or optional activities. These decisions are dependent on internal workflow variables. In this example the mapping from workflow variables to user decisions is done in a way that the user can directly select the next upcoming activity. As shown in the example, after the ‘Implement Solution’ activity, there are four possible successors the user can directly choose.

The second problem (B) deals with information requirements relating to different people and teams working on the same artifact base. The solution for this is a pre-configured *Notification* to inform users or teams being responsible for source code packages of changes made to them. As the *Notification* is pre-defined, it does not relate to a concrete *Person* or *Team* but to a *Role* defined for a *Section*. This *Role* is later taken by a *Resource*; when detecting that changes to *Artifacts* contained in that *Section* are made, the *Resource* is automatically notified. However, users can configure personalized *Notifications* as well: Assume that a user is interested in a certain *Assignment* of another user as her work relies on it. Therefore, she registers for a new *Notification* relating to the state of the *Assignment*. When the *Assignment* reaches that state (e.g., ‘completed’) she is automatically notified.

The third problem (C) deals with *intrinsic* activities whose outcome requires certain *extrinsic* follow-up activities. As illustrated in Section 2, the modification of a source code artifact that belongs to the interface of a component is the target. Such changes often require adapting integration tests or architecture documents. Dependent adaptations usually do not appear in the workflows belonging to SE processes and are thus *extrinsic* workflows. The given example illustrates the case for the follow-up actions regarding the tests as shown in Fig. 5. It shows two defined project areas ‘Implementation’ and ‘Test’. There is a *PotentialImpact* configured for relating technical issues from ‘Implementation’ to ‘Test’. For the implementation *area*, there are different modules with different packages. Modules x and y also appear in the test *area* and relate to the counterparts in the implementation *area* as indicated by the curved lines. Developer 2 is responsible for the tests of Modules x and y. Assume now that Developer 1 changes a class belonging to Package b, indicated by the change activity.

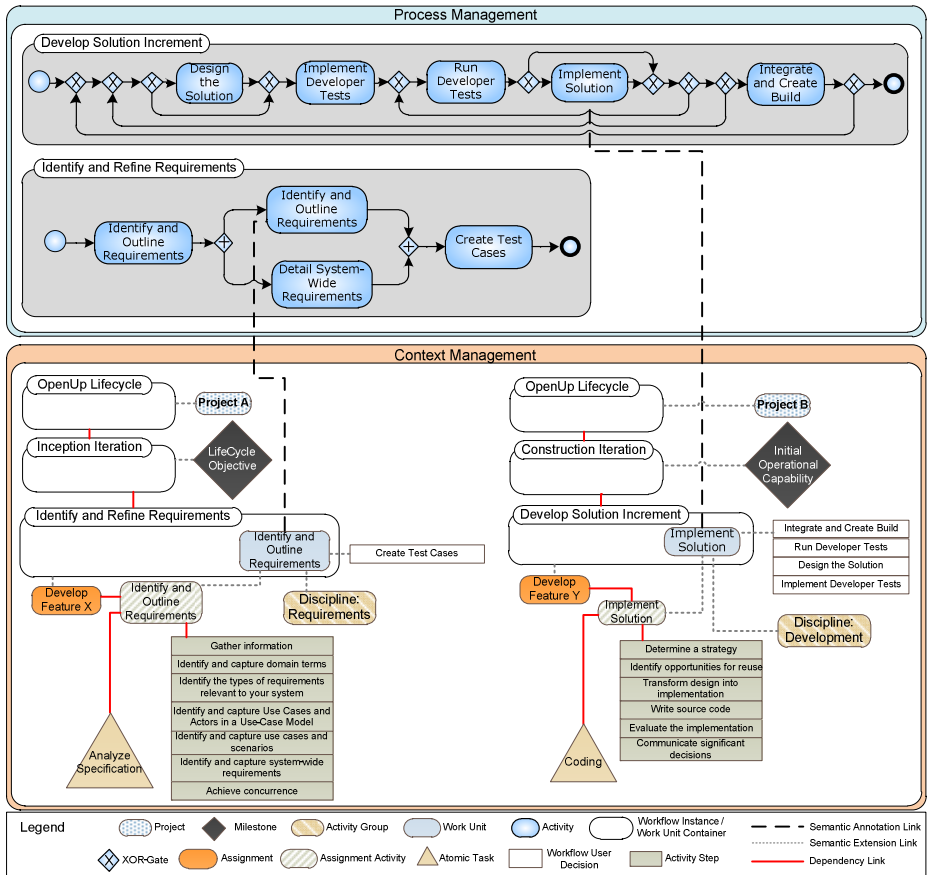


Fig. 4. Navigability information example

The information about the component, the kind of change applied to it, and the user ID of the responsible person are forwarded via an event to the *Process Management* module, which starts a workflow to govern the desired activities for the respective user. This workflow can be based on a predefined workflow template or be custom-built from a problem-oriented declarative definition as described in [7]. When a task of that workflow becomes available to a user, an event is automatically distributed to CoSEEEK’s web GUI shown in Fig. 6.

All tasks are shown at the bottom of the GUI. In order to avoid subjecting a user to information overload, only the current task and the next upcoming task proposed by the system are shown. The user may change the selection of the next upcoming task via a dropdown list. In this example, the current task is “Implement Tests” from an *intrinsic* workflow, while the next upcoming task is “Check Component due to Interface Change” from an *extrinsic* workflow. The upper part of the GUI contains information provided by the framework. Among other things, it can be used to display additional task information and notifications about components for which change notification is configured. This example shows the notification about the change of an artifact.

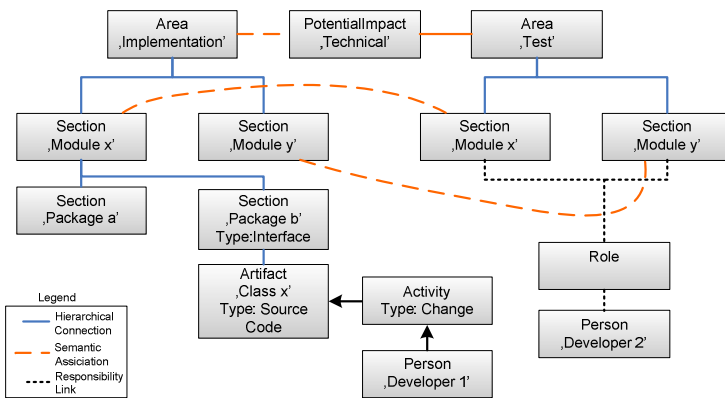


Fig. 5. Active coordination support example

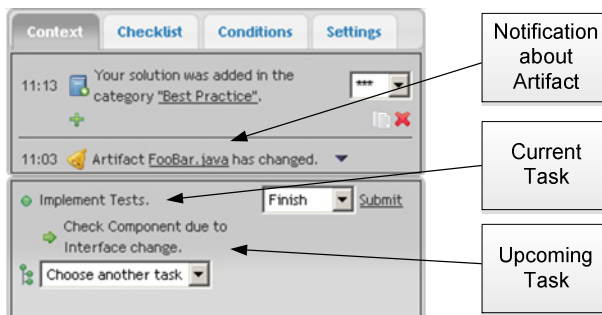


Fig. 6. CoSEEEK Web GUI



In summary, the resolution provides collaboration capabilities via coordination of *extrinsic* and *intrinsic* workflows in a PAIS and the availability and use of context information via semantic technology. Activities that are often omitted and not modeled in PCSEEs are explicitly modeled and automatically coordinated via CoSEEEK. Additional support is provided for software engineers working in multi-project environments by making navigability information available and fostering situational awareness. Finally, automatic information provision can keep users updated on artifacts states or other new events in the project.

## 5 Modeling Effort

Additional modeling effort is imposed by the approach. The processes are modeled not only in the PAIS but also in the ontology. Configuration is required for how various follow-up actions should be treated. To keep the effort reasonable, some default functions and definitions are provided in the framework. The semantic enhancements to process management (*WorkUnitContainers* and *WorkUnits*) are generated automatically from the workflow templates of the *Process Management* module. To gain an awareness of project artifacts, scans are conducted on specified folders. Since the system is aware of SE tools via sensors, it becomes aware of all processed and new artifacts, and the information is acquired on the fly. An initial set of *ProjectComponents* is provided and the structure of certain *Areas* can be imported, e.g., from a folder structure or a source code package structure. Examples include the *Areas* ‘Implementation’ and ‘Test’: the system can automatically read the package structure and thus import references to all artifacts into the ontology that are hierarchically organized under various *Sections* that are created from the different packages in the source code. The names of the packages can be automatically matched to those to which they may relate. For instance, relations between ‘Test’ packages and ‘Implementation’ packages can be automatically established.

## 6 Related Work

With regard to PCSEEs, [13] describe SOA-based extensible and self-contained sub-processes that are aligned to each task. A dynamic runtime selection is made depending on the context of the particular work instance. OPEN [26] is a CORBA-based PCSEE that addressed business, quality, model, and reuse issues. DiME [27] provides a proprietary, integrated, collaborative environment for managing product definition, development, and delivery processes and information. CASDE [28] and CoolDev [29] utilize activity theory for building an environment supporting collaborative work. CASDE features a role-based awareness module managing mutual awareness of different roles. CoolDev is a plug-in for the Eclipse IDE that manages activities performed with other plug-ins in the context of global cooperative activities. CAISE [30] is a collaborative SE framework with the ability to integrate SE tools. CAISE supports the development of new SE tools based on collaboration patterns.

An industry approach for collaborative development is provided by the IBM Jazz / Rational Team Concert products [31]. Jazz offers an infrastructure for distributed

development including the technical basis for integration of various clients as well as data and services. It enables comprehensive project, bug, and configuration management as well as event notifications, traceability, and other software development related tasks. Team Concert is a collaborative software development environment built on Jazz technology utilizing its capabilities to provide an integrated solution for software configuration management, work item management, and build management with additional features like customizable dashboards, milestone tracking, or process templates for common processes.

In contrast, CoSEEEK offers a combination of features not found in the aforementioned approaches: workflow guidance is not only offered for activities contained in development processes (*intrinsic*), but also for *extrinsic* activities, which are not explicitly modeled within those processes. The holistic combination of all project *areas* in conjunction with semantic technology also enables the framework to provide intelligent decisions and thus a higher level of automation. The tight integration of PAIS technology with context knowledge not only enables the distribution of information, but also the automated support and governance of activities in adapted workflows.

Modeling SE processes in semantic technologies can enhance reuse and leverage available tooling, as shown by [32]. [33] used an ontology for CMMI-SW assessments, and [34] used ontologies for the Software Engineering Body of Knowledge (SWEBOK). CoSEEEK leverages semantic usage for real-time contextual-awareness in SEEs to improve SE workflows and collaboration and for supporting *navigability* and situational-awareness. The main differentiation criterion to other approaches utilizing ontologies for collaboration is the holistic integration of all project *areas* to foster synergies, and in having collaboration not be the sole focus of the framework (e.g., software quality assurance is adaptively integrated as described in [11]). Other approaches have collaboration via ontologies as their focus [35][36]. [35] presents a workflow-centric collaboration system whereby the main component is an ontology repository with ontologies of different abstraction levels. The process model is based on enhanced Petri nets and thus lacks complementary support for dynamic adaptability. [36] presents an Ontology for Contextual Collaborative Applications (OCCA) that provides a generic semantic model specialized for distributed, heterogeneous, and context-aware environments. In contrast to these approaches, CoSEEEK utilizes querying and reasoning capabilities over an ontology and integrates these with process management to support automated dynamic process governance.

## 7 Conclusions

The high degree of dynamic collaboration in SE raises challenges for the automated support of process awareness and guidance in SEEs. Currently, SEEs lack contextual information and integration, especially with regard to adaptive collaboration and workflows. The presented CoSEEEK approach extends adaptive PAIS with semantic web technologies and advanced event processing techniques. CoSEEEK explicitly models and manages both *intrinsic* and *extrinsic* activities. These are coordinated, and the automatic initiation and distribution of activities can be individually configured. A dynamic information distribution strategy enables related components to be associated

even if no direct relations between the source component and the target component exist. The person being responsible for a component can also be determined if no direct responsibility is defined. The procedure requires neither rigidly predefined information channels nor relies on comprehensive and fine-grained predefined information on relating artifacts or responsible persons. The configuration effort to enable automated coordination is reduced by the ability to automatically import needed information and via the inference and reasoning capabilities.

As the automatic initiation of new follow-on activities is neither necessary nor desired in all cases, the system also provides passive collaboration support abilities. These comprise automatic user notifications on various events in a project. Both general pre-configured notifications and user-configured personalized notifications are possible.

*Extrinsic* activities that have hitherto typically been excluded from modeling are now guided by workflows. These capabilities enable the integration of general process models with concrete activities even if they are *extrinsic* to a particular SE process. Support for situational awareness and *navigability* becomes vital as collaborations become more complex. Additional process navigability information can be automatically provided by CoSEEEK. Individuals working in multi-project environments can profit from this information since it supports them operationally, e.g., when they are switching contexts by providing all relevant information for the current activity.

The presented scenario demonstrated a situation where improved coordination and situational awareness were supported while providing process guidance and *navigability* for collaborating software engineers, enhancing process quality.

Automated support for coordinated collaborative software engineering, with its human interactions and continuously changing tool and process environment, will remain a challenge. Further research potential lies in the aggregation and utilization of available contextual information to increase process effectiveness and efficiency. Future work will investigate industrial usage in production environments with our project partners. For efficiency, a planned feature will aggregate related tasks and, when a predefined threshold is reached, trigger a workflow instance with the cumulated task information. More complex task treatments can also be designated: e.g., in an agile project, emergent uncompleted tasks can be collected and stored in a backlog to inform team members at the beginning of the next iteration. A GUI that enables the easy definition of rules for the automatic initiation of follow-up activities is planned. It will also support the easy registration for notifications on state changes of activities or artifacts or other events.

**Acknowledgements.** This work was sponsored by BMBF (Federal Ministry of Education and Research) of the Federal Republic of Germany under Contract No. 17N4809.

## References

1. Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors, and implications. *IEEE Transactions on Systems, Man, and Cybernetics* 38(3), 280–291 (2008)

2. Gruhn, V.: Process-centered software engineering environments, a brief history and future challenges. *Annals of Software Engineering* 14(1), 363–382 (2002)
3. Rausch, A., Bartelt, C., Ternité, T., Kuhrmann, M.: The V-Modell XT Applied–Model-Driven and Document-Centric Development. In: Proc. 3rd World Congress for Software Quality, vol. III, pp. 131–138 (2005)
4. OpenUP (2011), <http://epf.eclipse.org/wikis/openup/>
5. Oberhauser, R., Schmidt, R.: Towards a Holistic Integration of Software Lifecycle Processes using the Semantic Web. In: Proc. 2nd Int. Conf. on Software and Data Technologies, vol. 3, pp. 137–144 (2007)
6. Oberhauser, R.: Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments. In: *Semantic Web*, pp. 157–179. In-Tech, Vienna (2010)
7. Grambow, G., Oberhauser, R., Reichert, M.: Semantic workflow adaption in support of workflow diversity. In: Proc. 4th Int’l Conf. on Advances in Semantic Processing, pp. 158–165 (2010)
8. Grambow, G., Oberhauser, R., Reichert, M.: Towards a Workflow Language for Software Engineering. In: Proc. 10th IASTED Conference on Software Engineering (2011)
9. Grambow, G., Oberhauser, R.: Towards Automated Context-Aware Selection of Software Quality Measures. In: Proc. 5th Intl. Conf. on Software Engineering Advances, pp. 347–352 (2010)
10. Grambow, G., Oberhauser, R., Reichert, M.: Contextual Injection of Quality Measures into Software Engineering Processes. *Int’l Journal on Advances in Software* 4(1 & 2), 76–99 (2011)
11. Grambow, G., Oberhauser, R., Reichert, M.: Employing Semantically Driven Adaptation for Amalgamating Software Quality Assurance with Process Management. In: Proc. 2nd Int’l. Conf. on Adaptive and Self-adaptive Systems and Applications, pp. 58–67 (2010)
12. Grambow, G., Oberhauser, R., Reichert, M.: Towards Automatic Process-aware Coordination in Collaborative Software Engineering. In: Proc. 6th International Conference on Software and Data Technologies, pp. 5–14 (2011)
13. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2006, Part I. LNCS, vol. 4275, pp. 291–308. Springer, Heidelberg (2006)
14. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science-Research and Development* 23(2), 81–97 (2009)
15. Weber, B., Sadiq, S., Reichert, M.: Beyond rigidity–dynamic process lifecycle support. *Computer Science-Research and Development* 23(2), 47–65 (2009)
16. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in Process-Aware Information Systems. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 115–135. Springer, Heidelberg (2009)
17. Gasevic, D., Djuric, D., Devedzic, V.: Model driven architecture and ontology development. Springer (2006)
18. McGuinness, D.L., Van Harmelen, F.: OWL web ontology language overview. W3C recommendation (2004)
19. World Wide Web Consortium: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (2004)
20. Prud’hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C WD 4 (2006)
21. McBride, B.: Jena: A semantic web toolkit. *IEEE Internet Computing* 6(6), 55–59 (2002)

22. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. *Web Semantics: Science, Services and Agents on the World Wide Web* 3(1), 41–60 (2005)
23. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2), 51–53 (2007)
24. Johnson, P.M.: Requirement and design trade-offs in Hackstat: An in-process software engineering measurement and analysis system. In: *Proc. 1st Int. Symp. on Empirical Software Engineering and Measurement*, pp. 81–90 (2007)
25. Luckham, D.C.: *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
26. Henderson-Sellers, B.: Process metamodelling and process construction: examples using the OPEN Process Framework (OPF). *Annals of Software Engineering* 14(1), 341–362 (2002)
27. Koenig, S.: Integrated process and knowledge management for product definition, development and delivery. In: *Proc. IEEE International Conference on Software-Science, Technology & Engineering*, p. 133 (2003)
28. Jiang, T., Ying, J., Wu, M.: CASDE: An Environment for Collaborative Software Development. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) *CSCWD. LNCS*, vol. 4402, pp. 367–376. Springer, Heidelberg (2007)
29. Lewandowski, A., Bourguin, G.: Enhancing Support for Collaboration in Software Development Environments. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) *CSCWD. LNCS*, vol. 4402, pp. 160–169. Springer, Heidelberg (2007)
30. Cook, C., Churcher, N., Irwin, W.: Towards synchronous collaborative software engineering. In: *Proc. 11th Asia-Pacific Software Engineering Conference*, pp. 230–239 (2004)
31. IBM Jazz, <http://www.jazz.net>
32. Liao, L., Qu, Y., Leung, H.: A software process ontology and its application. In: *Proc. ISWC 2005 Workshop on Semantic Web Enabled Software Engineering*, pp. 6–10 (2005)
33. Soydan, G.H., Kokar, M.: An OWL ontology for representing the CMMI-SW model. In: *Proc. 2nd Int'l Workshop on Semantic Web Enabled Software Engineering*, pp. 1–14 (2006)
34. Calero, C., Ruiz, F., Piattini, M.: *Ontologies for software engineering and software technology*. Springer-Verlag New York Inc. (2006)
35. Yao, Z., Liu, S., Han, L., Ramana Reddy, Y.V., Yu, J., Liu, Y., Zhang, C., Zheng, Z.: An Ontology Based Workflow Centric Collaboration System. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) *CSCWD. LNCS*, vol. 4402, pp. 689–698. Springer, Heidelberg (2007)
36. Wang, G., Jiang, J., Shi, M.: Modeling Contexts in Collaborative Environment: A New Approach. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) *CSCWD. LNCS*, vol. 4402, pp. 23–32. Springer, Heidelberg (2007)

**Part II**  
**Software Engineering**

# Hybrid Debugging of Java Programs

Christian Hermanns and Herbert Kuchen

Institute of Information Systems, University of Münster  
Leonardo-Campus 3, Münster, Germany

`chr.hermanns@gmx.de`, `kuchen@uni-muenster.de`

**Abstract.** Until today the most common technique to debug Java programs is trace debugging. In this work we present two different debugging approaches for Java: declarative debugging, which has its origins in the area of functional and logic programming, and omniscient debugging, which is basically an extension of trace debugging. To benefit from the advantages of both techniques we have integrated them into a single hybrid debugger called JHyde. We use JHyde to debug an erroneous merge sort algorithm and mention important aspects of its implementation. Furthermore, we show that the efficiency of the declarative debugging method can be significantly improved by a new debugging strategy.

**Keywords:** Java, Debugging, Hybrid, Declarative, Omniscient, Coverage criteria, Divide and query.

## 1 Introduction

Debugging is a complex and time-consuming task of the software development process. Software developers spend a considerable amount of time debugging, trying to locate bugs in software programs. Reducing the time and effort required to detect a bug can greatly increase the efficiency and productivity of software development [1].

No matter what debugging method or tools are used to locate a bug, the general debugging process looks as follows (cf. [2]). At first the user notices a *failure*, i.e. an externally observable error in the program behaviour. The failure is caused by an *infection*, that is a deviation of the actual program state from the intended program state. The user's task is to locate the *bug*, i.e. the defect statement(s) in the debuggee program which caused the *initial* infection. In many cases the initial infection does not directly result in an observable program failure. Instead, it is propagated into later program states causing further infections which eventually cause a program failure. As a consequence, the user has to reason backwards in the execution history from the failure to the bug.

Until today, the most common debuggers used for object-oriented programming languages like Java are program tracers which track the execution of a debuggee program in a step-by-step manner. A trace debugger works on a low level of abstraction. Starting at a breaking point which predates the bug in the execution history the user has to inspect the sequence of statements executed by the debuggee program. The debugging process ends when the user encounters a buggy statement, i.e. bug, which produces the initial infection. While this method works well for a narrow search space, it can be very annoying and time-consuming if there aren't any clues about the position of the bug.

To minimize debugging effort the starting point should be as close to the bug as possible. However, the defect and its position in the execution history are unknown. When selecting a starting point for program tracing, we risk to select a point which lies chronologically either far before or even after the bug. In the first case we will have to inspect a lot of instructions and in the second case we will have to restart debugging with an earlier starting point.

Another severe drawback of trace debuggers is the fact that they are only capable of executing the debuggee program forward in time, while the reasoning from the program failure to the program defect naturally happens backwards in time. *Omniscient debugging* [3], an extension of the trace debugging technique, addresses these shortcomings. The idea is to record the entire execution history of a debuggee program, making it possible to inspect the execution history back and forth in time. Nevertheless, even omniscient debugging takes place on a relatively low level of abstractions where the user still has to check single statements for validity.

A debugging method which abstracts from the details of implementation is declarative debugging. The method was developed by E. Y. Shapiro [4] for the Logic Programming paradigm. It was later applied to other declarative programming paradigms such as functional [5] and functional logic [6] programming. A declarative debugger asks the user questions about the validity of the program execution and infers the location of the bug from the user's answers. The precision by which the location of a bug can be detected depends on the level of abstraction of the declarative debugger. The more abstract the questions, the less precise the location of the bug can be determined. For example, a declarative debugger asking questions about the validity of method calls is able to detect a method containing a defect, but it cannot tell which statement inside the method is defect.

In this paper we present JHyde (Java Hybrid Debugger), a debugger which implements a hybrid debugging method for the object-oriented language Java. JHyde combines omniscient and declarative debugging into one debugging tool. In general, our hybrid debugging method looks as follows. At first the declarative debugging method is used to locate the method call causing the initial infection. After that omniscient debugging functionality is used to find the buggy statement inside the method. Thus, the user can at first, during the declarative debugging process, concentrate on "what" the method calls do and can ignore "how" the calls work internally. Afterwards, the identified call causing the initial infection provides a narrow search space for the omniscient debugging method. That way, our hybrid debugging technique abstracts from the implementation details during declarative debugging but is yet capable of identifying the exact buggy statement by means of omniscient debugging.

JHyde is based on a declarative debugger for the Java programming language which we developed in former work [7]. The main contribution of this paper is to show how omniscient and declarative debugging can be combined. We present JHyde which is available for download [8] and show important aspects of implementation. Furthermore, we have developed a declarative debugging strategy based on coverage information which reduces the number of method calls a user has to inspect. Tests show that this strategy saves up to 40 percent of the debugging effort.



The rest of the paper is organized as follows: In the next section we present our hybrid debugging technique for Java programs. Section 3 describes the user interface of JHyde, a prototypical plugin for the Eclipse IDE [9] which implements the hybrid debugging technique. To show how our debugger behaves in practice, we debug a sample program using JHyde in Section 4. In Section 5 we provide some implementation details. Section 6 presents some test results which proof the usefulness of the coverage-based search strategy. In section 7 we discuss how our paper is related to other works in this field. This paper ends with section 8, where we conclude and point out future work.

## 2 Hybrid Debugging Technique

### 2.1 Overview

As indicated before the general idea of our hybrid debugging technique is to combine elements of omniscient debugging and declarative debugging. Declarative debugging has the benefit that it works on a higher level of abstraction relieving the user from the task to inspect implementation details. Nevertheless, it is somewhat imprecise. For example, a declarative debugger for Java Programs is only able to identify the method containing the bug, but not the buggy statements inside the method.

This is where the omniscient debugging functionality of our debugger comes into play. Once the method call causing the initial infection is found using declarative debugging, the user can resort to the omniscient debugging. By tracking the execution of the method call in a step-by-step manner back and forth in time, the user can exactly determine the buggy statement(s).

### 2.2 Declarative Debugging

Declarative debugging requires the execution of the debuggee program to be represented by a computation tree (CT). To enable the declarative debugging of Java programs we have defined a suitable CT data structure in [7]. In our data structure each node of the CT will contain information about a particular method call of the computation. Let  $a$  be a node of the CT containing information about a call of method  $m_a$ . Child nodes of  $a$  correspond to calls that have been executed during the computation of the result of  $a$ .

During the declarative debugging process the user will be asked to classify nodes of the CT. A node has to be classified as *valid* if the represented method call produced a valid result, i.e. the return value including all side effects and as *invalid* if the represented method call produced an invalid result. Furthermore, a method can be classified as *trusted* if we know that it does not contain any defects. In this case every node representing a call of the trusted method is automatically trusted. A trusted node will not cause an infection, but it can have untrusted subnodes which could infect the program state.

The classification of the nodes is used to reduce the set of suspicious nodes, i.e. the set of nodes that can be buggy. We define  $N$  as the set of all nodes of the CT and  $A_i \subseteq N$  as the set of suspicious nodes after the  $i$ -th user answer. At the beginning  $A_0$  contains

all nodes of the CT, i.e.  $A_0 = N$ . Furthermore, let  $S_a \subseteq N$  be the set of all nodes of the subtree rooted at node  $a$  in the CT. If node  $a \in A_{i-1}$  is classified, the suspicious set  $A_{i-1}$  can be reduced by the following rules. If node  $a$  is classified as valid, all its subnodes are automatically valid, i.e.  $A_i = A_{i-1} \setminus S_a$ . If a method call is classified as invalid either the call itself or any of its sub calls must be the buggy method call we are looking for. Hence, the debugger will continue the search in the subtree of this method call and we get  $A_i = S_a$ . If a method call is classified as trusted, then all calls of the same method will also become trusted. JHyde will not ask any questions about trusted method calls. However, a trusted method call can have unclassified child calls. These calls must be considered in the further debugging process. Thus,  $A_i = A_{i-1} \setminus \{b \in A_{i-1} \mid m_b = m_a\}$ . A method call's state can be set to unclassified if the user is not sure about the classification and wants to categorize it later.

If at least one method call has been classified as invalid, the size of the suspicious set is eventually reduced to 1, i.e.  $\exists z \in \mathbb{N} : |A_z| = 1$ . The only node  $n$  remaining in  $A_z$  is an invalid node with no invalid children. The call represented by the buggy node produced an invalid result while its inputs, i.e. the arguments provided by the parent call and the return values of the child calls, are all valid. We call this node the *buggy node*. The method call associated to the buggy node must contain a bug we are looking for.

A method call in an object-oriented language can produce side effects which are considered to be part of its result. These side effects must be taken into account during classification of a method call. Hence, the data presented to the user at each node of our CT will be:

- The fully qualified name of the method being called.
- All local variables of the method call, i.e. the arguments of the call and all locally defined variables. In case of a call to a non-static method the “this” reference to the object whose method is being called is also considered as an argument. For each local variable an entry and an exit value will be stored. The entry value is the variable's value at the beginning of the method call and can be regarded as part of the method's input. The exit value is the value at the end of the call and belongs to the method's result.
- The fields of referenced objects. For any object directly or indirectly referenced by a local variable we need to know the entry and exit values of its fields w.r.t. the considered method call. An array is considered as a special type of object with all its fields of the same type. Note that the fields of referenced objects can in turn reference further objects, spanning a *graph of referenced objects*. We call this graph the state space which is accessible by the respective method call. The accessible state spaces spanned by the entry and exit values are part of the method's input and the method's result, respectively.
- Additionally, the entry and exit values of static class fields should be available because they can be part of the methods input and output as well.
- Finally, we need the return value.

The usability of the debugger will depend on a compact and clear representation of the relevant (changed) information.

### 2.3 Omniscient Debugging

Omniscient debugging is an extension of trace debugging. During trace debugging the user directly observes the effects of executed statements on the program state. To find a defect the first statement in the execution history which changed the program state from valid to infected must be identified.

A severe drawback of trace debugging is that we have to trace statements executed *before* the defect, while a defect effects the program states which occur *after* its execution. Before the defect is executed the program state is sane which is why we do not have facts we can use to reason about the location of the bug. Hence, the odds are that we miss the defect and have to start over again.

Omniscient debugging addresses these drawbacks as it allows to track the execution of a program back and forth in time. If we miss the execution of a defect, we can now simply step back in the execution history and do not have to restart our debugger. More importantly, we can now reason backwards from the program failure to the initial infection, following the infected program states to the bug. This is a great improvement over ordinary trace debugging.

It is obvious that these benefits come with costs. Omniscient debugging requires us to record the entire execution history of the debuggee program. The amount of data we have to record can be huge, especially for large and complex programs. But we have to record most of the execution information anyway for declarative debugging. Therefore, in the case of our hybrid debugger we get the benefits of omniscient debugging almost for free.

### 2.4 Coverage Based Navigation Strategy

How do the individual debugging efforts develop if the size of the recorded execution history grows? On average, we expect the omniscient debugging effort to remain almost constant as we are looking for the defect in a single method call, while we expect the declarative debugging effort to grow as we are facing a growing size of the CT. Especially when debugging complex programs the number of executed method calls can become huge. Thus, an efficient strategy for the declarative debugging process can greatly reduce the overall debugging effort.

One possible declarative debugging strategy is based on the divide and query (D&Q) strategy developed by Shapiro [4]. The idea of D&Q is to halve the set of suspicious nodes  $A_i$  with every user answer. Let  $w_i : N \rightarrow \mathbb{N}$ ,  $w_i(a) = |S_a \cap A_i|$  return the weight of a node  $a$  in the suspicious set after  $i$  user answers and let  $W_i = |A_i|$  be the weight of the suspicious set.

D&Q will select the node  $a$  whose weight  $w_i(a)$  is closest to half of the weight of the suspicious set, i.e.  $W_i/2$ , for classification. After the user answer, the weight of the remaining suspicious set  $W_{i+1}$  will be  $W_i - w_i(a)$  or  $w_i(a)$  if the answer was “valid” or “invalid”, respectively. Especially for big and unbalanced CTs D&Q performs better than a top-down strategy. In the worst case only  $O(\log n)$ ,  $n = |N|$  user answers are required.

D&Q assumes that every node of the CT has the same probability to contain a bug. In real applications this is rarely the case as method calls can have different complexities

which result in different probabilities to contain a bug. We can improve the performance of D&Q if we take varying error probabilities into account. For this, we define a coverage entity to be either a control-flow or data-flow element whose coverage is monitored during program execution. To estimate the complexity of a method call we assume that the complexity of a method call increases with the number of entities that are covered.

Let  $E$  be the set of all entities covered during the debuggee execution,  $E_a \subseteq E$  the covered entities of node  $a$ , and  $V_i \subset N$  the subset of nodes classified as valid after  $i$  user answers. Furthermore, the function  $v_i : E \rightarrow \mathbb{N}_{\neq}$  returns the number of valid classifications of a coverage entity after  $i$  user answers, i.e.  $v_i(e) = |\{a \in V_i : e \in E_a\}|$ . We compute the coverage-based weight of a node after  $i$  answers by  $cbw_i : N \rightarrow \mathbb{R}^+$  as follows:

$$cbw_i(a) = \sum_{b \in (S_a \cap A_i)} \sum_{e \in E_b} g(v_i(e)).$$

Where the function  $g : \mathbb{N}_0 \rightarrow \mathbb{R}^+$  returns the weight of a single entity  $e$  based on the number of valid classifications of  $e$ . Thus, the weight of a node  $a$  is the weight of all covered entities of all nodes which are element of the subtree rooted at  $a$  and the suspicious set  $A_i$ . In order to reduce the contribution of entities which have been classified as valid, the function  $g$  should be monotonically decreasing.

One possible definition of  $g$  is:  $g(x) = p^x$ , with  $p \in [0, 1]$ . In this case the contribution to the coverage-based weight of an entity  $e$  which has not been classified as valid yet, i.e.  $v_i(e) = 0$ , is  $p^0 = 1$ . The contribution of  $e$  is exponentially reduced as the number of valid classification of  $e$  increases. For example, if  $p = 1/2$  the contribution is halved for any additional valid classification.

Let  $CBW_i = \sum_{b \in A_i} \sum_{e \in E_b} g(v_i(e))$  be the coverage-based weight of the suspicious set  $A_i$ . Similar to D&Q, our coverage-based D&Q will select the node  $a$  from  $A_i$  whose coverage-based weight  $cbw_i(a)$  is nearest to half of the weight of the suspicious set,  $CBW_i/2$ , for the next classification. This way we try to halve the coverage-based weight of the suspicious set,  $CBW_i$ , with every user answer. We have implemented coverage-based D&Q using coverage of the edges of the control flow graph (CFG) and def-use chain coverage.

In a CFG each edge represents the flow of control between two nodes, i.e. basic blocks. A basic block is a sequence of code which has only one entry (no code within the sequence is destination of a jump) and one exit point (no jump statements within the sequence).

A def-use chain is a triple of a variable, a statement where the variable's value is computed, i.e. defined, and a statement where the variable's value is used. Moreover, the variable's value must not change between the definition and the use.

Both metrics, edge coverage of the CFG and def-use chain coverage, are common in the area of glass-box testing [10], where they are used to measure the coverage of test cases. The goal is to generate a minimal set of test cases which completely covers a tested component w.r.t. to a certain coverage criterion. It is assumed that the probability of a component containing a defect is almost zero, if all test cases pass and their coverage is complete. Similarly, we employ coverage information during the debugging

process to determine which parts of the program have the highest probability to contain a defect.

Please note, that we have employed edge and def-use chain coverage to reduce the number of questions during the declarative debugging process before [7]. In our earlier works we *avoid* questions about method calls whose covered entities are completely covered by method calls which are classified as valid. In other words, if  $E_a \subseteq \bigcup_{b \in V_i} E_b$  holds,  $a$  will be automatically assumed to be valid. This approach significantly reduces the debugging effort, but it has a major drawback: if our debugger infers a wrong classification, this strategy might identify a wrong buggy method call. Our new approach does not have these shortcomings because we do not infer any answers but simply change the order of questions. Thus, our method always returns the correct result.

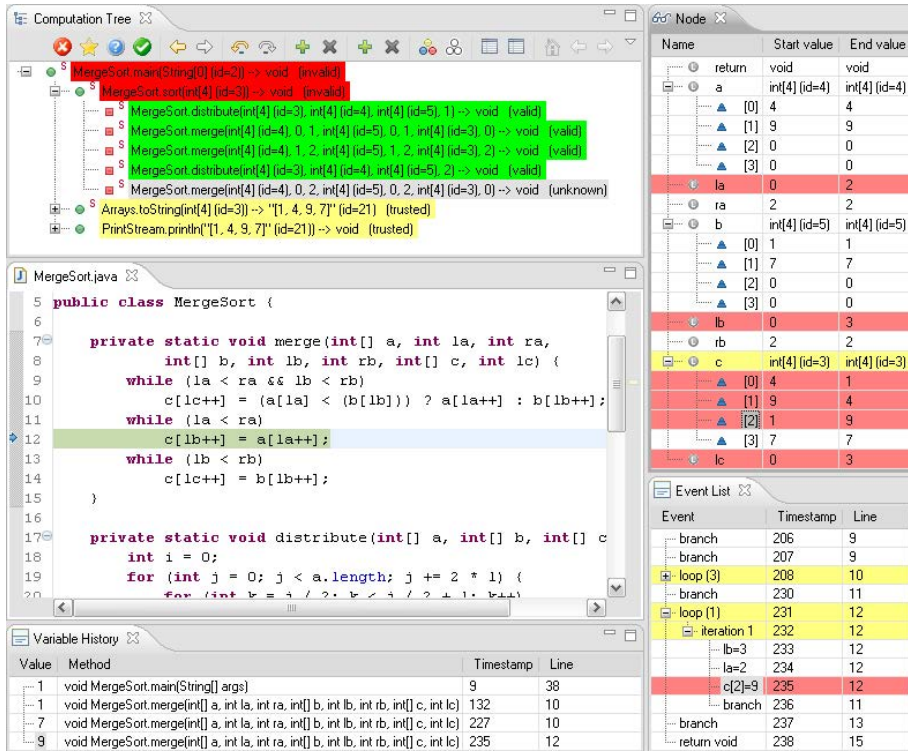
### 3 User Interface

The task of the JHyde front-end is to present the execution data of a program run in a structured and accessible way. The interface must be easy to use and understand to find bugs quickly. The JHyde front-end is an Eclipse [9] plugin. The plugin enables the user to conduct the hybrid debugging process for Java programs. When started JHyde executes the selected program and records its entire execution history. The search for the bug starts when the debuggee's execution is finished. The execution data is presented to the user in four different views shown in Figure 1. Each view is designed to present a different aspect of the execution data.

The Computation Tree View shows the CT of the debuggee program. Via the toolbar the user can classify the currently selected method call as valid, invalid, trusted or unclassified (unknown). According to its classification a method call is highlighted by green (valid), red (invalid), yellow (trusted), or white (unclassified) background color. After classification the debugger selects a new, yet unclassified method call. JHyde supports four different navigation strategies: top-down, D&Q, D&Q by edge coverage of the CFG, and D&Q by def-use chain coverage.

In order to classify a method call we need information about the inputs and outputs of a method call. This information is shown in the Node View of JHyde. As described in Section 2.2 we need all local variables, i.e. the arguments of the call, all locally defined variables, and the “this” reference to the object whose method is being called if the method is not static. Furthermore, we need all static variables. The Node View will always show these variables for the method call selected in the CT View. The Node View has a timestamp interval. The lower and the upper bound of the interval are set to the entry and exit timestamp of the method call selected in the Computation Tree View, respectively. For each variable its value at the lower bound (old value) and its value at the upper bound (new value) are shown. If the variable type is not a primitive but a reference type and at least one of the variable's values is not null then the fields of the referenced object will be shown in the variable's subtree.

To validate the selected method call a user can browse the old and new values of all relevant variables. If the method call's actual effect matches its intended effect the call is classified as valid. If actual and desired effect do not match, it must be classified as invalid. Validating the state change of the affected variables is a complex and



**Fig. 1.** Screenshot of the JHyde plugin which consists of the Computation Tree View, the Node View, Event List View, and the Variable History View, which help to navigate the execution data of a debuggee program. The views represent the execution of the MergeSort program shown in Listing 1.1

time-consuming task. In the Node View every variable whose value changed during the selected method call is marked by a certain background color. A variable whose value changed is marked red, e.g. `la`, `lb`, `c[0]...c[2]`, and `lc`. A variable whose value did not change can reference an object whose variables, i.e. fields, have changed. These variables are marked yellow, e.g. `c`. The highlighting of the state changes makes it much easier to evaluate a method call. The user can directly navigate to the changes and check their correctness.

The Event List View and the Variable History View are used for automatic debugging. The Event List View shows a chronological list of all events that occurred during the method call selected in the CT View. It is used to track the changes step-by-step, back and forth in time. If a user selects a specific event from the list, the source code line which produced this event is shown in the source code editor. Furthermore, the upper bound of the timestamp interval of the Node View is set to the timestamp of the selected event. This has the following effect: In the Node View the new value of each variable now shows the value the respective variable had right after the execution of the selected event. Hence, the Node View now shows all state changes which happened between the execution of the first event, i.e. the beginning of the method call, and the event selected

in the Event List View. This gives the user a summary of all state changes performed by the events of the method call which precede the selected event. Thus, the user can conveniently check if the program state is still valid at the selected event.

If we want to lookup the event which changed a variable's value in the Event List View, we can select the desired variable in the Node View. Every event which changed the value of the selected variable will now be highlighted in the Event List View. For example, the array field `c[2]` is selected in the Node View in Figure 4.1. Thus, the events which changed `c[2]` are highlighted in the Event List View. If a changing event is located in the subtree of an event it is marked with yellow (light gray) background color, e.g. timestamps 208, 231, and 232. If the event itself changed the value, it is marked red (dark gray), e.g. timestamp 235. The interaction between the Event List View and the Node View offers some nice and handy features which facilitate the search for a defect in the method call's execution history.

The second view which can be employed to search for the program bug is the Variable History View. It displays a chronological list of all events of the complete execution history that changed a specific variable. To assign a variable to the Event List View we can right click on any variable in the Node View and select "display in History View". For example, the History View in Figure 4.2 shows the value history of `c[2]`. With the help of the Variable History View we can check the value history of a Variable for invalid values. If we have identified an undesired value, we can select the corresponding event, e.g. the event with timestamp 235 which changed the value of `c[2]` to 9. Clicking an event in the History View will cause an update of the other views of JHyde. In the CT View the method call which executed the selected event is selected, the Event List View shows the events of the method call selected in the CT View. Furthermore, the examined event, i.e. the event with timestamp 235, is selected in the Event List View. The Node View will show the value changes of all variables of the corresponding method call from the beginning of the call to the timestamp of the selected event. Figure 4.3 shows the state of all JHyde views after the event with timestamp 235 has been selected in the History View. Hence, the Variable History View is a useful utility to navigate the execution data. By selecting an event in the History View we can directly jump to its location in the execution history. Thus, the History View allows us to reason backwards in the execution history. If we notice a program failure after the execution of the debuggee program, we can use the variable whose value caused the failure to trace back to the point in the execution history where a wrong value was assigned to the variable. If the wrong value was assigned due to a defect in the program, we have found the bug. Otherwise the wrong value must result from another variable which has a wrong value at the given time in the execution history. We can continue the backward reasoning process with this new variable. By this process we will eventually find the bug causing the program failure.

## 4 Debugging Session Example

To demonstrate how JHyde is used, we will show how to debug the Java implementation of the well-known merge sort algorithm [11] presented in Listing 4.1. We assume that our debugger uses a top-down declarative debugging strategy.

---

```
5 public class MergeSort {
6
7     private static void merge(int[] a, int la,
8         int ra, int[] b, int lb, int rb, int[] c,
9         int lc) {
10        while (la < ra && lb < rb)
11            c[lc++] = (a[la] < b[lb]) ? a[la++] :
12                b[lb++];
13        while (la < ra)
14            c[lb++] = a[la++]; // must be: c[lc++]
15            =a[la++];
16        while (lb < rb)
17            c[lc++] = b[lb++];
18    }
19
20    private static void distribute(int[] a, int[] b,
21        int[] c, int l) {
22        int i = 0;
23        for (int j = 0; j < a.length; j += 2 * l) {
24            for (int k = j / 2; k < j / 2 + l; k++)
25                b[k] = a[i++];
26            for (int k = j / 2; k < j / 2 + l; k++)
27                c[k] = a[i++];
28        }
29    }
30
31    public static void sort(int[] a) {
32        int[] b = new int[a.length];
33        int[] c = new int[a.length];
34        for (int size=1; size < a.length; size *= 2) {
35            distribute(a, b, c, size);
36            for (int i = 0; i < a.length / 2; i += size)
37                merge(b, i, i + size, c, i, i + size,
38                    a, 2 * i);
39        }
40    }
41
42    public static void main(String[] args) {
43        int[] a = new int[] {4,9,1,7};
44        MergeSort.sort(a);
45        System.out.println(Arrays.
46            toString(a));
47    }
48 }
```

---

**Listing 1.1.** Java implementation of the merge sort algorithm containing a defect in line [12](#). The line numbering starts at line [5](#) to match the numbering in the source code editor of Figure [1](#).



The semantics of the methods is straight forward. The method `merge` merges the values of array `a` from index `la` to index `ra` and `b` from index `lb` to `rb` into array `c`, starting at position `lc`. If `a` and `b` are sorted within the defined intervals, the result will be a sorted sequence of all elements of the intervals of `a` and `b` starting at `lc` in `c`.

The method `distribute` distributes the elements of `a` to `b` and `c`. Therefore, `a` is divided in subsequent chunks of size `l`. The chunks are copied to `b` and `c`, in turns.

The method `sort` uses `distribute` and `merge` to sort array `a`. Therefore, the contents of `a` are distributed to `b` and `c` and merged back into `a`. To get a completely sorted array `log2a.length`, `distribute` and `a.length-1` `merge` operations must be performed. The sequence of chunk lengths after each `distribute` steps is  $(2^0, \dots, 2^{\log_2 n-1})$ . Please note that this simple implementation of merge sort does only sort arrays of length  $2^i, i \in \mathbb{N}$ . Finally, the method `main` calls the `sort` method to sort the array `[4, 9, 1, 7]` and prints the result to the console output.

The implementation contains a defect in line [12](#), which should read `"c[lc++] = a[la++] ;"`. When executing the `main` method of the `MergeSort` class, the output is the incorrectly sorted array `"[1, 4, 9, 7]"`. We will now show how JHyde can be used to detect the bug.

To debug the `MergeSort` program we first have to execute it using JHyde. After the execution is finished we can use the views of JHyde to explore the recorded execution data and find the bug. [Figure 1](#) shows the state of the JHyde views at the end of the debugging session, when the defect has been detected.

The CT View shows the CT of the `MergeSort` execution. The call of the `main` and the `sort` method have been classified as invalid. The `main` method prints a wrong result array and the `sort` does not sort the elements of the argument array correctly. The two sibling calls of the `sort` call, `toString` and `println` are automatically marked as trusted because they are part of the Java API. The first 4 sub calls of `sort` are classified as valid as they return the expected result. The last call of the `merge` method is still unclassified. As this method is selected in the Computation Tree View, the Node View shows all local variables of this call. For each variable the start value and the end value are the variable's value at the beginning and the end of the selected method call, respectively. The variables whose values changed during the method call, e.g. `la`, `lb`, `c[0]...c[2]`, and `lc`, are marked red (dark grey). The local variable `c` is marked yellow (light grey) because the value of `c` did not change, but some fields of `c` changed.

Due to the intended semantics of the `merge` method, we would expect the chunks `[4, 9]` and `[1, 7]` stored in `a` and `b`, respectively, to get merged to `[1, 4, 7, 9]` in `c`. The Node View tells us that the actual result of the `merge` call stored in `c` is `[1, 4, 9, 7]`. If we mark this method as invalid in the Computation Tree View the debugger informs us that the method `merge` must contain a bug because its inputs were all valid, while it produced a wrong result.

To find the exact location of the bug in the `merge` source code we switch to omniscient debugging. The field at index 2 of the array `c` is a promising entry point. The expected value of this field is 7, while its actual value is 9. We right click this variable in the Node View and select "Show in History View". The result is the value history shown in the History View of [Figure 1](#). During the execution the value of the array field

`c[2]` has changed 4 times. The History View shows the value, the full method signature, the timestamp, and the source code line of each value change. If we select the last entry in the list, where the value is changed to 9, the source code line which caused the value change is highlighted in the source code editor. Furthermore, the event with the timestamp 235 is selected in the Event List View. At this point all views of JHyde look exactly as shown in Figure 1. A further look at the History View tells us that the correct value was already assigned to the `c[2]` in line 10 by the same method call at timestamp 227. The value 9 should have been assigned to `c[3]`. Hence, line 12 must contain the defect.

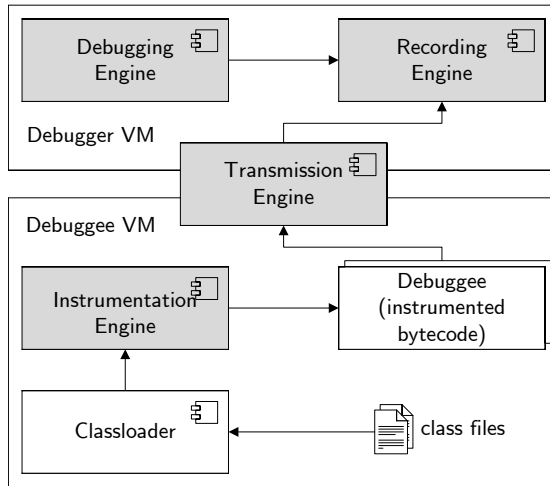
Note that we could have used the Variable History View even at an earlier point in the debugging process to directly jump to the bug. For example, if we evaluate the result of the call `sort`, we notice the erroneous result  $a = [1, 4, 9, 7]$ . Via the value history for `c[2]` we can directly jump from the method call `sort` to the buggy method call `merge`, skipping the previous children of `sort`. Thus, the value history is an excellent tool if we detect a variable with a wrong value. In this case we can easily track back to the origin of the infected program state.

Nevertheless, there are cases where the program state is infected but we do not have a wrong value which we can use to track back to the program bug. This is the case if the program state is not infected due to *wrong* values, but due to *missing* values. For example, consider a merge sort implementation with a defect `merge` method which misses lines 11-12. For the input  $[4, 9, 1, 7]$  this implementation would return  $a = [1, 4, 7, 7]$ . Although, the value of `a[3]`, which must be 9, is not correct, we cannot use the value history of `a[3]` to track back to the origin of incorrect value. During the computation `a[3]` is assigned two values. The first value (7) is assigned during the initialization of `a` in the `main` method and the second value, which is also 7, is assigned during the second invocation of the `merge` method. As both value assignments are correct, the incorrect value of `a[3]` is a missing assignment and not a wrong assignment. We cannot track missing assignments with the Value History. In this cases we need the declarative debugger to identify the buggy method.

If we use a D&Q strategy based on edge coverage instead of the top-down strategy for declarative debugging, the debugger would compute the weight of each node in the CT w.r.t. edge coverage. At the beginning of the declarative debugging process the weight of the nodes shown in the CT View is: 46, 45, 7, 8, 8, 7, 10, 0, and 0 (preorder traversal). Hence, the debugger would at first ask the user to classify the last call of `merge`, whose weight 10 is closest to  $\frac{46}{2}$ . In this case the buggy method call is found after the classification of only one method call, while 6 method calls can be skipped.

## 5 Implementation

Figure 2 shows the architecture of our declarative debugger. It consists of four basic components: the Instrumentation Engine, the Transmission Engine, the Recording Engine, and the Debugging Engine. The components are distributed over the Debugger JVM which conducts the debugging process and the Debuggee JVM which executes the debuggee program. The distribution over two separate JVMs decouples the execution process from the debugging process.



**Fig. 2.** Architecture of the declarative debugger

During the class loading process of the Debuggee JVM the bytecode of the loaded class is manipulated by the Instrumentation Engine. After the instrumentation process, the bytecode of the loaded class contains additional instructions which invoke debugging event methods of the Transmission Engine. The instrumentation is done with the ASM [12], a fast bytecode manipulation framework based on the visitor pattern [13]. The instrumentation process of ASM is controlled by user-defined visitors which inject additional bytecode into the visited class. As the instrumentation process is conducted during the dynamic class loading process at bytecode level, it is transparent from a programmer's point of view. In a debugging session a programmer will always work with the original source code.

The Transmission Engine forwards the received events to the Recording Engine in the Debugger JVM. To transmit the events, they are converted to a byte stream in the Debuggee JVM and sent to the Debugger JVM via a socket connection. In the Debugger JVM the Transmission Engine reconstructs the debugging events from the received byte stream and forwards them to the Recording Engine.

The Recording Engine stores the received events in an event file on the hard disk in chronological order. During the debugging process the execution data is lazily loaded from the event file and transferred to the Debugging Engine.

To reconstruct the program state of the debuggee program at an arbitrary time in the execution history the Recording Engine could theoretically reproduce the effect of all events from the beginning of the execution history to the desired point in the execution history. Especially for larger execution histories this would be much too slow to allow the user to navigate the execution history and the program state in real time. The debugging process would be unacceptably slow.

To speed up the reconstruction of the program states the Recording Engine constructs 3 additional data structures which are kept in the memory of the Debugger VM. The first data structure stores the static structure of the debuggee program, i.e. its classes, methods, fields, and local variables. The second structure is the CT of the debuggee

program, i.e. a tree representing the method call history. The third structure stores data-flow information. It stores an array of timestamps for every variable of the debuggee execution. Each timestamp refers to the time in the execution history when the value of the associated variable changed. For each timestamp the Recording Engine can directly compute the position of the associated event in the event file.

Writing the whole execution data to the event file and keeping only three data structures in memory to speed up the navigation, is a trade-off between memory usage and required effort to navigate the execution data.

The user interface of JHyde, i.e. the set of views as described in section 3, is implemented by the Debugging Engine. Hence, the Debugging Engine's task is to present the execution data to the user in a accessible way and conduct the interactive debugging process.

## 6 Test Results

To test the benefit of the coverage-based D&Q strategy, we have conducted the declarative debugging process with a set of selected test programs. For each of the programs which originally contained no errors we created several test samples. Each sample was generated from the original program by changing one distinct line of the source code such that the resulting test sample would contain a common programming error. This way we have created a set of 32 test samples out of 5 different programs (Avl: 10 test cases, binary tree: 5, B tree: 7, heap sort: 5, hash table: 5). Using the declarative debugger component of JHyde we have determined the number of questions a user would have to answer, i.e. how many method calls he has to evaluate, before the erroneous method is found. The debugging process has been conducted four times for each error sample using top-down (TD), ordinary D&Q, D&Q by def-use chain coverage (DUC), and D&Q by edge coverage (Edge) strategies. DUC and Edge have been conducted with  $p = 0.6$  and  $p = 0.5$ , respectively. These values minimize the number of questions w.r.t. the set of test cases. The number of questions we have to answer for each of strategy allows us to evaluate the efficiency of the different strategies.

Table II shows the test results. Column one indicates the name of the program which was subject to debugging. The values in column 2 indicate the average size of the generated CT. Column 3 shows the average number of trusted method calls in the CT. Columns 4-7 contain the average number of answers necessary to find the buggy method. Each column corresponds to one of the above mentioned strategies. On average D&Q by edge coverage performs best, asking 7.66 questions, while the other strategies yield 8.25 (D&Q by def-use chain coverage), 10 (D&Q), and 12.84 (top-down). The ratio of classified method calls in columns 8-11 is calculated by dividing the average number of answers by the average number of untrusted method calls in the computation. On average we have to validate 23% of the CT using top-down, 18% using D&Q, 15% using D&Q by def-use chain coverage, and 14% using D&Q by edge coverage. Columns 12-14 show the relative savings, i.e. the percentage of questions that is saved w.r.t. a top-down strategy. On average we save 22% with D&Q and 36% with D&Q by de-use chain coverage. D&Q by edge coverage performs best saving 40% percent of questions on average. The number of saved questions is nearly doubled w.r.t. ordinary D&Q.

**Table 1.** Results of test cases processed with the declarative debugger of JHyde

Program	Avg. # of method calls		Avg. number of answers				Classified method calls[%]				Savings [%]		
	Total	Trusted	TD	D&Q	DUC	Edge	TD	D&Q	DUC	Edge	D&Q	DUC	Edge
Avl	132.80	65.30	13.90	9.10	7.70	6.60	0.21	0.13	0.11	0.10	0.35	0.45	0.53
Binary tree	128.00	84.00	11.80	10.20	7.40	7.80	0.27	0.23	0.17	0.18	0.14	0.37	0.34
B tree	195.86	118.86	15.57	13.43	8.57	9.43	0.20	0.17	0.11	0.12	0.14	0.45	0.39
Heap sort	68.00	25.80	9.60	7.20	9.00	9.00	0.23	0.17	0.21	0.21	0.25	0.06	0.06
Hash table	73.40	22.40	11.20	9.60	9.00	5.80	0.22	0.19	0.18	0.11	0.14	0.20	0.48
Total Avg.	119.61	63.27	12.84	10.00	8.25	7.66	0.23	0.18	0.15	0.14	0.22	0.36	0.40

## 7 Related Work

The idea to apply declarative debugging outside the declarative programming paradigm is not new. In 1998 Shahmehri and Fritzson presented an approach for declarative debugging of the imperative language *Pascal* [14] which was further developed by the same authors in [15]. The main difference of our approach w.r.t. these earlier proposals is that Java is a language much more complex than Pascal. The declarative debugging of programs including objects and object states introduces new difficulties.

There are several approaches which use an execution history to locate bugs in Java programs. In a first step these methods trace and record the complete execution of the debuggee program. In a second step the recorded information is used to identify errors in the debuggee program. For example, JavaDD [16] follows a query-based approach, storing events occurring during the debuggee execution in a deductive database. The database can be queried to retrieve the states of different program entities (variables, threads, etc.) at different moments of the computation history to infer where the bug is located. Another approach is omniscient debugging [3] which can trace the execution of the debuggee program back and forth in time. Our debugger does also record the entire execution of a debuggee program before the debugging process is started and it supports omniscient debugging as well. But in contrast to these approaches, our debugger does also support declarative debugging, which concentrates on the logic of method calls, storing them in a structured way, i.e. the CT. The declarative debugging component allows our debugger to guide the debugging process, deducing the wrong method from user answers.

Hoon-Joon Kouh et al. [17] propose a debugging technique for Java which combines algorithmic and step-wise debugging. In contrast to our work they neither present a tool implementation nor do they present a solution to display side effects in an accessible form. Furthermore, they use slicing techniques to reduce the size of the CT, while our tool uses enhanced D&Q strategies.

JHyde is based on a declarative debugger we have developed for the Java programming language [7]. In this paper we extend our declarative debugger to a hybrid debugging tool which supports declarative and omniscient debugging. Furthermore, we have developed and tested new declarative debugging strategies which enhance the D&Q strategy by def-use chain and edge coverage information. These strategies improve our previous answer-inference based approach. First our new optimization is guaranteed to be correct and cannot infer any wrong answers, second the reduction of the debugging effort is slightly more efficient.

As described by Silva [18] a lot of different strategies have been developed to reduce the number of questions asked during declarative debugging. Like our approach some of the more advanced strategies estimate the error probability of CT nodes based on previous user answers. However, none of these approaches is based on coverage information. To the best of our knowledge, there exists no strategy which is based on data-flow and/or control-flow coverage to calculate the error probability of nodes of the CT.

## 8 Conclusions and Future Work

We have presented a tool which enables the hybrid debugging of Java programs combining declarative and omniscient debugging methods. The major advantage of declarative debugging compared to conventional debugging is that it works on a higher level of abstraction. The tester is relieved from the task to inspect the state space after each instruction starting from some break point. By answering questions about the soundness of some method calls the user can concentrate on the semantics. A major drawback of declarative debugging is the fact, that we can only determine a buggy method call but not the buggy statements inside a method. This downside is cured by the fact that JHyde does also support omniscient debugging. With omniscient debugging the user can track the execution of a debuggee program statement-wise back and forth in time. The combination of both debugging techniques JHyde can profit from the advantages both methods, the abstraction of the declarative debugging method and the precision of the omniscient debugging method.

The JHyde user interface consists of four views which allow the user to inspect and navigate the execution history of a debuggee program effectively. At any point in the debugging process the user is free to choose the most suitable views and/or debugging method.

A particular novelty of our approach is the usage of code-coverage criteria such as def-use chain coverage and coverage of the edges of the control-flow graph to calculate the error probability of method calls. The error probability increases with the number of covered entities, i.e. def-use chains or edges of the CFG. Furthermore, our strategy takes previous user answers into account. The more often an entity is classified as valid the lesser its contribution to the error probability of a method call.

We have conducted a number of tests, using different declarative debugging strategies to debug a set of buggy Java programs. The results show that up to 40% of the questions asked can be saved with our coverage-based D&Q strategy w.r.t. the ordinary top-down strategy. Furthermore, our strategy is an efficient improvement to the ordinary D&Q strategy. In our test scenario the number of questions saved could be nearly doubled if D&Q is enhanced by coverage-based error probability. The improved reduction of the debugging effort makes our debugger more suitable for real-world applications where the debugging effort has great influences on the costs of the software development process.

In the future we plan to investigate how our hybrid debugging method can be extended to multi-threaded program executions. Although, our debugger is already capable of recording multi-threaded programs, there are still some problems to be tackled regarding a multi-threaded hybrid debugging method. For example, it is much harder to check the validity of method calls if their execution is interleaved.

## References

1. Hailpern, B., Santhanam, P.: Software debugging, testing, and verification. *IBM Systems Journal* 41, 4–12 (2002)
2. Zeller, A.: How Failures Come to Be. In: *Why Programs Fail: A Guide to Systematic Debugging*, pp. 1–26. Morgan Kaufmann (2005)
3. Lewis, B.: *Debugging Backwards in Time*. CoRR cs.SE/0310016 (2003)
4. Shapiro, E.Y.: *Algorithmic Program DeBugging*. MIT Press (1983)
5. Nilsson, H.: How to look busy while being as lazy as ever: the Implementation of a lazy functional debugger. *Journal of Functional Programming* 11, 629–671 (2001)
6. Caballero, R., Rodríguez-Artalejo, M.: A Declarative Debugging System for Lazy Functional Logic Programs. *Electronic Notes in Theoretical Computer Science* 64 (2002)
7. Caballero, R., Hermanns, C., Kuchen, H.: Algorithmic Debugging of Java Programs. *Electronic Notes in Theoretical Computer Science* 177, 75–89 (2007)
8. Hermanns, C.: JHyde - Eclipse plugin (2011), <http://www.wi.uni-muenster.de/pi/personal/hermanns.php>
9. Eclipse Foundation: Eclipse IDE (2011), <http://www.eclipse.org/>
10. Pressman, R.S.: *Software Engineering: A Practitioner’s Approach*, 5th edn. McGraw-Hill (2001)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn. The MIT Press (2001)
12. Object Web: Asm (2011), <http://asm.ow2.org/>
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns*. Addison-Wesley, Boston (1995)
14. Shahmehri, N., Fritzon, P.: Algorithmic Debugging for Imperative Languages with Side-Effects. In: Hammer, D. (ed.) CC 1990. LNCS, vol. 477, pp. 226–227. Springer, Heidelberg (1991)
15. Fritzon, P., Shahmehri, N., Kamkar, M., Gyimothy, T.: Generalized algorithmic debugging and testing. *ACM Letters on Programming Languages and Systems* 1, 303–322 (1992)
16. Girgis, H.Z., Jayaraman, B.: JavaDD: a Declarative Debugger for Java. Technical report, Department of Computer Science and Engineering, University at Buffalo (2006)
17. Kouh, H.-J., Kim, K.-T., Jo, S.-M., Yoo, W.-H.: Debugging of Java Programs Using HDT with Program Slicing. In: Laganá, A., Gavrilova, M.L., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3046, pp. 524–533. Springer, Heidelberg (2004)
18. Silva, J.: A Comparative Study of Algorithmic Debugging Strategies. In: Puebla, G. (ed.) LOPSTR 2006. LNCS, vol. 4407, pp. 143–159. Springer, Heidelberg (2007)

# Combined Constraint-Based Analysis for Efficient Software Regression Detection in Evolving Programs

Anh D. Le, Tho T. Quan, Nguyen T. Huynh, Phung H. Nguyen, and Nhat-Van Le

Faculty of Computer Science and Engineering,  
Hochiminh City University of Technology, Hochiminh City, Vietnam  
tintinkool@gmail.com, {qttho,htnguyen,phung}@cse.hcmut.edu.vn,  
ccnhatvan@yahoo.com

**Abstract.** Software regression is a bug that makes software stop functioning normally after a certain event. In this paper, we investigate detecting regression when software evolves to a new version. In this context, regression bugs occur in parts of software that already passed testing process in the old version. Hence, such kind of bugs is difficult to be discovered if normal strategy like white-box testing is applied. Moreover, since both old and new versions must be taken into account during the testing process, the computational cost is usually high in this con-text.

Concolic testing in an emerging extension of white-box testing that can reduce significantly the number of execution paths needed to be analyzed. However, the typical concolic testing is not really efficient when dealing with software regression. Thus, we propose a new approach based on combined constraint to solve this problem, known as CTGE (Efficient Constraint-based Test-cases Generation) approach. The soundness of our theoretical contribution is formally proved and supported by some initial experiments conducted in education environment.

**Keywords:** Constraint-based Test-case Generation, Regression Bugs, Evolving Programs Debugging.

## 1 Introduction

A software regression is a bug which makes a feature stop functioning as intended after a certain event. In practice, this kind of bug involves in many contexts like testing documentation [8] or testing of component-based software [9].

In this paper, we investigate a situation where regression bugs regularly occur. That is, when a program is evolved into a new version to meet new requirements or just to refine the code, chances are that the new evolved program may accidentally violate the original requirements.

So far, software practitioners have still been commonly using testing techniques to detect program bugs. Traditionally, a set of test-cases will be generated for testing. Basically, a test-case is a set of inputs, execution conditions and desired outputs which can be tested by the system when functioning accordingly using some test



procedures and test scripts. However, the generation of test-case is usually costly and thus requiring a systematic method. White-box testing (or structural testing) technique, based on flow-control analysis, is typically applied in this case [6]. In this method, the tested program is analyzed in terms of control flow and data flow. Thus, the test-cases are generated accordingly to (1) exercise independent paths within a module or unit; (2) exercise logical decisions on both their true and false side; (3) execute loops at their boundaries and within their operational bounds; and (4) exercise internal data structures to ensure their validity [10].

The main advantage of white-box testing is that it does not only detect the bugs, but also help locate the piece of code that causes the problem. However, the primary disadvantage of this method is the suffering of high execution cost. If we have a program which has 10 independent if...then...else... statements, there are totally 210 execution paths needed to be explored. When dealing with evolving programs, this problem is still even more crucial, since we must do the combined analysis on both old and new versions.

There are many attempts which have been made to efficiently explore program paths for test-case generation purpose, in which the infeasible combined paths are eliminated [3]. Nowadays, the most prominent testing method based on the white-box approach technique is concolic testing approach. In this approach, apart from concrete execution of the tested program based on certain input, symbolic execution [4] is also involved to resolve the path constraints by means of theorem provers. Hence, only the feasible paths are considered for generating appropriate test-case, thus significantly reducing the numbers of test-case needed for path coverage. This technique is then adopted and exploited remarkably in various testing tools like PathCrawler [16], jCUTE [13] and SAGE [5]. However, DASH [1] is perhaps the most efficient concolic-based technique which uses abstraction to deal which real complex practical programs.

Unfortunately, the typical white-box testing and its concolic-based variations are not able to fully detect this regression in evolving programs, even when the flow analysis is performed on both original and evolved versions. To make it clearer, in the following discussion in Section 2, we will give some motivating examples on this issue. To overcome this problem, we suggest an approach of using constraints combined from path conditions of both original and evolved program versions. We then formally prove the soundness of this approach, under the context of well-conditioned programs. To make this approach practical, we also propose an algorithm, known as CTGE (Efficient Constraint-based Test-case Generation) that reduces the cost of test-case generation from exponential complexity to linear one.

The rest of the paper is organized as follows. Section 2 presents a motivating example which shows that when a program evolves, neither test-cases generated merely from the old version nor the new version are sufficient to detect regression bug. Section 3 discusses our proposed approach on generating test-case by combining execution paths from the previous version and the evolved version of a program into constraints. In Section 4, we introduce the ultimate CTGE algorithm, an improvement of the test-case generation algorithm to reduce its complexity significantly. Section 5 gives some experiments. Finally, Section 6 concludes the paper.

## 2 Motivating Example

To give a clear motivation of our work, we first define well-conditioned program as follows.

**Definition 1 (Well-conditioned Program).** A program  $P$  is said well-conditioned with respect to a property  $\pi$ , denoted as  $\angle(P, \pi)$ , if  $P$  produces same outcomes w.r.t.  $\pi$  for all inputs satisfying same path conditions in  $P$ .

---

```
void f(int n){
    if(n>0) return n = 2*n;
    else return n = -2*n;
}
```

---

**Listing 1.** An example program  $P_X$ .

**Example 1.** In the example program  $P_X$  given in Listing 1, there are two path conditions  $C1: (n > 0)$  and  $C2: (n \leq 0)$  corresponding to the execution paths of *if* and *else* clauses. Let us consider two properties: (i)  $\pi_1$ : the program result is a positive number; and (ii)  $\pi_2$ : the program result is a number dividable by 4. One can easily observe that all inputs satisfying  $C1$  (e.g.  $n = 4$ ) will make  $\pi_1$  true. Similarly, all inputs satisfying  $C2$  (e.g.  $n = -7$ ) will make  $\pi_1$  false. Thus, we can say  $PE$  is well-conditioned w.r.t.  $\pi_1$  or  $\angle(PE, \pi_1)$ . In contrast, all inputs satisfying  $C1$  or  $C2$  cannot guarantee the same outcomes w.r.t.  $\pi_2$ . For instance, the inputs of  $n=3$  and  $n=4$  both satisfy  $C1$  but making  $\pi_2$  false and true respectively.

In the situation of testing a program  $P$  against a requirement  $R$ , it is easily observable that if  $\angle(P, R)$  then a set of test-cases covering all of path conditions in  $P$  is sufficient to detect any bugs if occurring.

**Example 2.** (Requirement  $R_P$ ) Write a function  $f$  taking an integer parameter  $n$  that returns the absolute value of  $n$ .

For instance, let us consider the requirement  $R_P$  given in Example 2. In Listing 2(a), an implementation of  $f$  is given in program  $V_P$ , which has two path conditions  $P_1 = (n > 0)$  and  $P_2 = \neg(n > 0)$ . Since  $\angle(V_P, R_P)$ , two test-cases covering those two path conditions of  $V_P$ , e.g.  $n=5$  and  $n=-7$ , are sufficient to ensure the correctness of  $V_P$ <sup>1</sup>.

**Example 3.** (Requirement  $R_E$ ) Write a function  $f$  taking an integer parameter  $n$  that returns the absolute value of  $n$ . In addition, if  $n$  is greater than 3,  $f$  will increase the value of the global variable  $Global$  by 1.

Assume that the requirement is now upgraded in order to fulfill a new requirement  $R_E$  as present in Example 3. We say that  $R_E$  is *evolved* from  $R_P$  since we can consider  $R_E = R_P \cup R_N$ , where  $R_N$  is the additional requirement of “if  $n$  is greater than 3,  $f$  will increase the value of the global variable  $Global$  by 1”. Then,  $V_P$  is evolved accordingly as a new version  $V_E$  given in Listing 1(b).

---

<sup>1</sup> In this paper we only discuss generating test-cases covering all execution paths. The test-cases for extreme cases, for example  $n=0$  for the program in Listing 1(a), are out of the scope of this paper.

```

int f(int n){
    if(n>0) return n;
    else return -n;
}
(a) The previous version  $V_p$ 


---


int f(int n){
    if(n>3)
    {
        Global++;
        return n;
    }
    else return -n;
}
(b) The evolved version  $V_E$  with regression bugs
    
```

**Listing 2.** Evolving programs

Since  $\angle(V_E, R_N)$ , two test-cases covering all two path conditions  $Q_1 = n > 3$  and  $Q_2 = \neg(n > 3)$  of  $V_E$ , e.g.  $n = 5$  and  $n = -3$ , are sufficient to test whether  $V_E$  fulfills the additional requirement  $R_N$ . But those two test-cases cannot show that  $V_E$  violates the old requirement  $R_p$ . For example, if the input is 2, the result will wrongly be -2. Generally, this problem arises since we cannot always guarantee that  $\angle(V_E, R_p)$ .

**Table 1.** Constraints generated

Conjunction	Combined Constraint	Simplified Constraints	Test-case
$P_1 \wedge Q_1$	$n > 0 \ \&\& \ n > 3$	$n > 3$	$n = 5$
$P_1 \wedge Q_2$	$n > 0 \ \&\& \ \!(n > 3)$	$0 < n < 3$	$n = 2$
$P_2 \wedge Q_1$	$\!(n > 0) \ \&\& \ n > 3$		no test-case
$P_2 \wedge Q_2$	$\!(n > 0) \ \&\& \ \!(n > 3)$	$n \leq 0$	$n = -7$

Note that even though this is only a toy problem, the logic error in Listing 2(b) reflects a practical situation occurring in evolving programs. That is, while making the evolved program satisfy the additional requirements, we may accidentally violate the original requirements. We consider this kind of error as *regression bugs* as introduced in Section 1.

The above-discussed examples also show that even though employing test-case covering all path conditions in both previous and evolved versions of evolving programs, we can still miss the regression bugs. In the next section, we will introduce the combined constrain solving approach to deal with this problem.

### 3 Constraint Solving for Regression Bugs Detection

The motivating example in Section 2 has well illustrated that when a program evolves from an old version to a new evolved version, both path conditions of two versions

should be taken into account when generating test-cases. The philosophy here is that a program will be considered evolved when new requirement is added, like stated in Example 1 and Example 2. Thus, the new program should satisfy not only new requirements added but also old requirements as well.

In order to do this, we make use of an approach based on combined constraint as follows. From the path conditions of both old and evolved versions, we generate *combined constraints* by make conjunctions of the path conditions. Each combined constraint is a conjunction of a pair of path conditions, one from the old version and the other from the evolved version. Then, we generate test-cases that cover all of possibly combined constraints. That is, we generate some input values that satisfy the combined constraints. The constraint solving here is by no means an easy task to be done manually. In practice, we use the *theorem prover Z3* [1] to make the constraints simplified and generate test-cases accordingly for each constraint generated.

For instance, Table 1 presents the combined constraints generated from programs in Listing 1 and the test-cases generated accordingly. Obviously, we can detect the regression bug when the test-case of  $n = 2$  is executed.

The algorithm CTG to generate test-case is presented in Figure 1. In the algorithm, there is a particular operation of *solve\_constraint* included. This operation is in charge of generating combined constraints by conjunction and makes them simplified, then finds an appropriate test-case fulfilling the constraint. This operation is supposedly handled by means of a theorem prover.

The CTG algorithm should be sufficient to find any regression bugs. In Theorem 1, we show that this statement is sound under Assumption 1.

---

```

Algorithm: CTG (Constraint-based Test-cases
Generation)
Input:  $V_P, V_E$ : Original and evolved programs
Output:  $T$ : set of test-cases
Operations
   $T = \emptyset$ 
  Foreach (path condition  $\alpha \in V_P$ )
    Foreach (path condition  $\beta \in V_E$ )
       $t = \text{solve\_constraint}(\alpha \wedge \beta)$ 
      If  $t \neq \emptyset$  then
        Add  $t$  to  $T$ 
      Endif
    End for
  End for

```

---

**Fig. 1.** The CTG (Constraint-based Test-case Generation) algorithm

**Assumption 1.** Given a previous version  $V_P$  that is well-conditioned w.r.t an original requirement  $R_P$ , i.e.  $\angle(V_P, R_P)$ . When  $V_P$  is evolved into a new version  $V_N$  to fulfil new requirement  $R_N$ , then  $\angle(V_N, R_N)$ .

**Theorem 1.** *The set of test-cases generated by CTG algorithm is sufficient to detect regression bugs on a program  $V_N$  evolved from original program  $V_P$  when the requirements evolved from  $R_P$  to  $R_N$  respectively.*

*Proof.* If there is a regression bug  $\vartheta$  occurring in  $V_N$ , then exists an input  $I$  that results in different outcomes of  $O_P$  and  $O_N$  w.r.t.  $R_P$  when executed in  $V_P$  and  $V_N$  respectively. Assume that  $I$  belongs to condition paths  $\alpha \in V_P$  and  $\beta \in V_N$  respectively. Since  $\angle(V_P, R_P)$  and  $\angle(V_N, R_N)$ ; and  $R_P \subset R_N$  then  $\angle(V_N, R_P)$ , any input generated from the combined constraint  $\alpha \cap \beta$  will result in the same outcome  $O_P$  and  $O_N$  w.r.t.  $R_P$  when executed in  $V_P$  and  $V_N$  respectively. Since  $I \in \alpha$  and  $I \in \beta$  then  $\alpha \cap \beta \neq \emptyset$ , i.e. there is at least an input  $I' \in \alpha \cap \beta$  existing and will be generated when the CTG algorithm tries to make all possible combinations of condition paths between  $V_P$  and  $V_N$ , thus causing the corresponding regression bug  $\vartheta$  to be detected accordingly.

*Complexity Analysis.* It is easily observable that the CTG algorithm produces test-cases by solving of possible constraints generated from the old version  $V_P$  and the evolved version  $V_E$ . Thus, it suffers high complexity as it takes  $O(N \times M)$  times to make a solver process all constraints where  $N$  and  $M$  are the path conditions on  $V_P$  and  $V_E$  respectively. In the next section, we introduce the  $CTG^E$  algorithm, an enhanced algorithm that only involves solver to process solvable constraints, thus improving significantly the performance of test-case generation process.

## 4 The $CTG^E$ Algorithm

Among the 4 constraints presented in Table 1, there are 3 solvable constraints and one unsolvable one (i.e. a constraint that we cannot find any test-case/input satisfying it). However, the CTG algorithm requires a solver to process unnecessarily all of 4 constraints. To overcome this problem, in the new version of  $CTG^E$  algorithm presented in this section, we will take into account only solvable constraints. The  $CTG^E$  algorithm is shown in Figure 2.

The major improvement of  $CTG^E$  is that it does not try to make all possible combined constraints. Instead,  $CTG^E$  processes each path condition of the original version  $V_P$ . For each path condition,  $CTG^E$  first produces an appropriate test-case. Then, it calls a subprocedure named *combine* to further process.

For every test-case  $t$  processed in *combine*, a specific function named *symbolic\_exec* will be called to find the corresponding path conditions of  $t$  when executed in  $V_P$  and  $V_E$  respectively. The operation of *symbolic\_exec* will perform *symbolic execution*, a classical technique to trace the execution path of given input by tracking symbolic rather than actual values [7]. Based on the retrieved path conditions, *combine* keeps generating relevant constraints and calls itself recursively to generate more suitable test-cases. During the whole process of  $CTG^E$ , we also make use of a special constraint named  $C_{mark}$  which marks the explored parts in the space of test-case domain. Therefore,  $CTG^E$  can avoid duplication when generating constraints and test-cases.

---

**Algorithm:**  $CTG^E$  (Efficient Constraint-based Test-cases Generation)

**Input:**  $V_p, V_E$ : Original and evolved programs

**Output:**  $T$ : set of test-cases

**Operations**

```

 $T = \emptyset$ 
 $C_{mark} = \emptyset$ 
Foreach (path condition  $\chi \in V_p$ )
     $t = solve\_constraint(\chi \cap \neg C_{mark})$ 
    combine( $t$ )
End For

SubProcedure combine (test-case  $t$ )
Begin
    add  $t$  to  $T$ 
     $\alpha = symbolic\_exec(t, V_p)$ 
     $\beta = symbolic\_exec(t, V_E)$ 
     $C_{mark} = C_{mark} \cup (\alpha \cap \beta)$ 
    if  $(\alpha \cap \neg \beta \cap \neg C_{mark}) \neq \emptyset$  then
        combine( $solve\_constraint(\alpha \cap \neg \beta \cap \neg C_{mark})$ )
    end if
    if  $(\neg \alpha \cap \beta \cap \neg C_{mark}) \neq \emptyset$  then
        combine( $solve\_constraint(\neg \alpha \cap \beta \cap \neg C_{mark})$ )
    end if
End

```

---

Fig. 2. Efficient Constraint-based Test-case Generation ( $CTG^E$ ) algorithm

**Theorem 2.** *The set of test-cases generated by  $CTG^E$  algorithm is sufficient to detect regression bugs on a program  $V_N$  evolved from old program  $V_P$  when the requirements evolve from  $R_P$  to  $R_N$  respectively.*

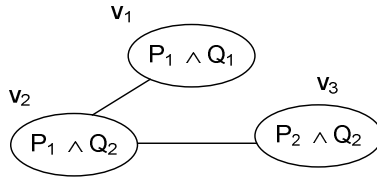


Fig. 3. A graph representation of combined constraints

*Proof.* We consider an undirected graph  $G = \langle V, E \rangle$  constructed as follows. Each vertex  $v$  in  $V$  corresponds to a solvable combined constraint generated by the CTE algorithm. We add an edge  $e_{ij} = (v_i, v_j)$  to  $E$  if  $v_i$  and  $v_j$  are subconditions of a path condition in either  $V_p$  or  $V_N$ .

For example, in Figure 3 is the graph constructed when we consider the program versions presented in Figure 1 and the combined constraints in Table 1. In the graph, there are three vertices corresponding to three solvable constraints in Table 1. There is an edge connecting  $v_1$  and  $v_2$  since their constraints are both subconditions of  $P_1$ . Similarly,  $v_2$  and  $v_3$  are connected since their constraints are both subconditions of  $Q_2$ . Next, we relate the test-case generation process in the  $CTG^E$  as graph traversal carried out in  $G$ . A vertex  $v$  is considered visited if  $CTG^E$  produces a test-case satisfying the corresponding combined constraint of  $v$ . According to Theorem 1, if all vertices in  $G$  are visited after  $CTG^E$  finishes, then  $CTG^E$  generates sufficient test-cases to detect any regression bugs.

When  $CTG^E$  begins, it starts by a certain test-case  $I$  generated to satisfy a path condition  $\alpha$  of  $V_p$ . Using symbolic execution, one can determine the path condition  $\beta$  of  $V_N$  which  $I$  belongs to. It means that a vertex  $q = \alpha \wedge \beta$  just has been initially visited.

Consider the formula  $\alpha \wedge \beta$  referring to a vertex  $q'$ , which should be connected to  $q$  since  $\alpha \wedge \beta$  and  $\alpha \wedge \beta$  are both subcondition of  $\alpha$ . Let  $C_{mark}$  be the formula representing all of vertices already visited (i.e. the combined constraints whose corresponding test-cases have been generated already). Similarly reasoning, we finally obtain that the two formulas  $\alpha \wedge \beta \wedge C_{mark}$  and  $\alpha \wedge \beta \wedge C_{mark}$  should represent all vertices connecting to  $q$  which have not been visited. By recursively solving those formulas and updating  $C_{mark}$  in the subprocedure *combine*,  $CTG^E$  will iteratively visit all of vertices in the connected component which  $q$  belongs to.

Lastly, one can note that by checking all of path conditions of  $V_p$ ,  $CTG^E$  will travel to all possible connected components of  $G$ . Thus, all vertices of  $G$  will be logically visited when  $CTG^E$  performed and there are no vertices doubly visited.

For instance, consider using  $CTG^E$  for generating test-case for evolving programs in Listing 1. Firstly, the two path conditions  $P_1$  and  $P_2$  are collected. Then,  $CTG^E$  generates randomly a test-case for a path condition. Let it be  $n = 4$  for  $P_1$ . Performing symbolic execution on the test-case, one can realize that the test-case falls into the combined constraint  $P_1 \wedge Q_1 = n > 0 \ \&\& \ n > 3 = n > 3$ . Then,  $CTG^E$  tries to solve the formula  $P_1 \wedge Q_1 \wedge C_{mark}$  with  $C_{mark}$  being updated as  $C_{mark} = P_1 \wedge Q_1$ . We have  $P_1 \wedge Q_1 \wedge C_{mark} = n > 0 \ \&\& \ (n > 3) \ \&\& \ (n > 3) = n > 0 \ \&\& \ n \leq 3$ . Then, a test-case is generated accordingly, e.g.  $n = 2$ .

Next, *combine*(2) is invoked, which is corresponding to the constraint  $P_1 \wedge Q_2$  with  $C_{mark}$  being updated as  $n > 3 \cup n > 0 \ \&\& \ n \leq 3 = n > 0$ . We then have  $P_1 \wedge Q_2 \wedge C_{mark} = n > 0 \ \&\& \ n > 3 \ \&\& \ !(n > 0) = \emptyset$ , then then this formula is not considered.

---

```

int grade(int n){
    if(n > 100) return Invalid;
    else if(n>=90) return Excellent;
    else if(n>=80) return Very good;
    else if(n>=70) return Good;
    else if(n>=60) return Fairly good;
    else if(n>=50) return Average;
    else if(n>=0) return Fail;
    else return Invalid;
}
    
```

(a) Student grading program – preliminary version

```

int grade(int n){
    if(n > 100) return Invalid;
    else if(n>90) return Excellent;
    else if(n>80) return Very good;
    else if(n>70) return Good;
    else if(n>60) return Fairly good;
    else if(n>50) return Average;
    else if(n>0) return Fail;
    else return Invalid;
}
    
```

(b) Student grading program – final version

---

**Listing 3.** Evolving programs

Meanwhile, we have  $P_1 \wedge Q_2 \wedge C_{mark} = !(n > 0) \ \&\& \ !(n > 3) \ \&\& \ !(n > 0) = n \leq 0$ . Solving this constraint, we, for instance, get a new test-case of  $n = -7$ . Then, *combine*(-7) is invoked accordingly. At the moment,  $C_{mark}$  is updated as  $n > 0 \cup !(n > 0) \ \&\& \ !(n > 3) = n > 0 \cup n \leq 0$ , making  $P_2 \wedge Q_2 \wedge C_{mark} = P_2 \wedge Q_2 \wedge C_{mark} = P_1 \wedge Q_1 \wedge C_{mark} = \emptyset$ . Thus, the algorithm stops with no more test-cases generated.

*Complexity Analysis.* Performing elementary analysis on  $CTG^E$ , one can realize that  $CTG^E$  will involve the embedded solver  $2K$  times, with  $K$  is the number of test-cases generated and  $K \leq N+M$  where  $N$  and  $M$  are the path conditions on  $V_P$  and  $V_E$  respectively. If we take into account the actions of generating  $N$  path conditions on  $V_P$ , the total complexity of  $CTG^E$  will be  $O(2K + M) \sim O(3N + M)$  which should be improved significantly compared to that of the original  $CTG$ .

To illustrate this, consider the two versions of evolving programs given in Listing 3. The program intends to grade students' works. After the preliminary version is finished as presented in Listing 3(a), a new version is released afterward as presented in Listing 3(b).

**Table 2.** Initial example test-cases for preliminary version in Listing 2

Condition	Test case
$n > 100$	103
$n \geq 90 \ \&\& \ n \leq 100$	91
$n \geq 80 \ \&\& \ n < 90$	85
$n \geq 70 \ \&\& \ n < 80$	73
$n \geq 60 \ \&\& \ n < 70$	66
$n \geq 50 \ \&\& \ n < 60$	54
$n \geq 0 \ \&\& \ n < 50$	32
$n < 0$	-7

**Table 3.** Total test-cases generated for evolving versions in Listing 2

Test case	$\alpha$	$\beta$	$\alpha \neg \beta$	$\neg \alpha \beta$
103	$n > 100$	$n > 100$	$\emptyset$	$\emptyset$
91	$n \geq 90 \ \&\& \ n \leq 100$	$n > 90 \ \&\& \ n \leq 100$	new test-case: 90	$\emptyset$
90	$n \geq 90 \ \&\& \ n \leq 100$	$n > 80 \ \&\& \ n \leq 90$	$\emptyset$	$\emptyset$
85	$n \geq 80 \ \&\& \ n < 90$	$n > 80 \ \&\& \ n < 90$	new test-case: 80	$\emptyset$
80	$n \geq 80 \ \&\& \ n < 90$	$n > 70 \ \&\& \ n < 80$	$\emptyset$	$\emptyset$
73	$n \geq 70 \ \&\& \ n < 80$	$n > 70 \ \&\& \ n < 80$	new test-case: 70	$\emptyset$
70	$n \geq 70 \ \&\& \ n < 80$	$n > 60 \ \&\& \ n < 70$	$\emptyset$	$\emptyset$
66	$n \geq 60 \ \&\& \ n < 70$	$n > 60 \ \&\& \ n < 70$	new test-case: 60	$\emptyset$
60	$n \geq 60 \ \&\& \ n < 70$	$n > 50 \ \&\& \ n < 60$	$\emptyset$	$\emptyset$
54	$n \geq 50 \ \&\& \ n < 60$	$n > 50 \ \&\& \ n < 60$	~ new test-case: 50	$\emptyset$
50	$n \geq 50 \ \&\& \ n < 60$	$n > 0 \ \&\& \ n < 50$	$\emptyset$	$\emptyset$
32	$n \geq 0 \ \&\& \ n < 50$	$n > 0 \ \&\& \ n < 50$	~ new test-case: 0	$\emptyset$
0	$n \geq 0 \ \&\& \ n < 50$	$n \leq 0$	$\emptyset$	$\emptyset$
-7	$n < 0$	$n \leq 0$	$\emptyset$	$\emptyset$



There are 8 path conditions in each version, therefore the CTG algorithm will make use of the solver 64 times to generate test-cases. Meanwhile, when  $CTG^E$  is performed, it will basically generate 8 initial test-cases covering 8 path conditions of the preliminary program as presented in Table 2 (the test-cases presented here are just of example basis).

Then, when the algorithm advances, there will be 6 additional test-cases generated corresponding to non-empty domain marked in Table 3. Totally, the solver only needs to be involved 14 times for generating test-cases and 8 times for initial path conditions.

**Table 4.** Programming problems used as experimental data

No	Problem	Constraint	Solver calls (CTG)	Solver calls (CTG <sup>E</sup> )
1	Leap year checking	14	42	40
2	Triangle classification	22	89	31
3	Date validation checking	62	736	90
4	Time validation checking	28	96	37
5	Factorial computing	28	96	58
6	Calculating $x^y$	28	96	56
7	Prime number checking	56	384	92
8	Sum of 1..n	25	84	54

**Table 5.** Bugs detected by white-box and combined constraints approach

Problem No	Real Bugs	Detected by white-box	Detected by DART	Detected by CTG <sup>E</sup>
1	12	11	11	12
2	10	6	8	10
3	12	10	10	10
4	13	10	11	13
5	14	14	14	13
6	11	11	11	11
7	12	12	12	12
8	12	12	12	11
<b>Total</b>	96	86(89%)	89 (93%)	94(98%)

## 5 Experimental Results

In order to evaluate the performance of the  $CTG^E$  algorithm, we have conducted an experiment in the education domain. The requirements to be fulfilled in this experiment are non-trivial programming problems given to students. The list of problems is given in Table 4, which also gives the information of the combined constraints make from path conditions. For loop-based programs, the path conditions are computed using the coverage analysis technique [14] in which the loops are

enforced to repeat respectively 0,1,2 and more than 2 times. Thus, our algorithm may have some limitations on programs with complicated loops.

The dataset used in this experiment is collected from the works of 50 students. In fact, there are actual marked programming works. Basically, for each programming problem, the teacher will produce a sample solution. In order to mark student works automatically, some test-cases are generated for testing. However, as discussed in Section 2, if we apply the typical white-box approach for generating test-cases, the test-cases are not sufficient to detect all of bugs in student works, even though both sample solutions and actual students' works are concerned when test-cases are generated. We also did experiment using DART algorithm, considered as the most popular concolic-based technique.

When manually inspecting, we observe that there are only 89% students' bugs detected using white-box approach. Exact information on improvement of bug detection is given in Table 5. When the constraint-based approach is applied with teachers' sample solutions playing the roles of original versions and student works evolved versions, the performance of bug detection is significantly improved with 98% bugs detected. Few bugs are still missed because the Z3 solver fails to resolve some complex non-linear expression in path conditions.

As compared to white-box testing, DART only improves slightly in terms of bug detection. However, DART reduced significantly the computational time due to its capability of path exploration reduction, as presented in Figure 4. As compared to CTG and white-box testing, DART outperforms in terms of execution time. However, with the improvement made on CTG<sup>E</sup>, the improved algorithm is comparable with DART in terms of speed and therefore practical for dealing with real programs.

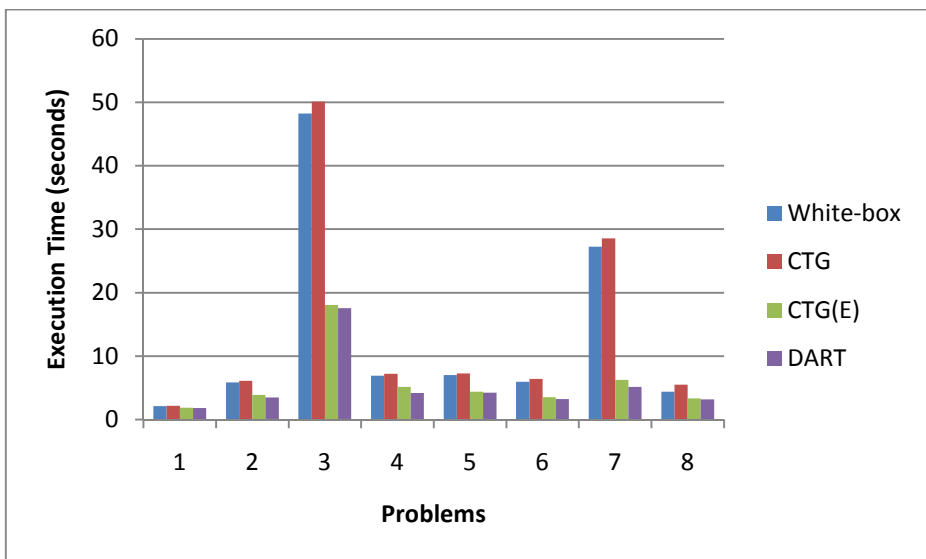


Fig. 4. Execution time of testing methods

## 6 Conclusions

In this paper, we propose an approach for effective detection of software regression in evolving programs. First, we explain by a means of motivation example why white-box testing may fail to discover regression bugs. Then, we introduce a concolic-based approach to detect the regression bugs. Instead of individually performing concolic testing on both old and new programs, we suggest combining constraints extracted from both programs, thus ensuring the detection of any regression error. Our approach is presented theoretically with formal definitions and proof provided. To avoid the explosion path problem, we refine the approach as ultimate algorithm known as  $CTG^E$  whose complexity is reduced significantly to linear time.

We have also preliminary tested our approach in education environment, with dataset being programming works collected from students. The experimental results showed that the  $CTG^E$  algorithm achieved better performance in terms of bug detection coverage and execution time, compared to the white-box testing and DART, a popular concolic-based testing. It also shows potential to apply  $CTG^E$  to industry environment.

**Acknowledgements.** This work is part of the Higher Education Project 2 project (supported by World Bank and Hochiminh – Vietnam National University).

## References

1. Beckman, N.E., Nori, A.V., Rajamani, S.K., Simmons, R.J., Tetali, S.D., Thakur, A.V.: Proofs from Tests. *IEEE Transactions on Software Engineering* (2012)
2. Bjørner, N., Moura, L.D.: Z310: Applications, Enablers, Challenges and Directions. In: *Proceedings of Workshop on Constraints in Formal Verification* (2009)
3. Cadar, C., Dunbar, D., Engler, D.R.: Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: *USENIX Symposium on Operating Systems Design and Implementations* (2008)
4. Godefroid, P., Klarlund, N., Sen, K.: DART: Directed automated random testing. In: *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, vol. 40(6), pp. 213–223. ACM (2005), doi:10.1145/1065010.1065036
5. Godefroid, P.: Random testing for security: blackbox vs. whitebox fuzzing. In: *Proceedings of the 2nd International Workshop on Random Testing: Co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering*, p. 1. ACM (2007), doi:10.1145/1292414.1292416
6. Hutcheson, M.L.: *Software Testing Fundamentals-Methods and Metrics*. Wiley Publishing (2003)
7. King, J.C.: Symbolic execution and program testing. *Communications of the ACM* 19(7), 385–394 (1976), doi:10.1145/360248.360252
8. Morasca, S., Taibi, D., Tosi, D.: T-DOC: A Tool for the Automatic Generation of Testing Documentation for OSS Products. In: Ågerfalk, P., Boldyreff, C., González-Barahona, J.M., Madey, G.R., Noll, J. (eds.) *OSS 2010. IFIP AICT*, vol. 319, pp. 200–213. Springer, Heidelberg (2010)

9. Orso, A., Harrold, M.J., Rosenblum, D., Rothermel, G., Soffa, M.L., Do, H.: Using component metacontent to support the regression testing of component-based software. In: Proceedings of IEEE International Conference on Software Maintenance (2001)
10. Pressman, R.: *Software Engineering: A Practitioner's Approach*. McGraw Hill, Boston (2001)
11. Qi, D., Roychoudhury, A., Liang, Z.: Test generation to expose changes in evolving programs. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 397–406. ACM (2010), doi:10.1145/1858996.1859083
12. Sen, K., Marinov, D., Agha, G.: CUTE: a concolic unit testing engine for C. In: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, vol. 30(5), pp. 263–272. ACM (2005), doi:10.1145/1081706.1081750
13. Sen, K., Agha, G.: CUTE and jCUTE: Concolic Unit Testing and Explicit Path Model-Checking Tools. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 419–423. Springer, Heidelberg (2006)
14. Spillner, A., Linz, T., Schaefer, H.: *Software Testing Foundations*. Rocky Nook, California (2006)
15. Wang, T., Roychoudhury, A.: Dynamic slicing on java bytecode traces. *ACM Transactions on Programming Languages and Systems* 30(2) (2008), doi:10.1145/1330017.1330021
16. Williams, N., Marre, B., Mouy, P., Roger, M.: PathCrawler: Automatic Generation of Path Tests by Combining Static and Dynamic Analysis. In: Dal Cin, M., Ka nliche, M., Pataricza, A. (eds.) EDCC 2005. LNCS, vol. 3463, pp. 281–292. Springer, Heidelberg (2005)

# Requirements-Driven Iterative Project Planning

Yves Wautelet<sup>1</sup>, Manuel Kolp<sup>2</sup>, and Stephan Poelmans<sup>1</sup>

<sup>1</sup>Hogeschool-Universiteit Brussel, Brussels, Belgium

<sup>2</sup>Université Catholique de Louvain, Louvain, Belgium

{yves.wautelet, stephan.poelmans}@hubrussel.be,  
manuel.kolp@uclouvain.be

**Abstract.** Organizational modeling with the i\* framework has widely been used for model-driven software development adopting a transformational approach, notably within the Tropos process. Its high-level representation elements allow to partition the software problem into adequate and manageable elements (*actors, goals, tasks, resources* and *dependencies*) leading to an agent-oriented design, and eventually an implementation with agent technologies (*JACK, Jadex, Chimera Agent, ...*). This paper proposes to use the i\* framework for iterative software planning; each of the goals from the i\* strategic dependency model are evaluated on the basis of the (high-level) *threats* they face and the expected *quality factors*. This allows to determine a priority among the model goals and “feed” an iterative template to plan the whole project realization. This framework is thus meant to be applied during the first iteration of the project for model-driven software project management. The development of a production management system in the steel industry is used as an example.

## 1 Introduction

Due to benefits and advantages such as efficient software project management, continuous organizational modeling and requirements acquisition, early implementation, continuous testing and modularity, iterative development is more and more adopted by software engineering professionals especially through methodologies such as the *Unified Process* inspired ones (*RUP, OpenUP, EUP, AUP, ...* [8–10, 12]). The idea governing the iterative approach is to decompose a software development process into a series of manageable entities avoiding to face as a whole, every aspect of the project. Moreover, variable and non-measurable effort is spent on each of the engineering disciplines during every iteration for fast prototyping and early testing. Obviously, the most risky issues i.e., the most difficult to be expressed by the users, understood by the analysts or those addressing the most sensitive issues (such as security, flexibility, adaptability,...) receive highest priority. Consequently, these “critical” questions are dealt with first so that the development team receives feedback from users and from the whole system environment early on. This improves the probability of adequately meeting user requirements and getting the right adoption of the system into its environment.

Most software methodologies based on the agent paradigm use a pure waterfall software development life cycle (SDLC) or advise their users to repeat stages “iteratively” during the project in an *ad-hoc* way. One main reason resides in the fact that no

theoretical framework to support this way of proceeding has ever been defined and published. We believe that iterative development requires a formal or semi-formal managerial framework to support dynamic requirements and risk-driven development planning so that we require an adequate way of breaking the software problem into independent manageable entities and then to prioritize them to plan their development and evaluate their achievement.

Tropos [4] is an agent-oriented requirement-driven methodology that uses the  $i^*$  ( $i$ -star) modeling framework [17, 18] during the analysis stage;  $i^*$  defines advanced organizational modeling features and semantics in the form of agents, goals, tasks and resources. This allows to partition a software problem on the basis of the agent paradigm which is the main reason why we have chosen to extend Tropos with an iterative template and use the  $i^*$  goals as fundamental entities to decompose the problem into several aspects. Let us note that the research is actually generic enough to be adopted to extend other agent-oriented software methodologies. Iterative planning is here based on the  $i^*$  strategic dependency diagram's goals. Moreover, when planning a project with an iterative SDLC, one needs a generic process template. For this matter, we define, in this paper, an "UP-compliant" reference framework in line with existing theory on iterative SDLCs. The contributions of this paper include this iterative template and a planning method illustrated with a running example based on the development of a production management system in the steel industry.

The paper is structured as follows: Section 2 overviews a proposal for an iterative template for Tropos developments. Section 3 presents a *model-driven architecture* method for iterative planning in the context of I-Tropos developments. This method deals with threats and quality factors as fundamental criterias for goal prioritization. Section 4 points to related work and finally Section 5 gives the reader a conclusion.

## 2 Iterative Template

The first proposal of this paper is to adopt the traditional Tropos models and stages to define a project management template used to drive the software process. We also propose a common engineering terminology.

An "I-Tropos development" is an extension of the Tropos methodology, made of disciplines iteratively repeated while the relative effort spent on each one is variable from one iteration to another. The phase and discipline notions are often presented as synonyms in the software engineering literature. Indeed, Tropos is described in [4] as composed of five phases (*Early Requirements*, *Late Requirements*, *Architectural Design*, *Detailed Design* and *Implementation*). However, the Software Process Engineering Metamodel [11] defines a discipline as *a particular specialization of Package that partitions the Activities within a process according to a common "theme"*, while a phase is defined as *a specialization of WorkDefinition such that its precondition defines the phase entry criteria and its goal (often called a "milestone") defines the phase exit criteria*. The Unified Process [12] defines disciplines as *a collection of activities that are all related to a major "area of concern"* while the phases here are not the traditional sequence of requirements analysis, design, coding, integration, and test. They are completely orthogonal to the traditional phases. Each phase is concluded by a major

*milestone*. In order to be compliant with the most generic terminology, traditional Tropos phases will be called disciplines in our software process description since “they partition activities under a common theme”. In the same way, phases will be considered as groups of iterations which are workflows with a minor milestone.

In I-Tropos, the Organizational Modeling and Requirements Engineering disciplines respectively correspond to Tropos’ Early and Late Requirements disciplines. The Architectural and Detailed Design disciplines correspond to the same stages of the traditional Tropos process. I-Tropos includes core disciplines, i.e., *Organizational Modeling, Requirements Engineering, Architectural Design, Detailed Design, Implementation, Test and Deployment* but also support disciplines to handle the project development called *Risk Management, Time Management, Quality Management and Software Process Management*. There is little need for support activities in a process using a waterfall SDLC since the core disciplines are sequentially achieved once for all. However, for an iterative process, the need for support disciplines to manage the whole software project is from primary importance to precisely understand which project aspect to work on (and through which activity) at a specific time and with the best resources. I-Tropos process’ disciplines are described extensively in [15].

Using an iterative SDLC implies repeating process’ disciplines many times during the software project. Each iteration belongs to one of the phases usually four to six depending on the process itself, four in our case. We relate to the phases defined in the UP-based methodologies (RUP, OpenUP, EUP, ...) but are redefined here to better match with the Tropos specificities. These phases are achieved sequentially and have different goals assessed at milestones through knowledge and achievement oriented metrics. Phases are informally described into the next section. Figure 1 offers a two dimensional view of the I-Tropos process depicting the disciplines on Y-axis and the four different phases they belong to on X-axis.

## 2.1 Core Disciplines

The I-Tropos process has been fully described using the Software Engineering Process Metamodel in [15] that details each process’ Discipline, Activity, Role, WorkDefinition and WorkProduct, so that it can be used as a reference, template, pattern or guide for managing the system project. A lightened overview is given below. As already pointed out, the first four disciplines are inspired by the Tropos original stage.

- the *Organizational Modeling* discipline aims to understand the problem by studying the existing organizational setting;
- the *Requirements Engineering* discipline extends models created previously by including the system to-be, modeled as one or more actors;
- the *Architectural Design* discipline aims to build the system’s architecture specification, by organizing the dependencies between the various sub-actors identified so far, in order to meet functional and non-functional requirements of the system;
- the *Detailed Design* discipline aims at defining the behavior of each architectural component in further detail;
- the *Implementation* discipline aims to produce an executable release of the application on the basis of the detailed design specification;

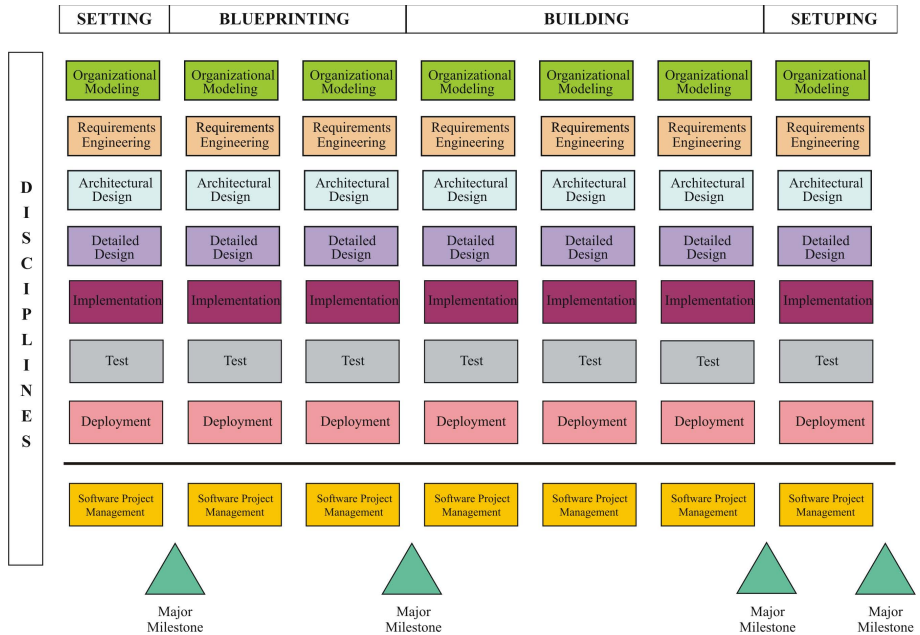


Fig. 1. I-Tropos: Iterative Perspective

- the *Test* discipline aims on evaluating the quality of the executable release;
- the *Deployment* discipline aims to test the software in its final operational environment.

## 2.2 Support Disciplines

These support disciplines provide features to support the software development on a particular project i.e., tools to manage threats, quality factors, time, effort, resources allocation but also the software process itself. All those features can be regrouped onto the term software project management [11].

- *Risk Management* is the process of identifying, analyzing, assessing risk as well as developing strategies to manage it. Strategies include transferring risk to another party, avoiding risk, reducing its negative effects or accepting some or all of the consequences of a particular one. Technical answers are available to manage risky issues. Choosing the right mean to deal with particular risk is a matter of compromise between level of security and cost. This trade-off requires an accurate identification of the threats as well as their adequate evaluation;
- *Quality Management* is the process of ensuring that quality expected and contracted with clients is achieved throughout the project. Strategies include defining quality issues and the minimum quality level for those issues. Technical answers are available to reach quality benchmarks. Choosing the right mean to deal with quality issues is a matter of compromise between level of quality and cost. This trade-off



requires an accurate identification of the quality benchmarks as well as their adequate evaluation;

- *Time Management* is the process of monitoring and controlling the resources (time, human and material) spent on the activities and tasks of a project. This discipline is of primary importance since, on the basis of the risk and quality analyses, the global iterations time and human resources allocation are computed; they are revised during each iteration;
- *Software Process Management* is the use of process engineering concepts, techniques, and practices to explicitly monitor, control, and improve the systems engineering process. The objective of systems' engineering process management is to enable an organization to produce system/segment products according to plan while simultaneously improving its ability to produce better products. In this context, Software Process Management regroups the activities aimed to tailor the generic process onto a specific project as well as improving the software process.

### 2.3 Process Phases

I-Tropos phases are inspired by UP-based processes and their milestones are based on the metrics from [3]; each phase is made of one or more iterations. Disciplines are conducted through the phases sequentially. Each phase has its own goal:

- the *Setting* phase is designed to identify and specify most stakeholders requirements, have a first approach of the environment scope, identify and evaluate project's threats and identify and evaluate quality factors;
- the *Blueprinting* phase is designed to produce a consistent architecture for the system on the basis of the identified requirements, eliminate most risky features in priority and evaluate blueprints/prototypes to stakeholders;
- the *Building* phase is designed to build a working application and validate developments;
- the *Setuping* phase is designed to finalize production, train users and document the system.

## 3 Iterative Planning

This section describes a method for planning Iterative Tropos developments. The relevant disciplines are illustrated on a running example, the development of an enterprise information system in the steel industry.

### 3.1 Running Example: Coking Process

CARSID, a steel production company located in the Walloon region, is developing a production management software system for a coking plant. The aim is provide users, engineers and workers with tools for information management, process automation, resource and production planning, decision making, etc. Coking is the process of heating coal into ovens to transform it into coke and remove volatile matter from it. Metallurgical Coke is used as a fuel and reducing agent in the production of iron, steel,

ferro-alloys, elemental phosphorus, calcium carbide and numerous other production processes. It is also used to produce carbon electrodes and to agglomerate sinter and iron ore pellets. The production of coke is one of the steps of steel making but further details about other phases of the production process are not necessary to understand the case study.

### 3.2 Agents for Steel Making

First of all, one question must be answered: *How can an industrial domain such as the steel industry that seems to belong to the past be interested in agent technologies? In other words why agent-oriented modeling (and development) would be more indicated than traditional - possibly object - technologies?*

The steel industry is by essence an agent-oriented world. Indeed, factories as a coking plant or a blast furnace are made of hundreds of different types of agents: *software agents, machines, automates, humans, sensors, releases, effectors, controllers, pyrometers, mobile devices, conveying belts*, etc. These are agents in the sense that:

- they are autonomous and dedicated to specific tasks;
- they are situated in a physical environment;
- they can act upon their environment if this is necessary by warning users, proposing solutions or taking autonomous action.

The whole I-Tropos project profile is illustrated in Figure 2. Rectangles represent the relative effort spent on each of the disciplines during each of the phases. Typically, the reader can notice that the effort spent on analysis disciplines (organizational modeling and requirements engineering) is higher into the setting and blueprinting phases and marginal for the building and setuping ones. On the contrary, the design (architectural design and detail design) and implementation ones are marginal for the setting phase (at the early beginning of the project) and higher in the blueprinting and building phases. The project management disciplines (i.e., the support disciplines) are addressed on a continuous basis with a stronger focus early on in the project (during the setting phase). We mostly concentrates in this paper on disciplines and activities performed during the setting phase since the focus is to describe a method for iterative planning. More precisely concerning the engineering disciplines, we will only focus here on organizational modeling and requirements engineering since they are the ones required for model-driven planning. The support disciplines will be covered in detail to depict the process of goal prioritization and development planning. However, the support discipline *Software Process Management* is not covered here since it goes into too many low-level details than we can afford in this research paper.

### 3.3 Engineering Disciplines

The i\* framework can be evaluated on a series of nine features following [14]: *refinement, modularity, repeatability, complexity management, expressiveness, traceability, reusability, scalability* and *domain applicability*. Those features are exhaustively assessed on the basis of a *not supported/not well supported/well supported* scale. Notably

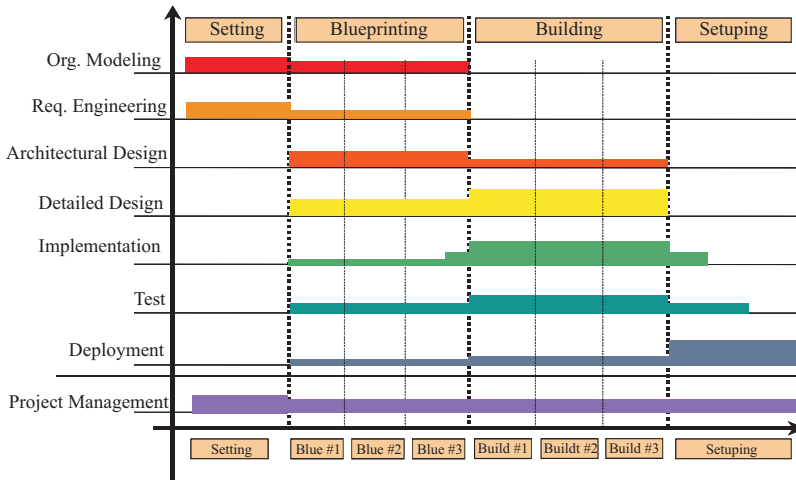


Fig. 2. I-Tropos profile for the Carsid Coking Plant project

they enlighten what is clearly needed to extend the  $i^*$  framework with mechanisms to manage granularity and refinement. Indeed, [14] points out the lacks of mechanisms in  $i^*$  for defining *granules* of information at different abstraction levels to structure, hierarchise or aggregate the semantics represented in the model. One of the flaws of  $i^*$  is actually that all of the organizational modeling elements are represented on a unique abstraction level with poor hierarchy and composition/aggregation. Moreover, except for specifying abstract primitives as building blocks, analysts must be provided with guidelines to model a complete business setting through a set of organizational processes. These building entities could then be enriched into a set of more specific components that capture a certain organizational behavior.

These discussions are from primary concern in the perspective of finding scope elements, i.e., primary abstractions to drive the whole development process including support disciplines rather than just software engineering ones. Instead of defining a new model reaching those criteria, as proposed in [6, 14, 16], we rather prefer to provide guidelines to the software analysts in order to specify an  $i^*$  strategic dependency model where each of the goals can be taken as input into the I-Tropos process. Those include:

- a goal must be defined at the highest abstraction level such as the organizational and strategic levels. Typically a scope element should describe a conceptual process, e.g., one business process so that a goal must encapsulate a high level service provided by the enterprise;
- lower level processes must be represented as tasks and a task must always be a refinement of a goal;
- goals are expressed independently, overlaps must be avoided and, if not possible, eventual redundancy among tasks of two different goals are addressed at a lower level.

By respecting those rules, all the goals of the  $i^*$  strategic dependency diagram (SDD) are scope elements for breaking down the software project into manageable parts and

are taken as input in the support disciplines as shown in the next sections. Figure 3 (also reproduced at the end of the document) represents the SDD model built up following those rules on the CARSID case study. Due to the lack of space we cannot detail each of the diagram elements here, nevertheless the reader should pay attention to the facts that:

- *Human actors* are represented as circles, for example the *FADS Team* is the team in charge of handling the coal reception;
- *Equipment actors* are also represented as circles, for example the *Coke Car* is the automotive wagon that transports the red-hot coke to the *Quenching Tower*;
- *Resources* are represented as rectangles, for example *Coal* is the raw coal received at the coking plant;
- *Goals* represented as rounded rectangles, for example *Baking* is the process for which the *Oven Team* supervises the baking of the Coal Charge in the Oven;
- *Softgoals* represented as clouds, for example *Quality Coke* represent the willingness of Management to insure that the Coke produced in the coking plant is good as the steel quality depends on the Coke quality;
- *Tasks* are represented as hexagonal forms, for example *Bake* is a sub-process of the *Baking* goal only concerned with the physical transformation of the *Coal Charge* into *Coke*.

### 3.4 Risk Management

The Risk Management discipline uses the goals identified into the organizational modeling and requirements analysis disciplines as fundamental scope elements. Consequently the identified threats are evaluated on the basis of their impact on these elements. We define a threat as *an event that can negatively affect the proper resolution of a goal or that can be the result of the misuse of a goal execution both in terms of goal achievement and degradation of quality*. A threat is expressed as an aggregate risk with a quantification of the negative impact and a occurrence probability. A threat is later refined into a series of softgoals with respect to the transformation process.

Risk analysis was done in collaboration with stakeholders estimating and validating the possible threats impact. Risk quantification is done on a double basis. Firstly the general threat weight is estimated on the basis of the impact it can have on the project under development, the system-to-be or the concerned organization. Secondly, the involvement of each goal with respect to the risk evoked is evaluated following a *Low/Medium/High* scale. This has led to the identification of six categories of threats:

- *Requirements poorly understood*: the system does not run as expected. This threat is particularly faced by user-intensive software applications. This kind of risk has a weight of 3 since huge resources can be devoted to produce an inadequate system;
- *Facility damage*: it concerns the damages caused to production facilities. A representative example is when the pusher machine pushes while hot coke is stuck in the oven, resulting in damaging the pusher machine and/or the oven's walls. This kind of risk has a weight of 2 since facilities repairs are cost-intensive;

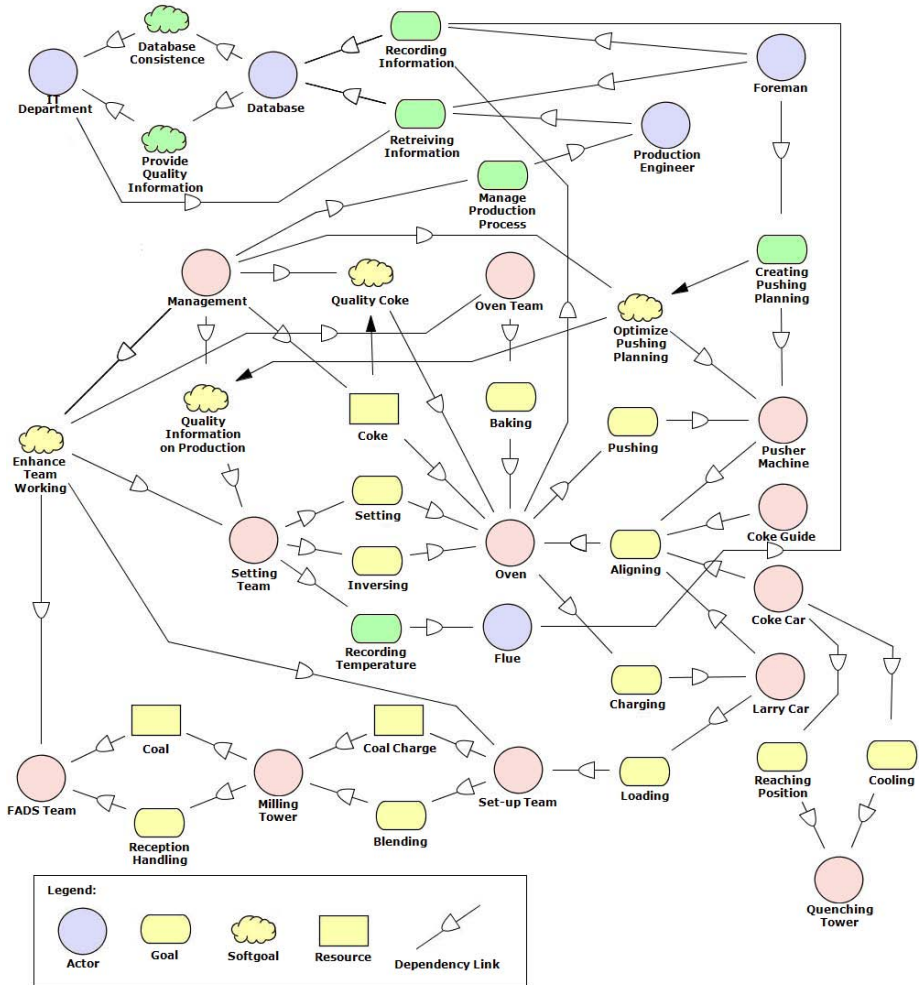


Fig. 3. Organizational Modeling

- *Mechanical error*: it concerns errors due to machinery dysfunction. For instance, a failure in the coke car engine, making impossible to cool down the coke. This risk has a weight of 1 since it will delay production;
- *Human injuries*: it concerns injury (or even death) of the staff and/or workers. A coking plant is a hazardous place; a replacement worker was recently killed in a coke plant in Ohio, and other numerous accident of this type took place in the past. This risk has a weight of 3 since it is very costly in terms of money and reputation;
- *Human error*: it concerns errors due to human intervention. This risk has a weight of 2 since it can potentially lead to other risks.
- *System failure*: the information system is not available. The system may be down and cannot be used for a certain amount of time. This risk has a weight of 1 because it will delay production.

	Threat weight	Aligning	Baking	Blending	Charging	Cooling	Creating	Pushing	Planning	Inversing	Loading	Manage Production Process	Pushing	Reaching Position	Reception Handling	Recording Information	Recording Temperature	Retrieving Information	Setting
Requirements poorly understood	3					L					H								H
Facility damage	2	L		L						M			H		L				
Mechanical error	1	L	L	L	M	M	L	M					M	M	L				
Human injury	3				L				L	M			L	L	L				
Human error	2						M								M	H			H
System failure	1										M					L	L	L	
Data loss	2										M					M	M		
Goal Risk Exposure		1	3	1	7	2	10	5	10	18	13	5	10	13	5	10	8		

Fig. 4. Project Goals Risk Exposure

- *Data loss*: some needed data is lost or never existed. It may happen if one of the worker forgets to encode some data. This risk has a weight of 2 because it will delay production and can potentially lead to other risks.

On the basis of the organizational model of Figure 3 and the risk analysis, the matrix in Figure 4 summarizes the impact of the identified threats onto the modeled goals. A specific goal is facing a particular threat is the marked and a measure is given. For example the goal *Aligning* faces the threat *Mechanical Error* even if the probability of occurrence of the threat on the goal remains *Low*. The overall risk exposure of each goal is finally computed on the basis of the threats they faced, the intensity and the threat’s weight.

### 3.5 Quality Management

The Quality Management discipline uses the goals identified into the organizational modeling and requirements analysis disciplines as fundamental scope elements. Consequently the identified *Quality Factors* are envisaged on the basis of their impact on them. Quality factors must firstly be distinguished from traditional softgoals in the sense defined in [5]. Since we address a higher level business view of the system to-be, we need an abstraction where we can specify the quality concerns of the system rather than its states as softgoals would characterize. In this sense, softgoals describe functions of a system while quality factors do not. We define a threat as *a constraint onto one or more goals in the form of a degree of excellence*. A quality factor is later refined into a serie of softgoals in the transformation process.

Quality analysis was done in collaboration with stakeholders estimating and validating the possible quality factors impact. That work has led to the identification of six categories of quality factors:

- *Reliability*: Software Reliability is the probability of failure-prone software operation for a specified period of time in a specified environment. This quality factor deals with the question: *What level of trust can be placed in what the system does?*
- *Efficiency*: Software Efficiency deals with execution time, the later can be influenced by source code optimization, the use of performing algorithmic techniques, faster sequential execution or code parallelization. This quality factor deals with the question: *How well are resources utilized?*
- *Usability*: Software Usability is the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions. This quality factor deals with the question: *How easy is it to use?*
- *Integrity*: Software Integrity is the ability of software to resist, tolerate and recover from events that threaten its dependability. This quality factor deals with the question: *How secure is it?*
- *Testability*: Software testability is a software characteristic that refers to the ease with which some formal or informal testing criteria can be satisfied. This quality factor deals with the question: *How easy is it to verify conformance to requirements?*
- *Flexibility*: Software flexibility is the ability of a software system to adapt to change. This quality factor deals with the question: *How easy is it to modify?*
- *Interoperability*: Software interoperability is defined as the ability for multiple software components written in different programming languages and distributed across multiple platforms to communicate and interact with one another. This quality factor deals with the question: *How easy is it to interface with another system?*

Based on the organizational model in Figure 3 and the quality analysis, the matrix in Figure 5 summarizes the impact of the identified quality factors onto the modeled goals. A goal facing a particular quality factor is marked and a measure is given, for example the goal *Aligning* faces the quality factor *Efficiency* even if the concern remains *Low*. The overall quality factor exposure of each goal is finally computed on the basis of the quality factors they faced, the intensity and the quality factor's weight.

### 3.6 Time Management

The previous two sections studied the threats and quality factors impact on the goals from the SDD. This section uses the global risk and quality exposures to determine a goal priority. Indeed, within the defined iterative life cycle we will plan the goals' realization. Goals with highest priority will firstly be designed, prototyped and tested during the *Blueprinting* phase and then implemented during the *Building* phase. This allows scrutinizing the trickiest issues first so that risks are addressed early on in the project when corrective actions are easier and cheaper to put into practice.

The relevant concepts and their computations are summarized in Figure 6. Particularly, we emphasize that:

- The *Overall Risk Exposure* is the sum of all threats involvement at any level on one goal;
- The *Total Risk Exposure* is the sum of the *Overall Risk Exposure* of all the project goals;

	Quality Factor Weight	Aligning	Baking	Blending	Charging	Cooling	Creating Pushing Planning	Inversing	Loading	Manage Production Process	Pushing	Reaching Position	Reception Handling	Recording Information	Recording Temperature	Retrieving Information	Setting
Reliability	3	M			L		M			M	H	M		H	M	H	L
Efficiency	2	L					H			H	M	L		L	L	M	M
Usability	1						M			M							M
Integrity	3						L			M				H	M	M	
Testability	1		L	L		L	M	L	L	M		M	L				M
Flexibility	2	M			M	L	H		L	M	M	M					M
Interoperability	2	M	L		L	L	L		L	M		L		M	M		L
Goal Quality Exposure		16	3	1	11	5	31	1	5	32	20	16	1	30	18	26	14

Fig. 5. Project Goals Quality Issues

Goal	Overall Risk Exposure	Relative Risk Exposure	Overall Quality Exposure	Relative Quality Exposure	Priority Level	Goal Priority	Goal Complexity	Goal Complexity Weight	Precedence
1 Aligning	1	0.008264463	16	0.069565217	0.023589651	13	L	5	
2 Baking	3	0.024793388	3	0.013043478	0.021855911	14	L	5	
3 Blending	1	0.008264463	1	0.004347826	0.007285304	16	M	10	
4 Charging	7	0.05785124	11	0.047826087	0.05344951	9	M	10	
5 Cooling	2	0.016528926	5	0.02173913	0.017831477	15	M	10	
6 Creating Pushing Planning	10	0.082644628	31	0.134782609	0.095679123	4	H	15	G4, G10
7 Inversing	5	0.041322314	1	0.004347826	0.032078692	12	L	5	
8 Loading	10	0.082644628	5	0.02173913	0.067418254	6	M	10	
9 Manage Production Process	18	0.148760331	32	0.139130435	0.146352857	1	H	15	
10 Pushing	13	0.107438017	20	0.086956522	0.102317643	3	H	15	
11 Reaching Position	5	0.041322314	16	0.069565217	0.04838304	11	M	10	
12 Reception Handling	10	0.082644628	1	0.004347826	0.063070428	8	L	5	
13 Recording Information	13	0.107438017	30	0.130434783	0.113187208	2	L	5	
14 Recording Temperature	5	0.041322314	18	0.07826087	0.050556953	10	L	5	
15 Retrieving Information	10	0.082644628	26	0.113043478	0.090244341	5	M	10	G13
16 Setting	8	0.066115702	14	0.060869565	0.064804168	7	M	10	
Sum	121	1	230	1	1			145	

Fig. 6. Goal Prioritization

- On the basis of the *Overall Risk Exposure*, the *Relative Risk Exposure* is computed using the formula:  $RelativeRiskExposure = \frac{OverallRiskExposure}{TotalRiskExposure}$ ;
- The *Overall Quality Exposure* is the sum of all the levels of involvement of all the quality factors on one goal;
- The *Total Quality Exposure* is the sum of the *Overall Quality Exposure* of all the project goals;
- Based on *Overall Quality Exposure*, the *Relative Quality Exposure* is computed with the formula:  $RelativeQualityExposure = \frac{OverallQualityExposure}{TotalQualityExposure}$ ;



	Goal	Priority	Complexity	Elaboration 1	Elaboration 2	Elaboration 3	Construction 1	Construction 2	Construction 3
1	Manage Production Process	1	H	X			X		
2	Recording Information	2	L	X			X		
3	Pushing	3	H	X			X		
4	Charging	9	M	X			X		
5	Creating Pushing Planing	4	H		X			X	
6	Retrieving Information	5	M		X			X	
7	Loading	6	M		X			X	
8	Setting	7	M		X			X	
9	Reception Handling	8	L		X			X	
10	Recording Temperature	10	L			X			X
11	Reaching Position	11	M			X			X
12	Inversing	12	L			X			X
13	Aligning	13	L			X			X
14	Baking	14	L			X			X
15	Cooling	15	M			X			X
16	Blending	16	M			X			X
Total Effort Weight			145	15	16,66667	16,66667	30	33,33333	33,33333

Fig. 7. Iteration Plan

- The *Priority Level* is computed by "balancing" the *Relative Risk Exposure* and the *Relative Quality Exposure*. For this particular project we chose a repartition key of 75 percent for the risk component and 25 for the quality component. This repartition key can vary from one project to another and should be calibrated from data collected on a large number of projects as well as considering the working team experience;
- On the basis of the *Priority Level*, each *Goals Priority* is deduced.

On the basis of the *Goal Priority* list, the *Precedence* constraints and the estimated *Goal Complexity* we instantiate the iteration template defined in Section 2. The process is summarized in Figure 7. The Goal complexity estimates the amount of effort required to develop it. I-Tropos uses three categories of goal complexity, i.e., *Low/Medium/High*, owning respectively a weight of 5/10/15. Transforming the overall weight to man-month requires calibration typically based on a regression model using statistics from large numbers of projects and remains an open issue. The proposed planning will be subject to modifications/reviews during the software project supported by users and environmental feedbacks.

## 4 Related Work

Numerous agent-oriented software development methodologies have been proposed in the past twenty years. We overview hereafter some of them with a particular focus on iterative SDLC and software project management. Tropos [4] offers the most advanced agent-based modeling features through the *i\** models, one of the reasons we use it as a basis for the process presented in this paper. Some papers related to this methodology propose to develop a software system in an iteratively way but no supporting theoretical framework has ever been proposed. Gaia [19], one of the most popular methodologies

due to its simple and clear process as well as its neutrality with respect to implementation techniques or platforms has been extended with an iterative SDLC in [7]. The main drawback of their proposal is that they decompose functionality on the basis of design models (called “parts”) and not analysis ones. Moreover, the priority given to those functional parts is done in an *ad hoc* manner since no framework is given to evaluate the criticality of these “functional parts”. There is also no real template provided for cutting out the project into phases so that each iteration can be considered as a “sub-waterfall project” with no rapid prototyping for testing as implied by the cut out between blueprinting and building phases in our framework. ADELFE [2] claims to follow the RUP but no guideline or planning method is actually given for that purpose. The process is depicted in [2] in a waterfall manner. Finally, MASSIVE [13] uses an *Iterative View Engineering* approach itself based on *Iterative Enhancement* and *Round-trip Engineering*. From the authors’ point of view, the method can be said to be incremental rather than iterative. As a matter of fact, it is founded on producing hybrid models to be tested and enhanced later on into the project on the basis of users’ feedback. However, the software project is not broken down into manageable elements that are prioritized and worked on during multiple iterations so that no software project management framework is required to manage the SDLC.

## 5 Conclusions

Since the emergence of spiral development, software engineering professionals have understood the benefits of iterative approaches. However, poor guidelines and too simple project management frameworks have failed to provide practitioners with clear methods to deal with such SDLCs in a structured manner. Moreover, *Model-Driven Architecture (MDA)* and *Model-Driven Development (MDD)* are extensively used in the software transformation process, i.e., the process of tracing high level analysis elements into design and implementation ones. This principle is applied here but analysis models are used to feed the process at a managerial level for *goal-driven project management*. Contributions thus include a structured (model-driven) method for iterative lifecycle management.

The process needs however to be refined and better calibrated on the basis of experience gained from multiple projects. It has recently been applied on the development of a collaborative supply chain management platform from which an in-depth analysis of the empirical results will be presented soon. A CASE-Tool called *DesCARTES Architect* supporting all of the methodology’s models and I-Tropos life cycle management has also been developed to better operationalize the method. The present research is clearly driven by the willingness to bring agent-based development to large enterprise software developments in order to propose an integrated process with clear guidelines supported by practical tools. Finally and as just mentioned, the method was used here in the context of agent-oriented development but the transformation process from *i\** models can lead to an object implementation. Similarly, the goal planning method can be very easily and flexibly adapted to (semantically poorer) UML use-cases and serve RUP practitioners in their everyday iterative development planning.

## References

1. Anonymous. Software & systems process engineering meta-model specification. version 2.0. Technical report, Object Management Group (2008)
2. Bernon, C., Gleizes, M.P., Picard, G., Glize, P.: The adelfe methodology for an intranet system design. In: AOIS@CAiSE (2002)
3. Boehm, B.: Software Project Management. Addison-Wesley (1998)
4. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the tropos project. *Inf. Syst.* 27(6), 365–389 (2002)
5. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Kluwer Academic Publishing (2000)
6. Estrada, H., Rebolgar, A.M., Pastor, O., Mylopoulos, J.: An Empirical Evaluation of the *i\** Framework in a Model-Based Software Generation Environment. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 513–527. Springer, Heidelberg (2006)
7. Gonzalez-Palacios, J., Luck, M.: Extending Gaia with Agent Design and Iterative Development. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 16–30. Springer, Heidelberg (2008)
8. IBM. The rational unified process. Rational Software Corporation, Version 2003.06.00.65 (2003)
9. Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley (1999)
10. Jacobson, I., Bylund, S.: The road to the unified software development process. Cambridge University Press (2000)
11. Jalote, P.: Software Project Management in Practice. Addison Wesley (2002)
12. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley (2003)
13. Lind, J.: The MASSIVE Method. LNCS (LNAI), vol. 1994. Springer, Heidelberg (2001)
14. Pastor, O., Estrada, H., Martínez, A.: The strengths and weaknesses of the *i\** framework: an experimental evaluation. In: Giorgini, P., Maiden, N., Mylopoulos, J., Yu, E. (eds.) Social Modeling for Requirements Engineering. MIT Press (2011)
15. Wautelet, Y.: A goal-driven project management framework for multi-agent software development. PhD thesis, Université catholique de Louvain, Belgium (2008)
16. Wautelet, Y., Achbany, Y., Kolp, M.: A service-oriented framework for mas modeling. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (3-1), pp. 120–128 (2008)
17. Yu, E.: Modeling strategic relationships for process reengineering. PhD thesis, University of Toronto, Department of Computer Science, Canada (1995)
18. Yu, E.: Social Modeling for Requirements Engineering. MIT Press (2011)
19. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.* 12(3), 317–370 (2003)

# An Approach for Model-Driven Design and Generation of Performance Test Cases with UML and MARTE

Antonio García-Domínguez<sup>1</sup>, Inmaculada Medina-Bulo<sup>1</sup>,  
and Mariano Marcos-Bárcena<sup>2</sup>

<sup>1</sup> Department of Computer Languages and Systems, University of Cádiz, Cádiz, Spain

<sup>2</sup> Department of Mechanical Engineering and Industrial Design, University of Cádiz,  
Cádiz, Spain

{antonio.garciadominguez,inmaculada.medina,mariano.marcos}@uca.es  
<http://neptuno.uca.es/~agarcia>, <http://neptuno.uca.es/~imedina>

**Abstract.** High-quality software needs to meet both functional and non-functional requirements. In some cases, software must accomplish specific performance requirements, but most of the time, only high-level performance requirements are available: it is up to the developer to decide what performance should be expected from each part of the system. In this work, we show several algorithms that infer the required throughput and time limits for each action in an UML activity diagram from a global constraint and some optional local annotations. After studying their theoretical and empirical performance, we propose an approach for generating performance test cases from the activity diagram after it has been implemented as code. Our approach decouples the performance analysis model from the implementation details of the code to be tested.

**Keywords:** Model-driven engineering, Performance testing, UML, MARTE, Non-functional requirements, Model weaving.

## 1 Introduction

In addition to functional requirements, software must meet non-functional requirements. Among them, performance plays a major role in shaping the user experience. In some cases, meeting specific performance requirements is critical. This is the case not only in soft and hard real-time systems, but also in service-oriented architectures [13], where Service Level Agreements (SLAs) may have been signed between the provider and the consumer of a service.

For these reasons, there has been considerable work in estimating and measuring the performance of software systems [28]. Estimating the performance of a prospective system usually requires building high-level execution and architecture models and deriving a formalism from them, as in [25][30], among many others. Measuring the performance of a system requires instrumenting it to produce the desired results, instead of building a model. These approaches complement each other: estimations can be performed before the actual system is implemented, while measurements are more accurate.

Measuring the performance of a system can be useful for many purposes: finding performance degradations over time, identifying load patterns over specific time periods

and checking if the system is meeting its performance requirements. Obviously, this last use case requires that the performance requirements have been previously defined. However, most of the time, detailed performance requirements are not provided [27]. Developers may have to meet high-level performance requirements without a clear view of what performance is required in each part of the system.

In this work we propose a model-driven approach to deriving the low-level performance requirements of a system from high-level performance requirements. The user creates UML models annotated with a small subset of the MARTE profile [23] and runs our inference algorithms to derive the low-level requirements. After the UML models have been implemented as code, the user can weave the analysis model with an implementation model to generate the concrete performance test cases.

The rest of this paper is structured as follows: in Section 2, we introduce the MARTE profile for UML, describe the subset used in our work and show our running example. Section 3 defines the inference algorithms and outlines some of the optimisations performed. Section 4 is dedicated to analysing the restrictions imposed upon the algorithms and evaluating their performance. Section 5 describes our proposed approach for generating the concrete performance test cases. Section 6 discusses related work. Finally, Section 7 condenses the main points of this paper and lists our future lines of work.

## 2 The MARTE Profile

UML has been widely adopted as a general purpose modelling language for describing software systems. However, UML itself does not include support for modelling scheduling, performance or time aspects, among other non-functional aspects.

For this reason, the Object Management Group proposed in 2005 the SPT (Schedulability, Performability and Time) profile [21], which extended UML with a set of stereotypes describing scenarios that various analysis techniques could take as inputs. In 2008, OMG proposed the QoS/FT (Quality of Service and Fault Tolerance Characteristics and Mechanisms) profile [22], with a broader scope than SPT and a more flexible approach: users formally defined their own quality of service vocabularies and used them to annotate their models.

When UML 2.0 was published, OMG saw the need to update the SPT profile and harmonise it with other new concepts. This resulted in the MARTE (Modelling and Analysis of Real-Time and Embedded Systems) profile [23], published in 2009. Like the QoS/FT profile, the MARTE profile defines a general framework for describing quality of service aspects. The MARTE profile uses this framework to define a set of pre-made UML stereotypes, as those in the SPT profile.

In this section, we will introduce the parts of the MARTE profile required for our algorithms and show an example model, using its predefined stereotypes.

### 2.1 Selected Subset

The MARTE specification provides support for model-based analysis and design of real-time and embedded systems. Among its sub-profiles, we are interested in a subset of the GQAM (Generic Quantitative Analysis Modelling) profile. The GQAM domain

model describes the concepts of the GQAM profile using the generic non-functional property modelling framework in MARTE. The stereotypes from the GQAM profile are prefixed with “Ga” (standing for “generic analysis”), and the non-functional property types from the normative MARTE model library are prefixed with “NFP”.

The stereotype and attributes used by our algorithms are:

- «GaScenario»: *hostDemand* is used to model requirements on the CPU time to be used and *throughput* indicates how many requests should be handled per second. *respT* combines both, specifying the maximum response time when handling *throughput* requests per second.
- «GaStep»: *prob* is the probability of traversing a control flow, and *rep* is the number of times the annotated activity is repeated.
- «GaAnalysisContext»: *contextParams* contains a list of context parameters. These are variables which can be used to parametrise the annotations using VSL (Value Specification Language) expressions. VSL is a textual language defined in MARTE.

All the non-functional property types in the normative MARTE library share several traits, as they inherit from *NFP\_CommonType*. Values can be specified as literals in the *value* attribute, or as VSL expressions in the *expr* attribute. The source of a requirement (estimated, measured, calculated or required) is described by the *source* attribute.

*NFP\_CommonType* is a VSL tuple type. In this paper we will use the notation  $(key1=value1, \dots, keyN=valueN)$  for VSL tuples. For instance, a *NFP\_Duration* of 5 milliseconds required by the client is written as  $(value=5, unit=ms, source=req)$ .

## 2.2 Usage

Activities must have the «GaScenario» and «GaAnalysisContext» stereotypes. «GaScenario» indicates the expected response time (*respT*) and throughput (*throughput*) for the entire activity. «GaAnalysisContext» only lists the context parameters (*contextParams*) which represent the slack per unit of weight assigned to each action in the activity.

Control flows leaving decision nodes are annotated with the «GaStep» stereotype, specifying the probability (*prob*) of traversing one of the conditional branches. The probabilities are estimated by the user.

Actions are annotated with the «GaStep» stereotype as well. The user must indicate their expected number of repetitions (*rep*) and how the available time is to be distributed among them. *hostDemand* must contain a tuple with a VSL expression matching  $M+W*swI$ :  $M \geq 0$  is its minimum time limit,  $W \geq 0$  is its weight and *swI* is its context parameter. The time limit inference algorithm will set *swI* to the slack per unit of weight assigned to that action.

After the algorithms are done, results are fed back into the activity diagram, replacing those from previous runs. Actions are annotated with the inferred time limits in *hostDemand*, and with the inferred throughputs in *throughput*. Context parameters are set to the slack per unit of weight assigned to their actions.

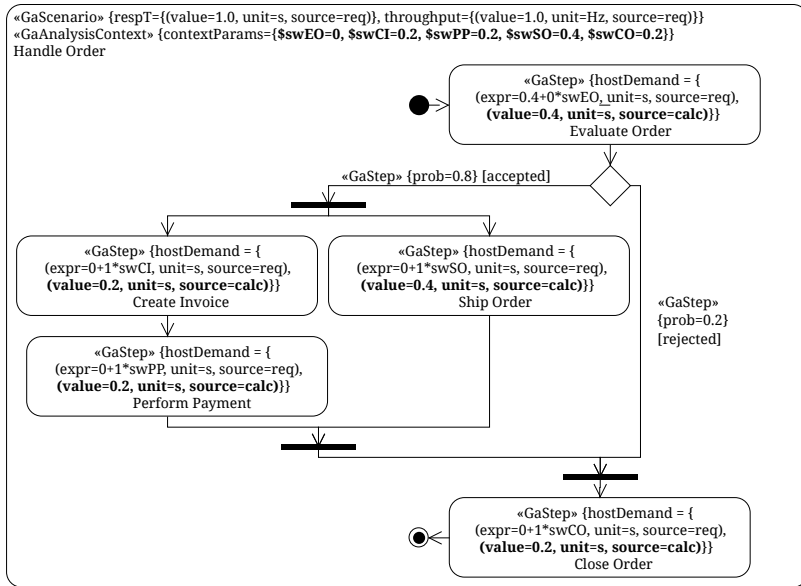


Fig. 1. Running example after inferring time limits

### 2.3 Running Example

Figure 1 shows the UML activity diagram which we will use as running example for the rest of this paper. Its activity, “Handle Order”, describes how to process a specific order: first, the order is evaluated. If rejected, we simply close the order. If accepted, we fork into two execution branches: one creates the shipping order and sends it to the shipping partner, and the other creates the invoice, sends it to the customer and receives the payment. Once both branches are done, the order is closed and we are done.

According to the MARTE annotations, the activity should complete its execution in one second when receiving one request per second. Most of the actions have no minimum time limit and weight equal to 1, except for “Evaluate Order”, whose CPU time is fixed by the modeller to 0.4s. All actions are run once, to simplify the discussion. The user has estimated that 80% of all orders are accepted. The annotations in bold have been inferred by our algorithms, and will be described more in depth in Section 3.2.

## 3 Inference Algorithms

In the previous section, we explained how we used the MARTE profile for our algorithms and described the running example for this paper (Figure 1). In this section we will outline the algorithms themselves. The first algorithm computes the expected throughput of each action, and the second algorithm computes the time limit for each action. They improve upon those in [16].

Both require that activities do not contain cycles, that they only have one initial node, and that all their actions are reachable from it. Let us define some terms:

- $s(e)$  and  $g(e)$  are the source and target vertex of the edge  $e$ , respectively.
- $i(n)$  and  $o(n)$  are the incoming and outgoing edges of the node  $n$ , respectively.
- $L > 0$  is the expected response time (the global time limit) of the activity.
- $c(n) = (m(n), w(n)) \in C(L)$  is the *constraint* of the node  $n$ , where  $m(n)$  is the minimum time limit of  $n$  and  $w(n)$  is its weight (see Section 2.2). The set of all valid constraints with  $L$  as global time limit is  $C(L) = \{(m, w) \mid 0 \leq m \leq L, w \geq 0\}$ .
- Each path  $p$  also has a constraint,  $c(p) = (m(p), w(p)) \in C(L)$ , with  $m(p) = \sum_{n \in p} m(n)$  and  $w(p) = \sum_{n \in p} w(n)$ .
- A node  $n$  is run  $R(n) \geq 1$  times (once by default).

### 3.1 Throughput Inference

We will define  $T$  as a function from a node or edge to its expected throughput. For a control flow  $e$ ,  $T(e) = P(e)T(s(e))$ , where  $P(e)$  is the probability of traversing  $e$ .

For a node  $n$ , the actual formula depends on its type. For an initial node,  $T(n)$  is the expected throughput of the activity. For a join node,  $T(n) = \min_{e \in i(n)} T(e)$ , since requests in the least performing branch set the pace. For a merge node,  $T(n) = \sum_{e \in i(n)} T(e)$ , as requests from mutually exclusive branches are reunited. For any other type of node,  $T(n) = T(e_1)$ , where  $e_1 \in i(n)$  is its only incoming edge.

Using these formulas, computing  $T(\text{Create Invoice})$  for the example shown in Figure 1 requires walking back to the initial node, finding an edge with a probability of 0.8, no merge nodes and an initial node receiving 1 request/second. Therefore, it would be equal to  $pL = 0.8$ .

To compute these values efficiently, the expressions are evaluated in a topological traversal of the graph. For each action  $a$ , *throughput* will contain a single tuple of the form  $(\text{value}=T(a), \text{unit}=\text{Hz}, \text{source}=\text{calc})$ .

### 3.2 Time Limit Inference

Inferring the time limits of each action inside an activity is considerably more complex than inferring their required throughputs. After more definitions, we will describe the algorithm, and then apply it to the running example in Figure 1.

**Preliminaries.** The algorithm adds a  $(\text{value}=t(n), \text{unit}=\text{s}, \text{source}=\text{calc})$  tuple to the attribute *hostDemand* of each action node  $n$ , where  $t(n)$  is its inferred time limit. The algorithm also updates the appropriate context parameter with the final slack per unit of weight distributed to  $n$ .

Let  $I$  be the initial node of the activity being annotated and let  $P_S(n)$  contain all paths from the node  $n$  to a final node.  $t(n)$  must meet the following constraints:

- For every action  $n$ ,  $t(n) \geq m(n)$ : the assigned time limit must be greater or equal than the minimum set by the user.
- For every path  $p$  in  $P_S(I)$ ,  $\sum_{n \in p} R(n)t(n) \leq L$ : the sums of the time limits over each path meet the global time limit.

The available time “flows” from the initial node. If a node  $n$  receives  $0 \leq r(n) \leq L$  seconds, every path  $p \in P_S(n)$  receives  $r(p) = r(n)$  seconds to distribute among its nodes.  $r(n)$  is not known *a priori* except for the initial node:  $r(I) = L$ .



If the «GaStep» and «GaScenario» annotations are consistent with each other, then  $r(p) \geq m(p)$  for every path  $p$ : the minimum time constraints of all actions are always met.  $s(p) = r(p) - m(p) \geq 0$  is known as the *slack* of the path  $p$ .  $s(p)$  is distributed over  $p$  according to the weight of each node: the *slack per unit of weight* initially assigned to each node is  $S_w(p) = s(p)/w(p)$ . When  $w(p) = 0$ , we assume that  $S_w(p) = 0$ : all nodes in  $p$  have a zero weight, so no slack can be distributed.

The algorithms must ensure that  $w(p) > 0 \Rightarrow s(p) > 0$ , so every path  $p$  with a non-zero weight has some slack to distribute. If this condition is not met or the annotations are inconsistent, the user should be notified and every change should be rolled back.

**Definition.** The algorithm is a recursive function, taking a node  $n$  and the time it receives,  $r(n)$ . Initially,  $n = I$  and  $r(n) = L$ , the global time limit. Its steps are as follows:

1. Select two paths from  $P_S(n)$ :
  - $p_{ms}(n)$  has the minimum  $S_w(p)$  when  $r(n)$  seconds are available. In case of a tie, pick the path with the maximum  $w(p)$ .
  - $p_{Mm}(n)$  has the maximum  $m(p)$ .
2. If  $s(p_{Mm}(n)) < 0$ , the minimum time limits cannot be satisfied: abort.
3. If  $s(p_{ms}(n)) = 0$  and  $w(p_{ms}(n)) > 0$ , there is no slack in a path with a non-zero weight: abort.
4. Set the time limit of  $n$ ,  $t(n)$ , to  $m(n) + S_w(p_{ms}(n))w(n)$ . The remaining time will be  $T_R = T - R(n)t(n)$  seconds. Mark  $v$  as visited.
5. Sort each edge  $e \in o(n)$  in ascending order of  $S_w(p_{ms}(g(e)))$  with  $r(g(e)) = T_R$ , the minimum slack per unit of weight when  $T_R$  seconds are available for all paths that start at the target of  $e$ .
6. Visit each edge in  $o(n)$ :
  - (a) If the target of  $e$  has been visited before, check if the time which was sent to it,  $T'_R$ , is strictly less than  $T_R$ , the time which would have been sent through  $e$ . In that case, try to reuse the surplus  $T_R - T'_R$  seconds on the source of  $e$  and its ancestors, and send  $T'_R$  seconds through  $e$ . Go back in the graph from the source of  $e$ , collecting nodes with non-zero weights into  $C$  until a node with more than one incoming or outgoing edge is found. Increase the time limit of each collected node by  $(T_R - T'_R)w(n)/w(C)$ , where  $w(C) = \sum_{n \in C} R(n)w(n)$ .
  - (b) If the target of  $e$  has not been visited before, invoke this algorithm recursively, setting  $n$  to the target of  $e$  and  $r(n) = T_R$ .
7. Set the context parameter related to  $n$  to 0 if  $w(n) = 0$ , and to  $(t(n) - m(n))/w(n)$  otherwise. This is the effective slack per unit of weight distributed to  $n$ .

**Key Optimisations.** The algorithm above uses several optimisations to improve its performance. First of all, each path  $p$  is not represented by its sequence of nodes, but by its constraint  $c(p) = (m(p), w(p))$ , saving much memory.

To select  $p_{Mm}(n)$  at each node we need to know the maximum  $m(p)$  for each path  $p \in P_S(n)$ , which we will note as  $m(p_{Mm}(n))$ . We can compute it in advance using (I). As it is recursive, we can evaluate (I) incrementally, starting from the final nodes (for which  $m(p_{Mm}(n)) = 0$ ) and going back to the initial node in reverse topological order:

$$m(p_{Mm}(n)) = R(n)m(n) + \max\{m(p_{Mm}(g(e))) \mid e \in o(n)\} \quad (1)$$

To select  $p_{ms}(n)$  at each node we need to know the strictest path starting from it. We cannot compute it in advance, as it depends on the time received by the node,  $r(n)$ , which is not known *a priori*. Instead, we remove redundant paths from  $P_S(n)$ . We will call this reduced set  $P'_S(n)$ . A path  $p_a \in P_S(n)$  is removed when it is said to be *always less or just as strict* than some other path  $p_b \in P_S(n)$ , independently of the time received by  $n$  or the common ancestors of  $p_a$  and  $p_b$ . We denote this by  $c(p_a) \preceq_{s(L)} c(p_b)$ , and define it formally as follows:

$$(a, b) \preceq_{s(L)} (c, d) \equiv \forall t \in [0, L] \forall x \in [0, L] \forall y \geq 0 \\ a + x \leq t \wedge c + x \leq t \wedge b + y > 0 \wedge d + y > 0 \Rightarrow \frac{t - (a + x)}{b + y} \geq \frac{t - (c + x)}{d + y} \quad (2)$$

We can simplify (2) into:

$$a \leq c \wedge (b \leq d \vee a < L \wedge b > d \wedge (b - d)L \leq bc - ad) \quad (3)$$

It can be proved that this defines a partial order (a reflexive, antisymmetric, and transitive binary relation) on  $C(L)$ . The proof is omitted for the sake of brevity.

Like  $m(p_{Mm}(n))$ ,  $P'_S(n)$  can also be computed incrementally by traversing the graph in reverse topological order. Let  $n_i$  be a child of  $n$  and  $p_a$  and  $p_b$  be two paths in  $P_S(n_i)$ , so  $c(p_a) \preceq_{s(L)} c(p_b)$ . By definition,  $p_a$  is less or just as strict as  $p_b$  regardless of their common ancestors, so  $\langle n \rangle + p_a$  will also be discarded from  $P'_S(n)$  over  $\langle n \rangle + p_b$ . This means that instead of comparing every path in  $P_S(n)$  for every node  $n$ , we can build  $P'_S(n)$  by adding  $n$  at the beginning of the paths in  $P'_S(n_i)$ , for every child  $n_i$  of  $n$ , and then filtering the redundant paths using  $\preceq_{s(L)}$ .

Let  $\max_{\preceq_{s(L)}} S$  select the paths in  $S$  which are not always less or just as strict than any other (maximal elements according to  $\preceq_{s(L)}$ ). We define  $P'_S(n)$  as:

$$P'_S(n) = \max_{\preceq_{s(L)}} \{ (R(n)m(n) + M, R(n)w(n) + W) \mid e \in o(n), (M, W) \in P'_S(g(e)) \} \quad (4)$$

Note that  $P_S(f) = (0, 0)$ , where  $f$  is a final node.

**Example.** Previously, we defined the algorithm and described the key optimisations performed. We will now apply the algorithm to the example in Figure 1, producing the outputs highlighted in bold. To save space, we will shorten action names to their initials: “Evaluate Order” will be simply “EO”.

First,  $m(p_{Mm}(n))$  and  $P'_S(n)$  are precomputed:

- $m(p_{Mm}(\text{CO})) = 0$ ,  $P'_S(\text{CO}) = \{(0, 1)\}$ .
- $m(p_{Mm}(\text{PP})) = 0$ ,  $P'_S(\text{PP}) = \{(0, 2)\}$ .
- $m(p_{Mm}(\text{CI})) = 0$ ,  $P'_S(\text{CI}) = \{(0, 3)\}$ .
- $m(p_{Mm}(\text{SO})) = 0$ ,  $P'_S(\text{SO}) = \{(0, 2)\}$ .
- $m(p_{Mm}(\text{EO})) = 0.4$ ,  $P'_S(\text{EO}) = \{(0.4, 3)\}$ .

After that, the algorithm sends the available second ( $L = 1s$ ) into the initial node and then into EO. EO takes 0.4s and sends the remaining 0.6 seconds through the decision

node. The next action in the strictest path is CI, which takes 0.2s and sends 0.4s into PP. PP takes another 0.2s and sends the remaining 0.2s to CO.

Once the strictest path is done, we back up and proceed with the next strictest path, sending 0.4s into SO. At first, SO takes only 0.3s, but since CO received only 0.2s before, we reuse the extra 0.1s into SO. The final time limit of SO is 0.4s. We back up and continue with the empty branch for rejected orders: we are done.

As for the context parameters:  $sw_{EO}$  is set to 0, as  $w(EO) = 0$ .  $sw_{CI}$ ,  $sw_{PP}$  and  $sw_{CO}$  are set to 0.2.  $sw_{SO}$  is set to 0.4: note that the initial slack per unit of weight for SO was 0.3, but after reusing the extra 0.1 seconds, it changed to 0.4.

## 4 Evaluation

The algorithms have been implemented using the Epsilon Object Language (EOL) [19] and integrated into the Papyrus graphical UML editors [12]. Code is available at [15]. In this section we will analyse their restrictions and performance.

### 4.1 Restrictions

The inference algorithms are limited in several ways. The most important restriction is that the graph formed by the nodes of the activity must be acyclic, which hinders the modelling of repetitive structures. We have partially addressed this issue by using the attribute *rep* of «GaStep» to indicate the expected number of repetitions of an action.

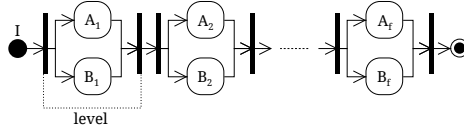
At first glance, the algorithm still requires to annotate each action with some knowledge from the modeller, so it would appear not to save much effort. However, the information annotated by the user on each activity only depends on the action (minimum time and weight) or control flow (probability) themselves, instead of all the paths they are part of. In addition, any sufficiently advanced tool can add the missing annotations with the default values set by the user. The time limit inference algorithm also ensures that the annotations are consistent with each other.

The algorithms do not take into account the fact that the same behaviour might be reused in several places: each action is assumed to be different from the rest. A simple and conservative solution would be simply taking the strictest constraint over all the occurrences of that behaviour. Integrating the “same behaviour” constraint would be interesting, but it might considerably increase the cost of the algorithm.

### 4.2 Theoretical Performance

Let us consider an activity with  $n$  nodes and  $e \in O(n^2)$  edges, with  $O(n)$  incoming edges in each node. The throughput inference algorithm is easy to analyse: by going back from the final nodes to the initial nodes, each node and edge in the activity needs to be visited exactly once. The throughput for the  $O(n)$  join and merge nodes requires evaluating an expression in constant time over their  $O(n)$  incoming edges. However, throughputs for the rest of the  $O(n+e)$  nodes and edges can be computed in constant time. Therefore, a conservative upper bound for the running time of the throughput inference algorithm is  $O(n)O(n) + O(n+e)O(1) = O(n^2)$ . The running time does not depend on the values of the annotations.

The time limit inference algorithm is harder to analyse. Its performance depends both on the structure of the graph and the values of the annotations. For this reason, we will use a specific kind of activity to frame the analysis, which we call a *fork-join activity*. As shown in Figure 2 it has an initial node,  $I$ , followed by a sequence of  $f$  “levels”. Each level has a fork node with two branches with a single action, joined before the next level. The activity has  $n = 2 + 4f \in \Theta(f)$  nodes and  $e = 1 + 5f \in \Theta(f)$  edges in total, and there are  $2^f$  paths from the initial node to the final node. These activities are inexpensive to generate, as the number of nodes and edges grows linearly. At the same time, they can represent the worst case of the algorithm, since the number of paths from the initial node to the final node grows exponentially.



**Fig. 2.** Example fork-join activity with  $f$  levels

Having defined the structure of the activities, let us analyse the worst case by parts:

- Computing  $m(p_{Mm}(n))$  in advance for each node always takes  $O(1)O(n) = O(n)$  operations, as it requires evaluating an arithmetic expression over the  $O(1)$  incoming edges of each of the  $n$  nodes.
- Computing  $P'_S(n)$  in advance for each node is actually the most expensive part of the algorithm: in the worst case,  $O(2^f)$  paths need to be considered at every node and selecting the strictest ones takes  $O(4^f)$  operations per node and  $O(n4^f)$  in total.
- The last step depends on the number of elements of  $P'_S(g(e))$  for each edge  $e$  in the graph: in the worst case,  $|P'_S(g(e))| = |P_S(g(e))|$  for every node and  $O(n2^f)$  operations are required.

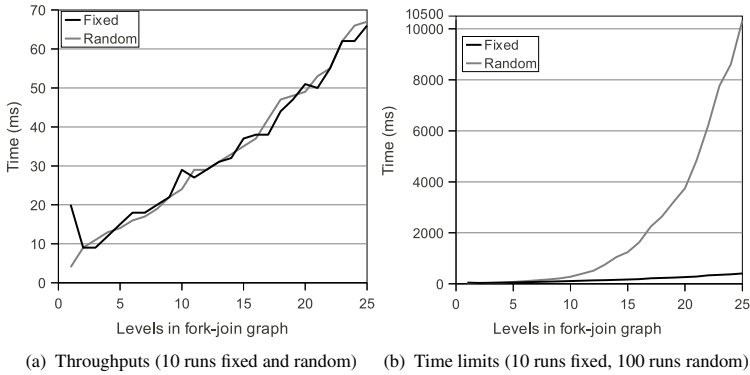
In total, we have  $O(n4^f)$  operations in the worst case, which can be very expensive.

### 4.3 Empirical Performance

Previously, we concluded that the throughput algorithm had polynomial cost regardless of the annotations, and that the time limit inference algorithm could reach exponential cost, depending on the annotations. In this section we will study how close are the average times to this absolute worst case.

We first measured the performance of the algorithms using fork-join activities with 1 to 25 levels. We ran the algorithms on these activities requiring 1s response time when 1 request was received per second. The actions were annotated in two ways: either using a fixed minimum time limit and weight (0 and 1, respectively) or using uniformly distributed random values, so the minimum time limits were consistent and weights were between 0 and 1. To simplify the analysis, each action had *rep* set to 1.

The results are shown in Figures 3(a) and 3(b). Figure 3(a) confirms that the time required for the throughput inference algorithm grows linearly, regardless of the annotations. Figure 3(b) suggests that the average times for fixed and random annotations are quite far from the  $O(n4^f)$  absolute worst case.



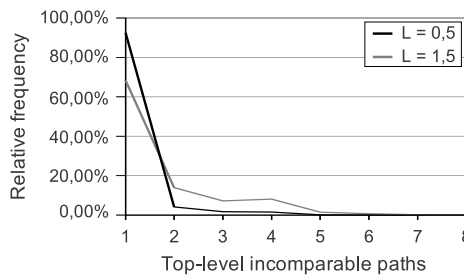
**Fig. 3.** Average running times by number of levels and type of annotation

It is interesting to note that when the minimum time limit is equal to 0 in all actions, the partial order in (3) can be simplified to  $a \leq c$ , which is a total order. Therefore, these fixed annotations are instances of the best case of the time limit inference algorithm, in which all paths are comparable. As shown in Figure 3(b), the time limit inference algorithm required 400ms on average with a fork-join activity with fixed annotations and 25 levels.

On the other hand, using uniformly distributed random annotations resulted in much larger running times, with 10s required on average to annotate a fork-join activity with 25 levels. Nevertheless, Figure 3(b) does not grow as quickly as would be expected from the  $O(n4^f)$  absolute worst case.

This suggests that removing redundant paths reduces the impact of the absolute worst case. However, its effectiveness depends on the relative magnitude of the minimum time limits and weights with regards to the global time limit  $L$ . The left operand of  $(b-d)L < bc - ad$ , part of (3), grows as  $L$  increases and reduces the number of comparable pairs of paths.

We performed an additional study to clarify how common the absolute worst case was and study its relationship with  $L$ . We sampled with  $L = 0.5s$  and  $L = 1.5s$  the space of all fork-join activities with 3 levels which contained a 2-level fork-join with 4



**Fig. 4.** Distribution of incomparable top-level paths over sampled 3-level fork-join activities, by global time limit

incomparable paths. Minimum time limits for the actions ranged from 0 to  $\min\{L, 1\}$ , in steps of 0.1s. Weights ranged from 0 to 10, in steps of 1 unit. Inconsistent graphs were discarded. For each activity, we measured the number of incomparable paths at the initial node (“top-level paths”): in a 3-level fork-join activity, there can be between 1 and  $2^3 = 8$  such paths.

Evaluating  $1.99 \times 10^6$  fork-join activities for  $L = 0.5s$  and  $7.16 \times 10^9$  for  $L = 1.5s$  produced the results in Figure 4. It is interesting to note that for  $L = 1.5s$ , while 31.842% of all 1-level fork-join activities were in the worst case, only 2.492% 2-level fork-join activities were in the worst case. With 3 levels, no fork-join activities were in the worst case with  $L = 0.5s$ , and only 0.047% were in the worst case with  $L = 1.5s$ . This suggests that the absolute worst case becomes harder to find with more complex graphs, explaining why average times did not grow exponentially in Figure 3(b). It also indicates that the worst case is more common when  $L$  grows in relation to the values in the annotations.

## 5 Generation of Test Cases

In previous sections, we have shown how to create the performance analysis models and how to infer the missing constraints from the existing annotations. These performance analysis models can already be useful as a way to check if a certain performance level is feasible or not, and to negotiate SLAs. In this section, we will propose another use case for the performance analysis model: test case generation.

Generating test cases directly from an abstract model such as that in Figure 1 is unfeasible, as it lacks the implementation details that are required. At the same time, it is undesirable to pollute an abstract model with these implementation details and couple it to a specific technology. To solve this situation, we propose linking the performance analysis model to implementation model, using a third generic *weaving model*. Model weaving is a well-known model management technique and is readily implemented in tools such as AMW [11] or Epsilon ModelLink [18].

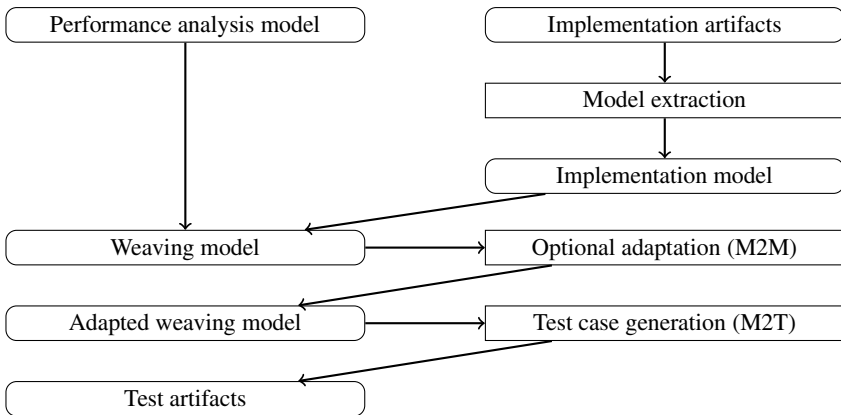


Fig. 5. Proposed approach for test case generation

The resulting approach is shown in Figure 5.

1. First, appropriate models of the implementation artifacts are extracted. Model extraction may involve parsing program code and creating an abstract syntax tree, using tools like MoDisco [8]. Alternatively, the model may already be available if the code was generated using a contract-first or model-driven approach. This is quite common for web services described using WSDL [31] documents: frameworks like Apache CXF [2] generate code from them.
2. Next, the user creates a weaving model that relates the implementation artifacts with the activities in the UML activity diagram. An activity could be linked to a JUnit test case to be reused as a performance test case, or as a web service described in a WSDL document.
3. The weaving model may not be usable as-is. In that case, an additional M2M (Model-to-Model) transformation will be required. Examples of M2T technologies include ATL [9] or ETL [19].
4. Finally, the weaving model will be used in an M2T (Model-to-Text) transformation to produce the test cases. In the case of a JUnit test case, it could be wrapped as a performance test case using a library such as JUnitPerf [10] to wrap the test case as a performance test case. For WSDL-based web services, a test plan for a performance testing tool such as Apache JMeter [14].

The main advantage to this approach is that it keeps the performance analysis model decoupled from both the methodology used to produce the implementation artifacts and the technologies used in them. Additionally, if the performance requirements change, the algorithms can be run again and the performance test cases can be regenerated with no additional work. The main challenge is managing the additional complexity introduced by the steps in Figure 5, but it can be overcome with proper tooling.

## 6 Related Work

Obtaining the desired level of performance has been a regular concern since the development of the first computer systems, as shown by the early survey in [20]. There are basically two approaches: evaluating a model of a prospective system, or measuring the performance of an implemented system. These approaches are complementary: using analytic models reduces the risk of implementing an inefficient software architecture, which is expensive to rework [25], and can find potential bottlenecks before they happen [4]. When the system is implemented, measuring its performance is more accurate, and can detect not only design issues, but also bad coding practices, unexpected workloads or platform issues. Avritzer et al. describe in [5] an interesting case study in which a simulation model was used to find the cause for a performance regression found during regular monitoring of a configuration derived from regular performance testing. Our work adapts the MARTE profile, a standard notation used for modeling non-functional requirements and creating analytic models from them, to generate the performance requirements for testing each part of the system.

Using analytic models requires highly specialised knowledge and notations. Widespread adoption of UML as a *de facto* standard notation has prompted researchers to

derive their analytic models from UML models, first with *ad hoc* annotations and later consolidating on the standard extensions to UML, such as QoS/FT [22] or SPT [21]. The survey in [29] reviews many of the approaches before MARTE replaced SPT in 2009. Since then, MARTE has been used for many purposes, such as deriving process algebra specifications [26] and extended Petri networks [32] or detecting data races [24], among others. We selected MARTE as it is based on UML, it is being actively used and offers both pre-made annotations (like SPT) and a generic framework (like QoS/FT).

Bernardi et al. have defined the Dependability and Analysis Modeling sub-profile for MARTE [7]. It has been combined with the standard GQAM and PAM sub-profiles of MARTE to evaluate the risk that a soft real-time system does not meet its time limits [6]. Our work also handles time limits, but our focus is different: we help the tester “fill in the blanks” using the available partial information. We use a model of the system to generate some of the parameters of the performance test cases.

Alhaj and Petriu generated intermediate performance models from a set of UML diagrams annotated with the MARTE profile, describing a service-oriented architecture [1]: UML activity diagrams model the workflows, UML component diagrams represent the architecture and UML sequence diagrams detail the behaviour of each action in the workflows. In our previous work, we similarly modeled workflows in a service-oriented architecture using an *ad hoc* notation based on UML activity diagrams [16]. However, our approach does not model the resources used by the system: we assume tests are performed in an environment which mimics the production environment.

There are many other recent approaches that use UML activity diagrams for generating performance test cases, without using MARTE. Avritzer et al. describe in [3] an approach for generating performance test cases considering the most common states in a system, modelled as a Markov chain. Garousi [17] uses UML sequence and activity diagrams in combination with other models to generate network stress tests for a distributed system, using evolutionary algorithms to drive the process. In general, these approaches attempt to generate test cases that cover the entire system. In comparison, our approach focuses on obtaining test cases for each part of the system, based on global requirements for the entire system.

## 7 Conclusions and Future Work

Software needs to meet its performance requirements in addition to its functional requirements. To achieve this goal, several approaches can be combined: the expected performance can be estimated using an early model, or the actual performance of the system can be measured. Currently, the research community is converging on the UML MARTE profile [23] as a standard notation to drive early performance and scheduling analysis. On the other hand, performance testing requires expectations to be defined for each part of the system. However, these are usually only available for high-level components: developers need to manually translate these to lower-level requirements for the smaller subcomponents.

In this work, we have adapted and improved the algorithms in [16] to operate on MARTE-annotated UML activity diagrams, inferring performance requirements from a global annotation and some local ones. One algorithm infers throughputs and has polynomial cost in relation to the number of nodes of the activity. The other infers time



limits and its worst case has exponential cost, as it may need to enumerate all paths from the initial node to the final nodes. However, further analysis of the average case suggests that this worst case is very rare, and becomes even harder to find as graphs are more complex. This is because the time limit inference algorithm discards redundant subpaths using a partial order relation.

After describing and evaluating the inference algorithms, we have propose an approach for generating concrete performance test cases for each action in the UML activity diagram. To keep it decoupled from the implementation technology and methodology, we propose weaving it to an implementation model and generating the actual test cases from the weaving model. The implementation model may already exist if using a contract-first or model-driven methodology. Alternatively, the model may be extracted from the actual code. We have selected some target technologies to implement our approach for regular JUnit functional tests and WSDL-based web services. We plan to implement the proposed approach for some of these technologies in the near future.

As for the algorithms, we intend to handle nested activities in a later version, so the user can describe the system as a hierarchy of components and infer time limits and throughputs in a top-down approach. Handling actions which are repeated in several places would be interesting, but the cost of the algorithms might increase.

**Acknowledgements.** This work was partly funded by the research scholarship PU-EPIF-FPI-C 2010-065 of the University of Cádiz.

## References

1. Alhaj, M., Petriu, D.C.: Approach for generating performance models from UML models of SOA systems. In: Proc. of the 2010 Conference of the Center for Advanced Studies on Collaborative Research, CASCON 2010, pp. 268–282. ACM, New York (2010)
2. Apache Software Foundation: Apache CXF (November 2011), <https://cxf.apache.org/>
3. Avritzer, A., Vieira, M.E.R.: Generating performance tests from UML specifications using Markov chains. USPTO Patent Application 11/386,971 (November 2006)
4. Avritzer, A., Weyuker, E.J.: The automatic generation of load test suites and the assessment of the resulting software. *IEEE Transactions on Software Engineering* 21(9), 705–716 (1995)
5. Avritzer, A., Weyuker, E.J.: The role of modeling in the performance testing of e-commerce applications. *IEEE Transactions on Software Engineering* 30(12), 1072–1083 (2004)
6. Bernardi, S., Campos, J., Merseguer, J.: Timing-Failure risk assessment of UML design using time petri net bound techniques. *IEEE Transactions on Industrial Informatics* PP(99), 1 (2010)
7. Bernardi, S., Merseguer, J., Petriu, D.C.: A dependability profile within MARTE. *Software & Systems Modeling* 10(3), 313–336 (2009)
8. Bruneliere, H., Cabot, J., Jouault, F., Madiot, F.: MoDisco: a generic and extensible framework for model driven reverse engineering. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, pp. 173–174 (September 2010)
9. Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P.: The AMMA platform support for modeling in the large and modeling in the small. Research Report 04.09, LINA, University of Nantes, Nantes, France (February 2005)
10. Clark, M.: JUnitPerf (October 2009), <http://clarkware.com/software/JUnitPerf.html>

11. Del Fabro, M.D., Bézivin, J., Valduriez, P.: Weaving models with the eclipse AMW plugin. In: Proceedings of the 2006 Eclipse Modeling Symposium, Eclipse Summit Europe, Esslingen, Germany (October 2006)
12. Eclipse Foundation: Homepage of the Eclipse MDT Papyrus project (2011), <http://www.eclipse.org/modeling/mdt/papyrus/>
13. Erl, T.: SOA: Principles of Service Design. Prentice Hall, Indiana (2008)
14. Apache Software Foundation: Apache JMeter (November 2011), <http://jakarta.apache.org/jmeter/>
15. García-Domínguez, A.: Homepage of the SODM+T project (January 2011), <https://neptuno.uca.es/redmine/projects/sodmt>
16. García-Domínguez, A., Medina-Bulo, I., Marcos-Bárcena, M.: Inference of performance constraints in Web Service composition models. In: CEUR Workshop Proc. of the 2nd Int. Workshop on Model-Driven Service Engineering, vol. 608, pp. 55–66 (June 2010)
17. Garousi, V.: Traffic-aware Stress Testing of Distributed Real-Time Systems based on UML Models using Genetic Algorithms. PhD thesis, Carleton University, Ottawa, Canada (August 2006)
18. Kolovos, D.S.: Epsilon ModeLink (2011), <http://eclipse.org/gmt/epsilon/doc/modelink/>
19. Kolovos, D., Paige, R., Rose, L., Polack, F.: The Epsilon Book (2010), <http://www.eclipse.org/gmt/epsilon>
20. Lucas, H.: Performance evaluation and monitoring. ACM Computing Surveys 3(3), 79–91 (1971)
21. OMG: UML Profile for Schedulability, Performance, and Time (SPTP) 1.1 (January 2005), <http://www.omg.org/spec/SPTP/1.1/>
22. OMG: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (QFTP) 1.1 (April 2008), <http://www.omg.org/spec/QFTP/1.1/>
23. OMG: UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) 1.0 (November 2009), <http://www.omg.org/spec/MARTE/1.0/>
24. Shousha, M., Briand, L.C., Labiche, Y.: A UML/MARTE Model Analysis Method for Detection of Data Races in Concurrent Systems. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 47–61. Springer, Heidelberg (2009)
25. Smith, C.U., Williams, L.G.: Software performance engineering. In: Lavagno, L., Martin, G., Selic, B. (eds.) UML for Real: Design of Embedded Real-Time Systems, pp. 343–366. Kluwer, The Netherlands (2003)
26. Tribastone, M., Gilmore, S.: Automatic extraction of PEPA performance models from UML activity diagrams annotated with the MARTE profile. In: Proc. of the 7th Int. Workshop on Software and Performance, Princeton, NJ, USA, pp. 67–78. ACM (2008)
27. Weyuker, E.J., Vokolos, F.I.: Experience with performance testing of software systems: Issues, an approach, and case study. IEEE Transactions on Software Engineering 26, 1147–1156 (2000)
28. Woodside, M., Franks, G., Petriu, D.: The future of software performance engineering. In: Proc. of Future of Software Engineering 2007, pp. 171–187 (2007)
29. Woodside, M.: From Annotated Software Designs (UML SPT/MARTE) to Model Formalisms. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 429–467. Springer, Heidelberg (2007)
30. Woodside, M., Petriu, D.C., Petriu, D.B., Shen, H., Israr, T., Merseguer, J.: Performance by unified model analysis (PUMA). In: Proc. of the 5th Int. Workshop on Software and Performance, Palma, Illes Balears, Spain, pp. 1–12. ACM (2005)
31. World Wide Web Consortium: WSDL 2.0 part 1: Core Language (June 2007), <http://www.w3.org/TR/wsdl20>
32. Yang, N., Yu, H., Sun, H., Qian, Z.: Modeling UML sequence diagrams using extended Petri nets. In: Proc. of the 2010 Int. Conference on Information Science and Applications, pp. 1–8 (2010)

# Typing Legacy COBOL Code

Alvise Spanò, Michele Bugliesi, and Agostino Cortesi

Dipartimento di Scienze Ambientali, Informatica e Statistica,  
Università Ca' Foscari Venezia, Venice, Italy  
{spano,michele,cortesi}@dsi.unive.it

**Abstract.** Maintenance of COBOL applications that still exist and work today is an open issue for many companies that have not yet undertaken the crucial decision of migrating to a modern development platform. And even those who did, most likely had to face a major challenge: understanding what those million lines of code do and what business processes they originally implemented. Automating the task of reconstructing the business logic of programs somehow is hard nonetheless: COBOL code is difficult to even get parsed - let alone applying Program Understanding techniques based on sophisticated deductions over information inferred from the code. Here we propose a system based on the translation of COBOL into a simpler and cleaner language and a type system capable of inferring types of program variables. Being COBOL a language in which variable reuse has been a widespread practice for decades, strong typing rules in the usual sense wouldn't simply fit, therefore our system provides special *flow types* for tracking multiple types a variable may assume and follows the program control-flow until no more type-changes occur for variables in the typing context. Alongside it detects also a number of error-prone situations, type mismatches and data corruptions due to misalignment or misfit in variable reuse when types have incompatible in-memory representations, while still guaranteeing the typing process does not halt.

**Keywords:** Type system, Type inference, Variable reuse, Type flow, Flow type, Island grammar, IBM z/OS, COBOL, COBOL85.

## 1 Introduction

In this paper we propose a revision of the system originally presented in [13], which is a light-weight system for *statically typing* COBOL with rich yet simple types that pursue a number goals:

1. model the COBOL picture system without losing storage format information such as computational fields or the size of a numeric; this let us reconstruct the exact in-memory representation of datatypes and perform precise comparisons among the many formats COBOL supports;
2. deal with what in [5] is called *pollution* in such a way that no complex relational property system among types is needed, by tracking type alterations that variables are subject to in the following scenarios:

- (a) when data is reused for different purposes in a program: many COBOL programmers have been used to this practice in order to save memory and the result is often poorly maintainable and error-prone code;
  - (b) when the language performs an implicit datatype cast, readapting value representation to fit a target variable having a different format than the source - and this can happen in COBOL either at compile-time or at run-time.
3. deal with branches in the program flow that are not statically decidable (i.e. conditional statements) by embedding into the type itself multiple types a variable may possibly assume during the execution. In other words, we don't try to guess how a condition is evaluated - we rather keep track of the types a variable might assume in multiple control-flow branches.

**Type System.** Usually types are given both to expressions and computations that do not lie in memory and to data bound to variable names or memory cells. We introduce a form of type for both cases plus a higher-order special type for variables having multiple types.

1. *storage types* are the *single* types having an in-memory representation a variable may assume;
2. *temporary types* are the *single* types a computation can assume before it gets stored in memory or bound to a variable;
3. *choice types* are the *multiple* storage types a variable can assume resulting from conditional branches in the program flow;
4. *flow types* are simply a pair encapsulating a choice type and the original initialization storage type of a variable.

**Jumps and Loops.** GOTO and PERFORM instructions are constantly found in legacy COBOL code as the main constructs for altering the control-flow of programs, hence our system cannot behave like an ordinary type-checker or type-inference algorithm: it looks more like a *type analyzer* capable of following jumps and branches in the code, detect cycles and avoid loops by checking for a convergence in the status of the typing function - in a way that somehow resembles that of typical Abstract Interpretation techniques [6].

**Implementation.** A prototype of the system described in this article has been developed in F# for the .NET 4.0 platform and includes a full-featured COBOL parser and translator to our *Intermediate Language* as well. A Lex & Yacc tweak has been designed for reproducing the behavior of Island Grammars [11] while keeping the benefits of an efficient LALR parser.

Typing-wise, it is currently able to parse large COBOL source programs (up to many hundreds of thousands of lines) and to type them generating as output the flow-types

---

<sup>1</sup> Finding a convergence in the status of the typing function is almost trivial compared to interpreting computations on abstract domains. That is basically thanks to the static nature of types: even including variable reuse, the overall number of reuses in any program is finite and can be statically determined in a finite number of passes.

annotated at every variable occurrence. Additionally, it annotates useful information on type usage in form of error messages, warnings and hints. Again as opposed to a compiler, errors do not make the system fail: typing carries on and is tolerant to ill-typed situations by simply switching back to the initialization type of a variable whenever ambiguous scenarios are found.

## 1.1 Overview

Our system does not manipulate COBOL code directly: as other remarkable systems do [3], we translate COBOL into an *Intermediate Language* (from now on referred to as IL) resembling modern imperative languages without altering COBOL semantics and principles. The syntax of IL is shown in section 3.1. Notably, what in COBOL speak is referred to as a program (i.e. a compilation unit), in IL becomes a procedure with its own set of static variable declarations. A COBOL application consisting of many programs translates into a single large IL program and the COBOL entry-point as its bottom unnamed block.

Before performing the type checking, the system labels all variable occurrences in the program with an unique identifier - for example a fresh integer. The type checker eventually proceeds statement by statement and recursively descends into expressions, basically performing two operations:

1. updating the possible type changes a variable is subject to and keeping track of multiple types variables could concurrently assume;
2. annotating each variable occurrence with its current flow-type at that point in the program.

Assignments and call-by-reference argument passing are the two scenarios where variables could be subject to an implicit cast, hence the type of a variable could change. And conditional constructs are responsible for merging multiple types resulting from the typing of the two sub-blocks of an *if-then-else* statement into one *choice type*.

Look at the following sample code directly translated from COBOL into IL:

```
{
  x := x + 1;
  if x > 0 then x := "foo";
  x := x + 23;
}
where x : num[2] := 11
```

Our system can reconstruct the types of the program and annotate each occurrence of variable *x* with its flow-type in that point of the code. Typing follows all branches in the control-flow: after the *if* block the system has to remember that *x* *might* have become a string. Also, where an ambiguous or ill-typed operation takes place, the system reports that and recovers to a default decision.

```
{
  (x : num[2]) := (x : num[2]) + 1; [WARNING] possible truncation in assignment: num[3] :> num[2]
  if (x : num[2]) > 0 then
    (x : alpha[2]) := "foo"; [ERROR] truncation in assignment: alpha[3] :> num[2]

  (x : num[2]) := (x : num[2]|alpha[2]) + 23;
  [HINT] type of 'x' is ambiguous in expression: assuming initialization type num[2]
  [WARNING] possible truncation in assignment: num[3] :> num[2]
}
where x : num[2] := 11
```

In the statement at line 1, in which `x` gets incremented by 1, its type is annotated both at its usage as a right-value and as the target at the left hand of an assignment. In the right-hand its type is the initialization type `num[2]`, which obviously happens to be its current type at the beginning of the program; in the left-hand `x` should apparently be given a wider numeric type, because the result of the sum of a `num[2]` and a literal whose type is `num[1]` actually leads to a `num[3]`<sup>2</sup>, but it gets truncated in order to fit the initialization type, exactly as COBOL run-time does. The final type happens to be equivalent to the initialization type and nothing seems changed, but internally the whole process has passed through the creation and the truncation of the temporary type `num[3]`.

Encountering the `if` statement makes the analyzer descend into the `then` block: a truncation is detected therein (being `alpha[3]` wider than the target type `num[2]`) and the truncated type `alpha[2]` is finally given to `x`, which fits the initialization type. Such information must then be merged to what had been previously inferred before the `if` statement: hence why, in the assignment under the `if` block, the type of `x` in the right hand is not a simple type. A choice type has been introduced here by the merge: it shows the multiple types `x` might have at that program point. Which leads to an ambiguity when typing the sum operation and so the system needs to recover to the initial type declaration to prevent from failing. That might seem odd, but is in fact a viable solution: in COBOL every variable strictly adheres to its picture declaration, thus falling back to the initialization type is not unsafe - it is just inaccurate, but it serves just as a last resort in unsolvable situations.

## 1.2 Comparisons and Motivation

As already mentioned, the legacy software analysis system thoroughly presented in [12]<sup>3</sup> rely on mechanisms for producing information over types that mainly serve Program Understanding techniques, Concept Analysis [10] and other high-level elaborations. In general, its scope is wider than ours and not entirely overlapping. While the basic goal may look similar, i.e. giving somehow interesting types to COBOL variables, there are several differences.

- We translate COBOL into a simpler intermediate language as [2] does, though without leaving out important language constructs whose behavior is relevant to typing real-world programs, such as jump primitives, procedure calls and conditional statements.
- Our type syntax is more complete and open to orthodox type manipulation, as it doesn't provide a flat representation of COBOL picture system<sup>4</sup>.

<sup>2</sup> Simply because a number of 2 digits plus a number of 1 digit could possibly lead to a number of 3 digits, as  $99 + 9 = 109$ . See the type rules for expressions in table 6 for details on how arithmetic operations formally affect numeric type formats.

<sup>3</sup> That is a Ph.D. thesis collecting previous works on the same subject and anticipating some that yet had to come. In general, that system has been proposed several times in more articles with some additions - we might therefore refer to either [2], [5], [4], [10] or [12].

<sup>4</sup> Syntax of types in [2], weirdly, include variable names and picture format strings into type terms, leaving unclear how the type environment and type comparison functions handle them.

- The type inference rules given in [5]<sup>5</sup> are sometimes a tad trivial. In our type system the type reconstruction is more detailed, e.g. our type rules for arithmetic operators in table 6 recalculate the resulting type format length in order to detect a number of size errors at typing time.
- We don't infer a type equivalence when two or more types are expected to be the same. Our system rather falls back to a variable initialization type in case a type mismatch or ambiguity is detected. This trade off does not necessarily imply a loss of information and reflects COBOL run-time semantics better.
- The system in [10] represents the inferred set of type relations via a Relational Algebra and resolves it by applying an algorithm written in Grok [7]: the resolution is actually a *simplification* process performing iterative unification. This approach doesn't seem to take into account control-flow jumps. Our system performs a code analysis at typing time by following branches and jumps and thus detects a wider range of possible type anomalies and variable reuses.
- According to [5], *pollution* occurs whenever a type-equivalence involves types that are not equivalent or subtypes: we do not handle this as a special case, but it automatically comes from *non-singleton choices* within flow-types, which are natively supported by our type-system and do not require any further processing.
- Our *subtype* relation deals with the in-memory representation of a wider range of type formats and qualifiers that are very common in COBOL programs, such as all COMP fields (translated into native integer, floating point and binary-coded-decimal types), signed/unsigned numeric formats and mixed alphabetic/alphanumeric strings.
- In [5] there is no mention on how COBOL references<sup>6</sup> are handled, nor on how COBOL run-time data conversions affect type rules manipulating different picture formats and computational fields (e.g. a COMPUTE instruction using mixed numeric variables and literals). A major feature of our system is to statically reproduce COBOL run-time and compile-time behaviors, keeping track of numeric formats and sizes and introducing *temporary types* for R-values<sup>7</sup> which are eventually *promoted* to storage types when a type coercion to a L-value occurs (see definitions [1] and [6]).

## 2 Type System

### 2.1 Storage Types and Flow-types

COBOL picture declarations in the Working Storage section of the Data Division define data instances along with their own storage format: they're not just type declarations.

<sup>5</sup> The word *inference*, with a clear reference to ML and type theory in the functional language world, is a bit improper for the context: we'd rather prefer *reconstruction* or *analysis*, as there is no use of type variables and unification for resolving a set of constraints over type equations as in actual type inference systems [1].

<sup>6</sup> According to COBOL syntax specification in [8], accessible memory cells are called *references*. We renamed them as *left values* in our intermediate language for the sake of symmetry with imperative languages such as C that define them as a sub-class of expressions which can appear at the left side on an assignment and refer to a real memory location [9].

<sup>7</sup> Symmetrically, emphright values are expressions that can stand on the right side of an assignments, hence evaluate to a temporary value [14] that does not lie in memory.

Our system must of course reproduce this design, but mapping COBOL picture format strings into types. For example, consider the following picture declaration:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
  01 A PIC 9(3) COMP-3 OCCURS 10.
  01 N PIC COMP 9(8).
  01 R1.
    02 R1-S PIC A(2).
    02 R1-B PIC X(3)9(2)A(3).
  01 R2 OCCURS 7.
    02 X PIC S99V9 COMP-2.
```

We translate it into more readable and compact type bindings that are quite self-explanatory:

$$\begin{aligned} A &: num_{bcd}[3] \text{ array}[10] \\ N &: num_{int_{32}}[8] \\ R1 &: \{S : alpha[2]; B : alphanum[8]\} \\ R2 &: \{X : num_{float_{64}}[S2.1]\} \text{ array}[7] \end{aligned}$$

Picture format strings are mapped into either numeric, pure alphabetic or alphanumeric types according to their structure; arrays and records are also first-class citizens of the type language in our system and can therefore be nested at will, yielding to types that resemble those of modern functional languages. Moreover, numeric types carry along detailed information on their in-memory representation at machine level, sign and length of both integral and fractional parts; while arrays and alphabetic/alphanumeric strings simply carry their length. The full syntax of the type-system follows:

$$\begin{aligned} \tau &:= && \text{storage types} \\ & \quad num_q[\rho] && \text{numeric} \\ & \quad | \quad alpha[n] && \text{alphabetic string} \\ & \quad | \quad alphanum[n] && \text{alphanumeric string} \\ & \quad | \quad \tau \text{ array}[n] && \text{array} \\ & \quad | \quad \{x_1 : \tau_1 \dots x_n : \tau_n\} && \text{record} \end{aligned}$$

$$\begin{aligned} \sigma &:= && \text{temporary types} \\ & \quad \tau \\ & \quad | \quad bool && \text{boolean} \\ & \quad | \quad \overline{num}[\rho] && \text{abstract numeric} \end{aligned}$$

$$\begin{aligned} q &:= && \text{numeric storage qualifier} \\ & \quad ascii && \text{display or ASCII} \\ & \quad | \quad bcd && \text{binary-packed decimal} \\ & \quad | \quad int_{16|32|64} && \text{native integer} \\ & \quad | \quad float_{32|64} && \text{native float} \end{aligned}$$

$$\rho := [S]n.d \quad \text{numeric format}$$

$$\varphi := \{\tau_1 \dots \tau_k\} \quad \text{flow-item or choice}$$

$$\Phi := \langle \varphi; \tau \rangle \quad \text{flow-type}$$

where  $k \geq 1$ ,  $n \in \mathbb{N}^*$ ,  $d \in \mathbb{N}$



There are two distinct classes of types:

- $\tau$  is the type of storage variables and L-values in general, i.e. the type of data that stands in memory and has some representation<sup>8</sup>;
- $\sigma$ , where  $\sigma \supset \tau$ , is the type given to expression terms only and is never produced by picture translation, serving just as a temporary light-weight type whose in-memory representation is yet to be known in that context.

As typing rules will show, such temporary types are eventually promoted to ordinary  $\tau$  types as soon as the storage type of an actual variable becomes known, for example when an expression that's given a temporary is then assigned to a L-value or passed as a call-by-ref argument in an procedure call. Finally, a *flow-type* is a simply a pair of possibly multiple storage types (those a variable may concurrently have following statically undecidable conditional branches in the program flow, as stated in section [11](#)) and an additional single storage type, which is the type initially declared for the variable in the global environment. We'll be often referring to the first component of a flow-type as *flow-item* or *choice*.

## 2.2 Environments

Type rules operate over a number of environments mapping different entities.

**Type Environment**  $\Gamma$  maps variable identifiers to flow-types: this environment is initially populated with global type declarations and its bindings are then updated when the current flow-type changes during typing. It contains bindings of form  $x : \Phi$ .

**Topological Environment**  $\Theta$  collects all annotations produced by the type analyzer by mapping labeled variable occurrences  $x^k$  to its flow-type at that program point. It represents also the status of the typing function in the detection of loop termination. It contains bindings of form  $x^k : \Phi$ .

**Procedure Environment**  $\Pi$  maps procedure names to signatures (see definition [8](#)). It contains bindings of form  $p \mapsto \langle y_1 : \tau_1^p \dots y_n : \tau_n^p ; \Gamma_p \rangle$ .

**Block Environment**  $\Sigma$  maps label identifiers to blocks of statements. It contains bindings of form  $l \mapsto \{ \underline{st}_1 \dots \underline{st}_n \}$ .

## 2.3 Coercion of L-Values

Take the following example:

```
{
a[0].1 := "boo";
}
where a : { l : num[2]; m : alpha[10] } array[5]
```

And its annotated form resulting from the type analysis:

<sup>8</sup> ASCII is the default qualifier for numeric types: whenever unspecified this one holds, as in `num[3]` for example.

```

{
(a : { { 1 : alpha[2]; m : alpha[10] } array[5] )[0].1
  := "boo";
}
where a : { 1 : num[2]; m : alpha[10] } array[5]

```

The literal "boo" having type  $\text{alpha}[3]$  is assigned to field 1 of a record within a cell of an array. The flow-type of variable  $a$  needs to be updated here somehow with the type of the right-hand of the assignment - and of course it's not to a that such type must naively be given, but to the record field 1 nested within. Nonetheless the environment binds variable identifiers to flow-types, thus there is no way to update the type of a record label (as 1 in our case) or of an array cell alone. Therefore the whole type of a variable must be updated keeping the original structure layout and replacing the appropriate bit nested within it. Hence, the whole type of  $a$  in the example becomes  $\{ 1 : \text{alpha}[2]; m : \text{alpha}[10] \}$  array[5].

This shows also that the expected type  $\text{alpha}[3]$  of the literal "boo" has been adapted to *fit* into the initialization type  $\text{num}[2]$ : coercion in assignments needs therefore both to replace a piece of a type and to resize it accordingly, keeping the original storage class ( $\text{num}$  in our example) and recalculating the format in such a way that the overall size of the new resulting type fits the initialization one.

For this reasons, judgements for L-value terms are slightly different:  $\Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} lv : \tau \setminus \theta^{x^k} \triangleright \Theta_1$  means that the L-value  $lv$  has a storage type  $\tau$  coercible by the substitution  $\theta^{x^k}$ , where  $x$  is the root variable of  $lv$  (formally  $x = \mathfrak{R}(lv)$  as of definition 7) and  $x^k$  is its labeled occurrence.  $\theta$  is a function from storage types to storage types that can be passed by typing rules that need to update the type of the root variable of an L-value to the coerce function  $\mathcal{C}$  (see definition 6), which performs the proper fit operation among other things.

## 2.4 Loops and Convergence

As informally stated in section 1, the type analyzer follows `goto` and `perform` statements unless already visited and a convergence in the status of the typing function is detected. In subsection 2.2 we said that this status actually consists of the topological environment  $\Theta$ . The typing function at step  $i$  of the analysis can be defined as a function taking the statement fetched at that step and the topological environment:

$$\mathcal{T}_i(st_{B,p}, \Theta_i) = \Theta_{i+1}$$

where  $st_{B,p}$  is the statement located within block  $B$  at position  $p$ .

Each time the typing function encounters a jump statement, it performs a number of operations. Say a jump statement  $st_{A,q} \equiv \text{goto } l$  is encountered by  $\mathcal{T}$  at step  $i$  while typing block  $A = \underline{\{st_{A,1} \dots st_{A,n}\}}$  (with  $q \in [1, n]$ ):

1. it saves the topological environment  $\Theta_i$  built up so far, binding it to the current program location;
2. it looks up the destination block of statements from the block environment, hence  $B = \underline{\{st_{B,1} \dots st_{B,m}\}} = \Sigma(l)$ ;
3. it continues the analysis from there, i.e. from statement  $st_{B,1}$ .

Let's consider that later at step  $j$  (obviously  $j > i$ )  $\mathcal{T}$  reaches the jump statement  $st_{A,q}$  again: then the new current topological environment  $\Theta_j$  is compared against  $\Theta_i$ , which had formerly been saved at that program location. If  $\Theta_j \sqsubseteq \Theta_i$  (see definition [11](#)) then it means that no further type information has been collected during the second pass and we can therefore assume that the analysis can safely skip the jump statement  $st_{A,q}$  and continue from  $st_{A,q+1}$ . Else, the new topological environment  $\Theta_j$  is saved (replacing the old  $\Theta_i$  previously stored) and the analysis continues from the jump statement destination  $st_{B,1}$  again.

We observed that the system detects a convergence averagely in 1 and anyway in up to 3 reiterations of the same piece of code. The reason is twofold:

- the topological environment cannot by definition be subject to binding removal, hence  $\forall x^k \in \Theta_i. x^k \in \Theta_{i+1}$  at any given step  $i$ ;
- flow-types bound to variable occurrences in the topological environment can only grow - they can never diminish in width. Given we're dealing with types and not values, the stability is certain: storage types of variables do not change from pass to pass for obvious reasons and the only thing that could change and modify the status  $\Theta$  of the typing function  $\mathcal{T}$  is the flow-item  $\phi$  part of flow-types bound to variable occurrences.  $\phi$  is defined as a set of storage types  $\tau$  in table [2.1](#) and it is subject to a single operation: the merge function as of definition [9](#), which basically consists in a set-union between flow-items. Duplicate types can therefore never occur and no element could be removed.

### 3 Formal Specification

In this section we give the full specification of the IL language and the type-system described in section [2](#).

#### 3.1 IL

Grammar rules for IL syntax are given below together with lexical rules for identifiers, literals and operators. Terminal symbols are underlined, non-terminal symbols are in italics and EBNF meta-operators are in plain form.

$$P := \underline{\text{proc}} \ p \underline{(y : \tau)^*} \ B \ \underline{\text{in}} \ P \ \text{procedure}$$

$$\quad | \ B \quad \text{main}$$

$$B := st \ \underline{\text{where}} \ (x : \tau \ [::= \ \underline{\text{lit}}])^* \ \text{body with environment bindings}$$

$$st := \underline{\text{lv}} \ ::= \ e \quad \text{assignment}$$

$$\quad | \ \underline{\text{if}} \ e \ \underline{\text{then}} \ st \ [\underline{\text{else}} \ st] \ \text{if-then/if-then-else}$$

$$\quad | \ \underline{p} \underline{(a)^*} \quad \text{call}$$

$$\quad | \ \underline{\text{goto}} \ l \quad \text{goto}$$

$$\quad | \ \underline{\text{perform}} \ l \ [l] \quad \text{perform/perform-thru}$$

$$\quad | \ \underline{\text{return}} \quad \text{return}$$

$$\quad | \ [\underline{l}] \ \{ (st) + \} \quad \text{anonymous/named-block}$$

$lv := x$	variable
$lv[e]$	array subscript
$lv.z$	record field select
$a := \underline{val} e$	call-by-value
$\underline{ref} lv$	call-by-reference
$e := e (op_a   op_l   op_r) e$	binary operator application
$(-   not) e$	unary operator application
$lit$	literal
$lv$	l-value
$lit := [\_ ]n[n]$	numeric literal
$'' [\_ ] * ''$	string literal
$true   false$	boolean literal
$x, y, z, l, p := [\underline{a} - \underline{z} \underline{A} - \underline{Z}] [\underline{a} - \underline{z} \underline{A} - \underline{Z} \underline{=} \underline{=} \underline{0} - \underline{9}] *$	identifiers
$op_a := \pm   \_   *   /$	binary arithmetic operators
$op_l := \underline{and}   \underline{or}$	binary logic operators
$op_r := \leq   \equiv   !\equiv   \leq\equiv   \geq   \geq\equiv$	binary relational operators
$n := [\underline{0} - \underline{9}] +$	natural number

### 3.2 Definitions

A number of formal definitions is needed before presenting type rules.

**Definition 1 (Promote).** *The promotion  $\llbracket \sigma \rrbracket^\tau$  of a temporary type  $\sigma$  to a storage type  $\tau$  produces a storage type that transform  $\sigma$  into a storable type inheriting the characteristics of  $\tau$ . The promotion function is defined as follows (top-down closest-match rule on the left hand holds):*

$$\begin{aligned} \llbracket \underline{num}[\rho_2] \rrbracket^{num_q[\rho_1]} &= num_q[\rho_2] & \llbracket \underline{num}[\rho] \rrbracket^\tau &= num_{ascii}[\rho] \\ \llbracket \underline{bool} \rrbracket^\tau &= \llbracket \underline{num}[1.0] \rrbracket^\tau & \llbracket \tau_2 \rrbracket^{\tau_1} &= \tau_2 \end{aligned}$$

**Definition 2 (Representation).** *We define a function  $rep : \tau \rightarrow \mathbb{N}$  for calculating the in-memory byte size of a storage type:*

$$\begin{aligned} rep(num_{ascii}[n.d]) &= n + d & rep(alpha[n]) &= n \\ rep(num_{bcd}[n.d]) &= \lceil \frac{n+d+1}{2} \rceil & rep(alphanum[n]) &= n \\ rep(num_{im_b}[\rho]) &= b/8 & rep(\tau array[n]) &= rep(\tau) * n \\ rep(num_{float_b}[\rho]) &= b/8 & rep(\{x_1 : \tau_1 .. x_n : \tau_n\}) &= \sum_{i=1}^n rep(\tau_i) \end{aligned}$$

**Definition 3 (Subtype).** *We define a total-order between storage types such that the relation  $\tau_1 \preceq \tau_2$  holds when  $rep(\tau_1) \leq rep(\tau_2)$ .*

**Definition 4 (Var-Bound Substitution).** A substitution  $\theta^{x^k}$  is a function from storage types to storage types that carries along a labeled identifier  $x^k$  which stands for the variable occurrence whose type the substitution has been built from and is supposed to replace<sup>9</sup>.

**Definition 5 (Fit).** The fit  $[\tau_1]_{\tau_2}$  of a storage type  $\tau_1$  to a storage type  $\tau_2$  produces a storage type whose storage class is equivalent to that of  $\tau_1$  and whose size fits into that of  $\tau_2$ . The fit function is defined as follows:

$$\begin{array}{l|l} [num_q[\rho]]_{\tau} = num_q[\rho'] & \text{rep}(num_q[\rho']) = \text{rep}(\tau) \\ [alpha[n]]_{\tau} = alpha[n'] & \text{rep}(alpha[n']) = \text{rep}(\tau) \\ [alphanum[n]]_{\tau} = alphanum[n'] & \text{rep}(alphanum[n']) = \text{rep}(\tau) \\ [\tau_a \text{ array}[n]]_{\tau} = \tau'_a \text{ array}[n'] & \text{rep}(\tau'_a \text{ array}[n']) = \text{rep}(\tau) \\ [\{l_1 : \tau_1..l_n : \tau_n\}]_{\tau} = \{l_1 : \tau'_1..l_n : \tau'_n\} & \text{rep}(\{l_1 : \tau'_1..l_n : \tau'_n\}) = \text{rep}(\tau) \end{array}$$

**Definition 6 (Coerce).** The coerce function  $\mathcal{C}$  updates the given type and topological environments by applying a given substitution function  $\theta^{x^k}$  to the types a given flow-item  $\phi$  consists of; it produces a new pair of form  $\langle \Gamma; \Theta \rangle$  consisting of the type and topological environments endowed with updated bindings for the variable  $x$  and the occurrence label  $\kappa$  annotated on the substitution function  $\theta^{x^k}$  itself:

$$\mathcal{C}(\phi, \theta^{x^k}, \Gamma, \Theta) = \langle \Gamma, x : \Phi'; \Theta, \kappa : \Phi' \rangle \text{ with } \begin{array}{l} \langle \phi; \tau_x \rangle = \Gamma(x) \\ \Phi' = \langle \{\forall \tau_i \in \phi. [\theta^{x^k}(\tau_i)]_{\tau_x}\}; \tau_x \rangle \end{array}$$

**Definition 7 (Root Variable).** Given an L-value  $lv$ , its root variable is the identifier  $x$  evaluated by the recursive function defined as:

$$\mathfrak{R}(x) = x \quad \mathfrak{R}(lv[e]) = \mathfrak{R}(lv) \quad \mathfrak{R}(lv.l) = \mathfrak{R}(lv)$$

**Definition 8 (Signature).** A signature is a pair  $\langle Y_p; \Gamma_p \rangle$  where  $p$  is a procedure name,  $Y_p$  are its formal parameters  $y_1 : \tau_1^p..y_n : \tau_n^p$  and  $\Gamma_p$  is the output type environment returned by typing the body of  $p$ .

**Definition 9 (Type Environment Merge).** The binary function  $\oplus$  merges two given type environments into one as  $\Gamma_1 \oplus \Gamma_2 = \Gamma^* \cup (\Gamma_1 \setminus \Gamma_2) \cup (\Gamma_2 \setminus \Gamma_1)$  where  $\Gamma^* = \{x : \langle \phi_1 \cup \phi_2; \tau_1 \rangle | \Gamma_1(x) = \langle \phi_1; \tau_1 \rangle \wedge \Gamma_2(x) = \langle \phi_2; \tau_2 \rangle \wedge \tau_1 = \tau_2\}$ .

**Definition 10 (Partial Ordering of Flow-Types).** We define a partial order between flow-types such that  $\Phi_1 \sqsubseteq \Phi_2$  holds when, let  $\Phi_1 = \langle \phi_1; \tau_1 \rangle$  and  $\Phi_2 = \langle \phi_2; \tau_2 \rangle$ , then  $\phi_1 \subseteq \phi_2 \wedge \tau_1 = \tau_2$ .

**Definition 11 (Partial Ordering of Topological Environments).** We define a partial order between topological environments such that  $\Theta_1 \sqsubseteq \Theta_2$  holds when  $\forall x : \Phi_1 \in \Theta_1.x \in \text{dom}(\Theta_2) \wedge \Phi_1 \sqsubseteq \Phi_2$ , where  $\Phi_2 = \Theta_2(x)$ .

<sup>9</sup> Substitution functions are recursively defined by type rules for L-Values as shown in table 4. They're meant for generically replacing a term nested within a storage type of arbitrary complexity by reproducing its original structure of recursive type terms and changing the innermost part only.

### 3.3 Type Rules

Syntax-directed type rules are divided by category. Rules for Programs are shown in table 1 for Statements in table 2 for Expressions in table 6 for Arguments in table 3 and for Literals in table 5.

**Table 1.** Type Rules for Programs and Body

$$\begin{array}{c}
 \text{MAIN} \\
 \frac{\Pi; \emptyset; \Theta_0 \vdash_B B \triangleright \Gamma; \Theta_1}{\Pi; \Theta_0 \vdash_P B \triangleright \Theta_1} \\
 \\
 \text{PROC} \\
 \frac{\Gamma_p = \emptyset, y_1 : \langle \{\tau_i^p\}; \tau_i^p \rangle..y_n : \langle \{\tau_n^p\}; \tau_n^p \rangle}{\Pi; \Gamma_p; \Theta_0 \vdash_B B \triangleright \Gamma_p; \Theta_1} \\
 \frac{\Pi, p \mapsto \langle y_1 : \tau_1^p..y_n : \tau_n^p; \Gamma_p \rangle; \Theta_1 \vdash_P P \triangleright \Theta_2}{\Pi; \Theta_0 \vdash_P \underline{\text{proc}} p(y_1 : \tau_1^p..y_n : \tau_n^p) B \underline{\text{in}} P \triangleright \Theta_2} \\
 \\
 \text{BODY} \\
 \frac{\forall i \in [1, n]. \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{lit} lit_i : \sigma_i \wedge \llbracket \sigma_i \rrbracket^v \preceq \tau_i}{\Pi; \emptyset; \Gamma_0, x_1 : \langle \{\tau_1\}; \tau_1 \rangle..x_n : \langle \{\tau_n\}; \tau_n \rangle; \Theta_0 \vdash_{st} st \triangleright \Gamma_1; \Theta_1} \\
 \Pi; \Gamma_0; \Theta_0 \vdash_B st \underline{\text{where}} x_1 : \tau_1 := lit_1..x_n : \tau_n := lit_n \triangleright \Gamma_1; \Theta_1
 \end{array}$$

Most judgements give a type to a term of the language in a context consisting of a tuple of environments and output the updated  $\Gamma$  and  $\Theta$ , except judgements for Statements and Programs that give no type and simply update the environments. As a general rule, the topological environment  $\Theta$  is always forwarded to and returned by all judgements (except literals), because flow-types must be annotated recursively on each variable occurring in any subterm of the program. While the type environment  $\Gamma$  is output only by rules that actually update it: consider it as returned back untouched when there's no mention of it among outputs.

Judgements are of a number of forms, each syntactic category having its own, though most of them are quite self-explanatory. For example,  $\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \sigma \triangleright \Theta_1$  denotes that, in the given environments, expression  $e$  is given a temporary type  $\sigma$  and the topological environment  $\Theta_1$  is output.

Judgements for Arguments probably need some extra words. Call-by-ref calls need to update the type environment of the the caller because the flow-type of argument might be modified by the invoked procedure. The procedure environment  $\Pi$  stores the type environment  $\Gamma_p$  for each procedure  $p$  of the program, thus the flow-type of a variable passed by reference to  $p$  can be updated according to the flow-type of the corresponding formal parameter bound in  $\Gamma_p$ . Such update is carried on by the coerce function  $\mathcal{C}$ , as shown by rule BYREF in table 3. The mechanism resembles that in rule ASSIGN in table 2: call-by-reference argument application indeed behaves like an assignment (call-by-value doesn't).

Rules for Arguments have form  $\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_a a : \tau_i \triangleright \Gamma_1; \Theta_1$ , meaning that, in the given environments, the actual argument  $a$  has type  $\tau_i^p$ , which is the type of the  $i$ -th formal parameter of procedure  $p$ .

**Table 2.** Type Rules for Statements

<p><b>ASSIGN</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_e e : \sigma_e \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma_0; \Theta_1 \vdash_{lv} lv : \tau_{lv} \setminus \theta^{x^k} \triangleright \Theta_2 \\ x^k = \mathfrak{R}(lv) \\ \langle \Gamma_1; \Theta_2 \rangle = \mathcal{C}(\llbracket \sigma_e \rrbracket^{\tau_{lv}, \theta^{x^k}}, \Gamma_0, \Theta_2) \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} lv := e \triangleright \Gamma_1; \Theta_2}$	<p><b>IF</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_e e : bool \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma_0; \Theta_1 \vdash_{st} st \triangleright \Gamma_1; \Theta_2 \\ \Gamma_2 = \Gamma_0 \oplus \Gamma_1 \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \underline{\text{if}} e \underline{\text{then}} st_1 \triangleright \Gamma_2; \Theta_2}$
<p><b>IF-ELSE</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_e e : bool \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma_0; \Theta_1 \vdash_{st} st_1 \triangleright \Gamma_1; \Theta_2 \\ \Pi; \Sigma; \Gamma_0; \Theta_2 \vdash_{st} st_2 \triangleright \Gamma_2; \Theta_3 \\ \Gamma_3 = \Gamma_1 \oplus \Gamma_2 \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \underline{\text{if}} e \underline{\text{then}} st_1 \underline{\text{else}} st_2 \triangleright \Gamma_3; \Theta_3}$	<p><b>PERFORM</b></p> $\frac{\begin{array}{l} \{st_1..st_n\} = \Sigma(l) \\ \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \{st_1..st_n\} \triangleright \Gamma_1; \Theta_1 \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \underline{\text{perform}} l \triangleright \Gamma_1; \Theta_1}$
<p><b>PERFORM-THRU</b></p> $\frac{\begin{array}{l} \forall i \in [a, b] \quad \{st_{i,1}..st_{i,n_i}\} = \Sigma(l_i) \\ \Pi; \Sigma; \Gamma_{i-a}; \Theta_{i-a} \vdash_{st} \{st_{i,1}..st_{i,n_i}\} \triangleright \Gamma_{i-a+1}; \Theta_{i-a+1} \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \underline{\text{perform}} l_a l_b \triangleright \Gamma_{b-a-1}; \Theta_{b-a-1}}$	<p><b>GOTO</b></p> $\frac{l \in \text{dom}(\Sigma)}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} \underline{\text{goto}} l \triangleright \Gamma_0; \Theta_0}$
<p><b>CALL</b></p> $\frac{\begin{array}{l} \langle y_1 : \tau_1^p .. y_n : \tau_n^p; \Gamma_p \rangle = \Pi(p) \\ \forall i \in [1, n], \Pi; \Sigma; \Gamma_{i-1}; \Theta_{i-1} \vdash_a a_i : \tau_i^p \triangleright \Gamma_i; \Theta_i \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} p(a_1..a_n) \triangleright \Gamma_n; \Theta_n}$	<p><b>BLOCK</b></p> $\frac{\begin{array}{l} \forall j   st_{0,j} \equiv l_j : \{st_{j,1}..st_{j,n_j}\} \\ \Sigma' = \Sigma, l_j \mapsto \{st_{j,1}..st_{j,n_j}\}.. \\ \forall i \in [1, m]   m \leq n \wedge st_i \neq \underline{\text{goto}} l \\ \Pi; \Sigma'; \Gamma_{i-1}; \Theta_{i-1} \vdash_{st} st_i \triangleright \Gamma_i; \Theta_i \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{st} l_0 : \{st_{0,1}..st_{0,m_0}\} \triangleright \Gamma_n; \Theta_n}$

**Table 3.** Type Rules for Arguments

<p><b>BYVAL</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_e e : \sigma \triangleright \Theta_1 \\ \llbracket \sigma \rrbracket^{\tau_i^p} \preceq \tau_i^p \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_a \underline{\text{val}} e : \tau_i^p \triangleright \Gamma_0; \Theta_1}$	<p><b>BYREF</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_{lv} lv : \tau' \setminus \theta^{x^k} \triangleright \Theta_1 \\ x = \mathfrak{R}(lv) \quad \tau' \preceq \tau_i^p \\ \langle y_1 : \tau_1^p .. y_n : \tau_n^p; \Gamma_p \rangle = \Pi(p) \\ \langle \phi_i^p; \tau_i^p \rangle = \Gamma_p(y_i) \\ \langle \Gamma_1; \Theta_2 \rangle = \mathcal{C}(\phi_i^p, \theta^{x^k}, \Gamma_0, \Theta_1) \end{array}}{\Pi; \Sigma; \Gamma_0; \Theta_0 \vdash_a \underline{\text{ref}} lv : \tau_i^p \triangleright \Gamma_1; \Theta_2}$
---	--

**Table 4.** Type Rules for L-Values

<p><b>VAR-INIT</b></p> $\frac{\begin{array}{l} \Gamma(x) = \Phi = \langle \{\tau_1 \tau_2 .. \tau_n\}; \tau_0 \rangle \\ \Theta_1 = \Theta_0, x^k : \Phi \quad \theta^{x^k}(\bar{\tau}) = \bar{\tau} \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} x^k : \tau_0 \setminus \theta^{x^k} \triangleright \Theta_1}$	<p><b>VAR-CURR</b></p> $\frac{\begin{array}{l} \Gamma(x) = \Phi = \langle \{\tau_1\}; \tau_0 \rangle \\ \Theta_1 = \Theta_0, x^k : \Phi \quad \theta^{x^k}(\bar{\tau}) = \bar{\tau} \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} x^k : \tau_1 \setminus \theta^{x^k} \triangleright \Theta_1}$
<p><b>SUBSCRIPT</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \overline{\text{num}}[p] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_{lv} lv : \tau \text{ array}[n] \setminus \theta_{lv}^{x^k} \triangleright \Theta_2 \\ x = \mathfrak{R}(lv) \\ \theta^{x^k}(\bar{\tau}) = \theta_{lv}^{x^k}(\bar{\tau} \text{ array}[n]) \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} lv[e] : \tau \setminus \theta^{x^k} \triangleright \Theta_2}$	<p><b>SELECT</b></p> $\frac{\begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} lv : \{z_1 : \tau_1 .. z : \tau .. z_n : \tau_n\} \setminus \theta_{lv}^{x^k} \triangleright \Theta_1 \\ x = \mathfrak{R}(lv) \\ \theta^{x^k}(\bar{\tau}) = \theta_{lv}^{x^k}(\{z_1 : \tau_1 .. z : \bar{\tau} .. z_n : \tau_n\}) \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} lv[z] : \tau \setminus \theta^{x^k} \triangleright \Theta_2}$

**Table 5.** Type Rules for Literals

$\frac{\text{NUM-U} \quad n = \text{len}(n_1) \quad d = \text{len}(n_2)}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} n_1[n_2] : \overline{\text{num}}[n.d]}$	$\frac{\text{NUM} \quad n = \text{len}(n_1) \quad d = \text{len}(n_2)}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} -n_1[n_2] : \overline{\text{num}}[Sn.d]}$
$\frac{\text{STRING-ALPHANUM} \quad \{0..9\} \cap \text{"str.."} \neq \emptyset \quad n = \text{len}(\text{str})}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} \text{"str.."} : \text{alphanum}[n]}$	$\frac{\text{STRING-ALPHA} \quad n = \text{len}(\text{"str.."})}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} \text{"str.."} : \text{alpha}[n]}$
$\frac{\text{TRUE}}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} \text{true} : \text{bool}}$	$\frac{\text{FALSE}}{\Pi; \Sigma; \Gamma; \Theta \vdash_{lit} \text{false} : \text{bool}}$

**Table 6.** Type Rules for Expressions

$\frac{\text{DEMOTE-NUM} \quad \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \text{num}_q[\rho] \triangleright \Theta_0}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \overline{\text{num}}[\rho] \triangleright \Theta_0}$	$\frac{\text{LV} \quad \Pi; \Sigma; \Gamma; \Theta_0 \vdash_{lv} lv : \tau \setminus \theta^{\text{xk}} \triangleright \Theta_1}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e lv : \tau \triangleright \Theta_1}$
$\frac{\text{LIT} \quad \Pi; \Sigma; \Gamma; \Theta \vdash_{lit} lit : \sigma}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e lit : \sigma \triangleright \Theta_0}$	$\frac{\text{NEG-S} \quad \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \overline{\text{num}}[Sn.d] \triangleright \Theta_1}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e -e : \overline{\text{num}}[Sn.d] \triangleright \Theta_1}$
$\frac{\text{NEG-U} \quad \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \overline{\text{num}}[n.d] \triangleright \Theta_1}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e -e : \overline{\text{num}}[Sn.d] \triangleright \Theta_1}$	$\frac{\text{NOT} \quad \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e : \text{bool} \triangleright \Theta_1}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e \underline{\text{not}} e : \text{bool} \triangleright \Theta_1}$
$\frac{\text{PLUS-U} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \overline{\text{num}}[n_1.d_1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \overline{\text{num}}[n_2.d_2] \triangleright \Theta_2 \\ n = \max(n_1, n_2) \quad d = \max(d_1, d_2) \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 + e_2 : \overline{\text{num}}[Sn.d] \triangleright \Theta_2}$	$\frac{\text{PLUS-MINUS-S} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \overline{\text{num}}[S_1 n_1.d_1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \overline{\text{num}}[S_2 n_2.d_2] \triangleright \Theta_2 \\ S = S_1 \vee S_2 \quad n = \max(n_1, n_2) + 1 \\ d = \max(d_1, d_2) \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 (+ -) e_2 : \overline{\text{num}}[Sn.d] \triangleright \Theta_2}$
$\frac{\text{MULT} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \overline{\text{num}}[S_1 n_1.d_1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \overline{\text{num}}[S_2 n_2.d_2] \triangleright \Theta_2 \\ S = S_1 \vee S_2 \\ n = n_1 + n_2 \quad d = d_1 + d_2 \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 * e_2 : \overline{\text{num}}[Sn.d] \triangleright \Theta_2}$	$\frac{\text{DIV} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \overline{\text{num}}[S_1 n_1.d_1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \overline{\text{num}}[S_2 n_2.d_2] \triangleright \Theta_2 \\ S = S_1 \vee S_2 \\ n = n_1 + d_2 \quad d = d_1 + n_2 \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 / e_2 : \overline{\text{num}}[Sn.d] \triangleright \Theta_2}$
$\frac{\text{BIN-REL-NUM} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \overline{\text{num}}[S_1 n_1.d_1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \overline{\text{num}}[S_2 n_2.d_2] \triangleright \Theta_2 \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 \text{op}_r e_2 : \text{bool} \triangleright \Theta_2}$	$\frac{\text{BIN-REL-ALPHANUM} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \text{alphanum}[n1] \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \text{alphanum}[n2] \triangleright \Theta_2 \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 \text{op}_r e_2 : \text{bool} \triangleright \Theta_2}$
$\frac{\text{BIN-LOGIC} \quad \begin{array}{l} \Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 : \text{bool} \triangleright \Theta_1 \\ \Pi; \Sigma; \Gamma; \Theta_1 \vdash_e e_2 : \text{bool} \triangleright \Theta_2 \end{array}}{\Pi; \Sigma; \Gamma; \Theta_0 \vdash_e e_1 \text{op}_l e_2 : \text{bool} \triangleright \Theta_2}$	

As a final notice, for the sake of simplicity we assume that all labels in the program are named in order of occurrence: if  $l_n$  and  $l_m$  are two labels and  $m > n$ , then  $l_m$  appear below  $l_n$  in the program. That makes type rules for jump statements simpler.



## References

1. Damas, L., Milner, R.: Principal type-schemes for functional programs. In: POPL, pp. 207–212 (1982)
2. van Deursen, A., Moonen, L.: Type inference for cobol systems. In: WCRE (Working Conference on Reverse Engineering), pp. 220–230 (1998)
3. van Deursen, A., Moonen, L.: Understanding cobol systems using inferred types. In: IWPC. IEEE Computer Society (1999)
4. van Deursen, A., Moonen, L.: Exploring legacy systems using types. In: WCRE (Working Conference on Reverse Engineering), pp. 32–41 (2000)
5. van Deursen, A., Moonen, L.: An empirical study into cobol type inferencing. *Sci. Comput. Program.* 40(2-3), 189–211 (2001)
6. Nielson, F., Nielson, H.R., Hankin, C.: *Principles of Static Analysis*. Springer (1999)
7. Holt, R.C.: WCRE 1998 most influential paper: Grokking software architecture. In: WCRE (Working Conference on Reverse Engineering), pp. 5–14. IEEE (2008)
8. IBM: Cobol z/OS language reference (2009),  
[http://publib.boulder.ibm.com/info-center/pdthelp/v1r1/index.jsp?topic=/com.ibm.debugtool.doc\\_7.1/eqa7rm0293.html](http://publib.boulder.ibm.com/info-center/pdthelp/v1r1/index.jsp?topic=/com.ibm.debugtool.doc_7.1/eqa7rm0293.html)
9. Kernighan, B.W., Ritchie, D.: *The C Programming Language*, 2nd edn. Prentice-Hall (1988)
10. Kuipers, T., Moonen, L.: Types and concept analysis for legacy systems. In: IWPC, pp. 221–230. IEEE Computer Society (2000)
11. Moonen, L.: Generating robust parsers using island grammars. In: WCRE (Working Conference on Reverse Engineering) (2001)
12. Moonen, L.: Exploring software systems. In: ICSM, pp. 276–280. IEEE Computer Society (2003)
13. Spanò, A., Bugliesi, M., Cortesi, A.: Type-flow analysis for legacy cobol code. In: ICISOFT (2), pp. 64–75 (2011)
14. Stroustrup, B.: *The C++ Programming Language*, 3rd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2000)

# A Repository for Integration of Software Artifacts with Dependency Resolution and Federation Support

Rodrigo García-Carmona, Félix Cuadrado, Juan C. Dueñas, and Álvaro Navas

Departamento de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación,  
Universidad Politécnica de Madrid, Madrid, Spain  
{rodrigo, fcuadrado, jcduenas, anavas}@dit.upm.es

**Abstract.** While developing new IT products, reusability of existing components is a key aspect that can considerably improve the success rate. This fact has become even more important with the rise of the open source paradigm. However, integrating different products and technologies is not always an easy task. Different communities employ different standards and tools, and most times is not clear which dependencies a particular piece of software has. This is exacerbated by the transitive nature of these dependencies, making component integration a complicated affair. To help reducing this complexity we propose a model-based repository, capable of automatically resolve the required dependencies. This repository needs to be expandable, so new constraints can be analyzed, and also have federation support, for the integration with other sources of artifacts. The solution we propose achieves these working with OSGi components and using OSGi itself.

**Keywords:** Component Distribution, Model-driven Engineering, OSGi, Open Source, Software Integration.

## 1 Introduction

In recent years software development has been undergoing a huge change, evolving from a closed software paradigm to new processes that incorporate open source software in products and services. The number and relevance of software developments based in the open source paradigm have experienced an exponential growth [1].

The reason for this lies in the particular strengths that open source can bring into the table, like its ability to reduce IT costs, deliver products faster and improve the security and reliability of systems. This situation has been also fostered by the numerous success cases in industry that have followed this model. In fact, sources like Forrester have described 2009 as “the year IT professionals realized that open source runs their business”, and predict that this trend is going to continue in the following years [2].

Most of the strengths that open source provide are the product of the open communities and the development models that arise from them. At this point it has become clear that the community efforts are leveraged by the participating elements, with everyone benefitting from the created ecosystem. Example successful communities are the Apache Software Foundation, the Eclipse Foundation, the ObjectWeb community and SourceForge.

Those communities have matured with different collaboration and architecture models. As a consequence, these communities are like isolated islands which no communication between them. Unfortunately, while they achieve a very high internal consistency, there is a severe lack of compatibility and integration among them. This hampers one of the most important factors for the success of open source; the reusability of code, since the lack of integration complicates this process. These integration challenges are also aggravated by the multiplicity of tools used by different projects. To further complicate this issue, most of these tools are not concerned in working with other solutions.

The heart of this problem lies in evaluating the interdependencies of software components. These dependencies tend to form a complex mesh that can span several projects and code bases and it is difficult and costly to navigate. One of the most severe problems of open source development is figuring which already existing components one has to use.

In addition to the technology impedance mismatch, there are additional factors which must be considered. An often overlooked factor with open source usage is the existence of several software licenses. While on first thought these elements should not interfere, they do actually restrict the potential uses, since some of these licenses are incompatible between them or with commercial ones.

It can be seen that the problem lies not only in code interoperability but also in additional aspects, such as legal license compatibility, or design according to similar hardware capabilities. In practice, all these problems tend to produce fragmentation, complicating the use of what software it is already available.

The meeting point for this integration is, in almost every community, a software repository. Since the repository act as a central hub for all development efforts, the difficulties exposed here are particularly evident in it.

In this article we present a comprehensive, model-based component repository that provides two features that ease the integration of software elements: 1) An automatic dependency resolution that can work with several types of concerns (software, hardware, etc...) and 2) A federation system that can aggregate the contents of other repositories and in turn expose their own components to the outside world.

This repository has been developed in the context of the ITEA-OSAMI European project and its objective is to act as a main hub in an Internet federation of repositories, while being publicly available. It integrates artifacts published by the members of both the ITEA-OSAMI project itself and external partners.

This article is structured as follows: The next section gives a brief explanation of the most recent developments in the topics that concern our work. Section 3 explains our proposal in detail, and section 4 performs a validation of our work using an implementation of the repository. Finally, the last section outlines the conclusions we have reached and shows how our work could be further developed in the future.

## 2 State of the Art

In this section we provide a brief state of the art of the technologies that are especially relevant for our proposal: the OSGi component model and the already existing software repository solutions.

## 2.1 The OSGi Component and Service Model

OSGi is an open specification that defines a component and service model for the Java platform. The latest version of the specification is 4.2 [3], and is maintained by the OSGi Alliance, a consortium formed by embedded and enterprise companies, such as IBM, Oracle, Red Hat or Siemens. It was originally designed for home gateways and embedded systems, but its adoption has greatly increased lately in desktop tools and enterprise application servers.

The relevance of the OSGi specification has increased mainly because it provides a modularity layer that was missing in the Java platform. This is enabled by the definition of OSGi bundles. Bundles extend java libraries (JAR files), allowing them to expose their functionality to the rest of the platform in a controlled way. Bundles use the Java manifest file to declare explicitly what does the component provide to the rest of elements (in the form of java packages) and what does it require in order to work properly (either java packages or complete bundles), providing in both cases version compatibility information. This directly addresses the ‘JAR hell’ problem of complex Java-based systems, greatly easing the deployment and configuration of new software.

Additionally, OSGi bundles collaborate through a lightweight service mechanism, with services being runtime Java objects that implement interfaces. This enables effective decoupling between collaborating components and simplifies the development of extensible systems. The OSGi framework provides an execution platform for OSGi bundles, enabling dynamic deployment and configuration of the components. These factors make OSGi bundles an ideal specification for open, composable service-based ecosystems, as it provides simple mechanisms for effective interoperability and modularization.

All in all, OSGi is the best solution in the Java world for the design and implementation of modular applications. It enables an even lower coupling and brings the SOA principles to the virtual machine. Our proposal will use OSGi for both the components the repository will manage and the actual repository itself.

## 2.2 Software Repository Standards

At this moment there are several repository technologies that are popular in open source communities. Every one of them has its own component model and capabilities. We detail each in this section, with a special focus in their support of OSGi bundles and federation capabilities.

Maven [4] is one of these solutions, a software project management and comprehension tool. Maven has become the de facto standard for managing Java projects, thanks mainly to the support and its extended use inside the Apache community. Maven uses a generic project description model for describing software projects named Project Object Model (POM). The POM file of a project defines the project’s lifecycle as well as its dependencies and configuration parameters. However, this model has been defined as generic as possible, in order to cover a wide range of software projects. Hence, it does not accurately reflect the special relationships of specific types of software components, such as OSGi bundles. For example, dependencies onto a particular software package can be defined, but not onto a complete bundle.

POM cannot describe these kinds of dependencies, losing information in going from manifest to POM. Despite this disadvantage, Maven repositories provide other interesting capabilities, such as being able to store all the information concerning a project (source code, documentation, etc) or the hierarchical federation with other Maven repositories, augmenting the Maven basic dependency resolution mechanism. However, this mechanism does not work with repositories implemented using other technologies.

Another model used to describe bundles is the OBR (OSGi Bundle Repository) project. This model was presented as the draft OSGi RFC 112 [5]. The RFC defines both an XML schema for bundle description and the Java API for browsing OBR repositories. An OBR repository is very simple in its structure, providing just an XML file describing the server contents. This eases the creation of OBR repositories as only the bundles and how to download them need to be described, leaving plenty of freedom to design the architecture supporting those operations. This simplicity has the drawback that the clients are forced to carry out most of the operations, a problem aggravated by the fact that there is no standard definition of an OBR client. The draft status of the OBR presents additional disadvantages, such as the lack of mechanisms for managing repository contents (e.g. upload new bundle, update, or delete). and that the federation mechanism between OBR repositories is not well-defined.

In the 3.0 version of Eclipse, the Eclipse architecture was changed to use OSGi as the project core. This change pushed the Eclipse community to develop their own bundle repository, named P2 [6]. The P2 repository is widely used, since version 3.4 of the Eclipse Platform uses it as the management mechanism for its components (OSGi bundles). The P2 specification defines two repository types: metadata and artifact. The metadata repository stores Installable Units, which are the P2 representation of an artifact. This means that almost anything can be described as an Installable Unit (configuration files, bundles, executables, etc). The metadata repository also provides the P2 federation mechanism. Complementing it is the artifact repository, which stores the binary and description files associated to the Installable Units. There is also a third component, the Director, which is part of the repository client. The Director is in charge of resolving dependencies and installing and uninstalling the artifacts. However, this solution has an important drawback: Its component model is concerned only with software direct dependencies, being oblivious to other constraints that could affect artifacts.

Also, the increasing success of the OSGi platform has stimulated the creation of proprietary bundle repositories especially dedicated to store this type of software components. The Spring Bundle Repository [7] is the most notorious example of this trend. This repository stores a collection of bundles and library description files ready for production use. A library description file is a document describing a set of bundles that are frequently used together. The access to the repository is made through Maven, Ivy or a web interface. The web interface shows information related to the dependencies and exported resources of a bundle, offering links to download them. However, the proprietary nature of this solution greatly limits its applicability and usefulness.

It can be seen that there are a lot of existing solutions for a bundle repository. But with the exception of the Spring repository (which supports Maven), there are no federation mechanisms between repositories of different types. On top of that, different development communities have chosen different repository solutions. This fact makes implementing a dependency resolution mechanism a difficult task.

Despite a previous attempt at creating standard-complying repositories [8], this work has not been followed up since its publication.

However, in the digital contents world there are many studies [9] and proposals [10-12] on this topic. But none of them have been applied to a software artifact repository. There are huge differences in nature and needs between software components and multimedia contents, and the solution that works with one cannot be used with the other without severe modifications.

### 3 Proposed Solution

In this section we detail our proposed solution. For this aim we have divided this chapter in several subsections.

As we have already said in the introduction, our aim was to provide an artifact repository that helps to improve software integration. To achieve this, our solution provides two main features: A faceted dependency resolution, and a repository federation engine. One subsection is devoted to each of them.

Also, to fully grasp how we propose to fulfill both, first we introduce two basic topics needed for the proper understanding of our proposed repository: 1) the characteristics of the model representation of software artifacts, and 2) the architecture of the repository itself.

#### 3.1 Software Component Metamodel

To enable the correct processing of the components the repository needs to manage, and the integration of information to and from other solutions, it is imperative to have a model representation of the software elements. Therefore, we have defined a metamodel with enough expressivity to capture all the information that we need, but at the same time hiding non needed data.

From this metamodel, a model instance describing each software artefact, its capabilities and its needs can be created. We have named these model instances Deployment Descriptors.

Although the metamodel will be primarily used to represent OSGi-related artefacts, it has been designed to support without modifications other elements, such as non-bundle JARs or additional component/service models (such as EJBs, Spring beans or Web Services). The metamodel aims to capture all the relevant information of all types of software elements. This is enabled by the concept of Resource, which we have adopted from the OMG Deployment & Configuration [13] standard.

Figure 1 depicts a subset of the metamodel. As can be seen in it, Resources are the main building blocks. A Resource represents any logical or physical manageable system element, and it is defined by three fixed parameters (name, version and type) common to all Resources and an undefined list of specific Properties for each Resource type.

The core element is the Deployment Unit. Deployment Units represent the artifacts which can be deployed over the environment containers. In an OSGi context, Deployment Units represent OSGi bundles. Conceptually, a Deployment Unit would be the lowest abstraction level of our software model, being the unit of software distribution. The Deployment Unit is composed by a set of children Resources, Dependencies and Constraints that provide computable information about the developer,

software license, packaging type, exported packages, logical dependencies and hardware compatibility restrictions.

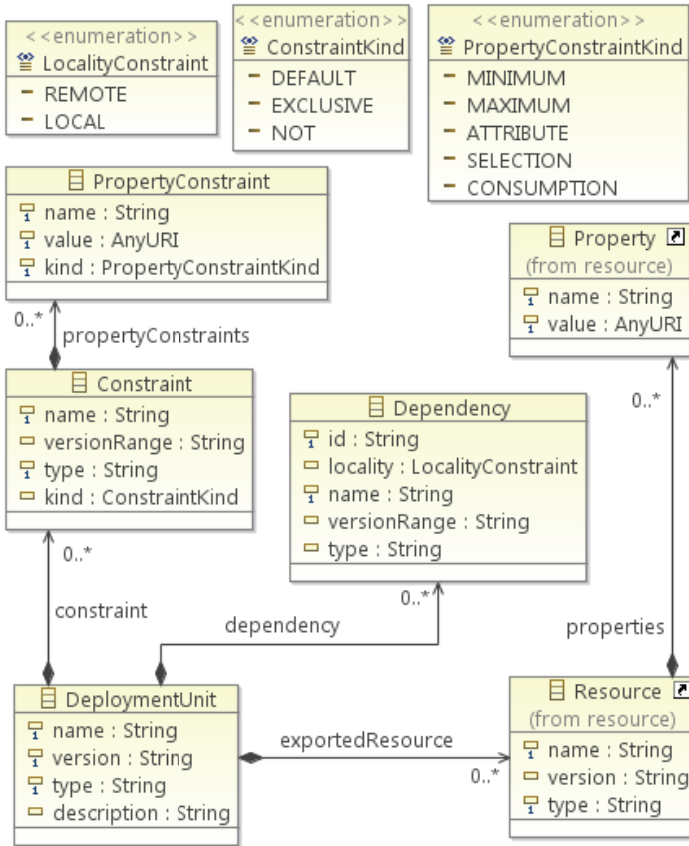


Fig. 1. Software component metamodel

Dependencies represent the required physical and logical dependencies needed in order to assure a smooth running of the Deployment Unit. Two types of elements can satisfy a Dependency: specific Resources or whole Deployment Units. This represents the two main mechanisms defined in OSGi: Require-Bundle and Import-Package. In addition to this, the metamodel allows us to further describe the type of Dependency, enabling to differentiate requirements on remote resources such as Web or REST Services from requirements that must be satisfied by a unit in the same host (e.g. as it provides a Java package). This aspect is identified by a locality parameter that could be remote or local.

Finally, Constraints enable the expression of requirements over the runtime execution environment, each one requiring a specific Resource to be present at (or absent from) the environment. Following this definition, we have defined three kinds of Constraints, depending on the required behaviour: *default* (to be present), *exclusive* (to be present and not used more than once) and *not* (to not be present). To further

extend the Dependency and Constraint models, Properties can be defined. Each needed Property can be defined by a name, an evaluation function and a threshold value.

As an example, a typical Constraint would identify a Resource of type “hardware.processor” with an additional Property “speed” of a kind “minimum” and an expression value of “2000”. This means that the Deployment Unit requires a micro-processor with a minimum speed of 2 GHz.

Both Dependencies and Constraints are shown in Figure 2.

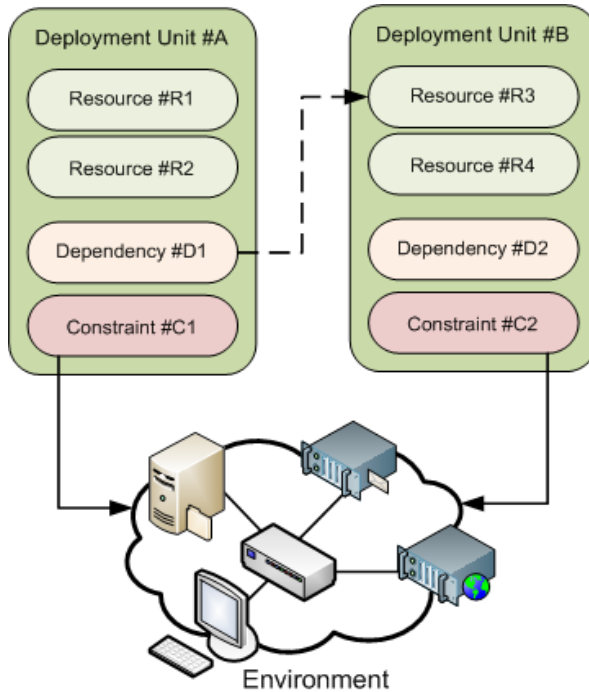


Fig. 2. Dependencies and constraints

The metamodel allows us to represent all the OSGi-specific mechanisms, as well as expressing important information that is not contemplated on the original format (the manifest file) or the information models of other repository solutions. This means that a conversion from one of those solutions into our proposal would not result in the loss of information, although the opposite would.

### 3.2 Repository Architecture

The need to federate with multiple types of repositories, as well as evaluating component dependency taking into account multiple factors have motivated us to design the architecture of the repository with a modular and extensible approach. We have selected the OSGi platform as the base technology to achieve those requirements. On a side note, this allowed us to test the system from the start, in order to check whether



the repository was able to host itself successfully. Figure 3 shows a layered view of the repository components.

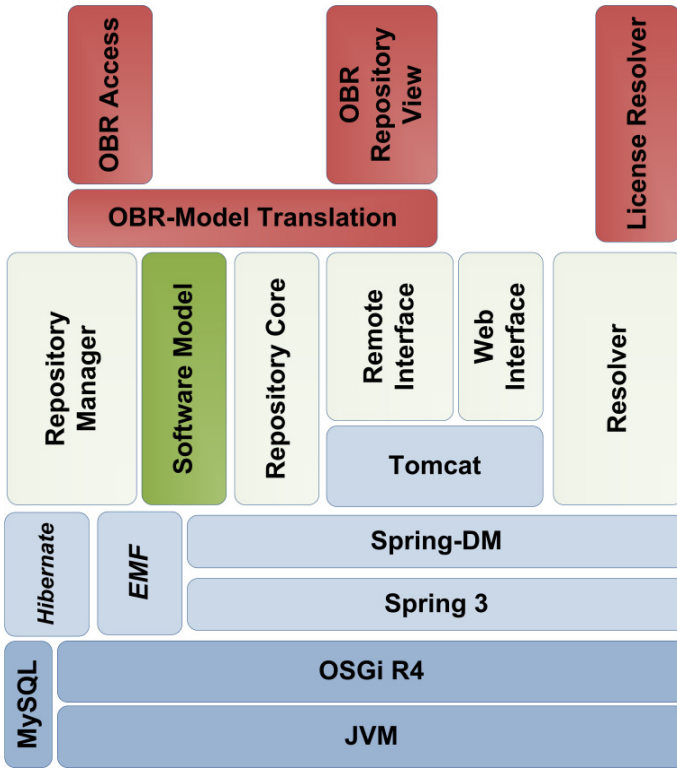


Fig. 3. Repository architecture

In the lower levels lie the Java Virtual Machine, the OSGi framework, and a database solution for storing the relevant information. On top of it are the basic OSGi components from third parties that are needed for the repository to work. The repository uses Spring Dynamic Modules for structuring the inter-bundle communications. The EMF (Eclipse modeling Framework) components enable the definition of the metamodel and provide the tools needed to work with them programmatically. Hibernate Provides an ORM (Object-Relational Mapping) interface with the database system. Finally, the Apache Tomcat bundles embed a lightweight application server that will host the remote access interfaces developed for the repository.

The next layer contains the basic components of the repository:

- **Software Model:** Provides the metamodel defined at the previous section, as well as Java bindings and marshalling mechanisms.
- **Repository Core:** The basic component of the repository. Provides the main service interfaces of the components and defines the extension semantics.

- **Repository Manager:** This component manages the physical artifacts and the component information. It provides CRUD (Create-Read-Update-Delete) operations over the managed Deployment Units,
- **Web Interface:** Web-based graphical user interface that allows human users to browse the repository contents.
- **Remote Interface:** Exposes a REST interface that enables the communication between the repository and other software.
- **Resolver:** The component that processes and resolves Deployment Unit dependencies, obtaining unit closures that work correctly together.

Finally, the topmost level of the diagram shows some extensions to the repository that expand the base functionality to federate with an additional type of repositories (OBR), and apply additional criteria for the dependency resolution. Over the next sections we will present additional details on the federation and resolution capabilities of the repository.

### 3.3 Faceted Dependency Resolution

This modular architecture makes possible the easy expansion of the repository capabilities. This feature is used to define different types of dependencies, each one resolved by a different component. Moreover, this structure enables the definition of a faceted dependency resolution engine. In it, there are not only several dependency types, but also additional conditions that the candidates to satisfy one need to comply. These conditions are called Facets.

An example of a Facet is the license compatibility. It is perfectly possible that a Deployment Unit satisfies every dependency that another has, but at the same time do not be valid because their licenses are incompatible. Other Facets could be security settings, packaging procedures or execution requirements. Each Facet can be added to the resolution engine as an OSGi bundle, and it offers its features as services that are called by the resolution core component.

An example of this process that uses a License Compatibility Facet we developed is shown in Figure 4. In it a user calls the resolver with the intention of knowing the dependencies needed for a Deployment Unit (DU). The resolver processes every Dependency, looking for other Deployment Units that can satisfy it. After some of them have been found, the resolver searches for Facets that need to be checked and, after finding one (License Compatibility), makes use of it. In this particular example only one unit is valid after this check.

The integration of new facets to the dependency resolution is straightforward, and the license compatibility is just one example of application. As the diagram shows, the second loop will check each DU for every detected facet.

### 3.4 Repository Federation

To enable the integration between open source components it is not enough to just resolve their dependencies. It is also necessary to be able to provide the artifacts that fulfill those requirements.

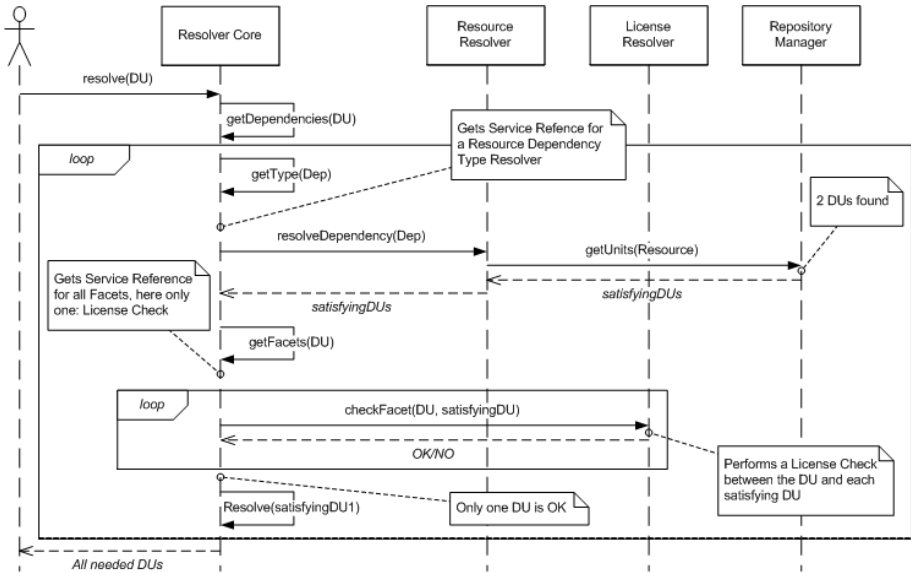


Fig. 4. Faceted dependency resolution

Since open source communities are fragmented and each one uses different tools and techniques, the repository needs to access and understand the information that lies in other repositories. To achieve this end, the federation capabilities allow our repository to communicate with external types of repositories currently available. Federation support is designed with extensibility in mind, and each technology extension can be federated just by providing three services:

- Model transformation service between our information model instances and the format the target solution uses to represent software artifacts.
- Remote manager service that accesses the information contained in the external federated repositories.
- Remote interface service that can be accessed by the target solution repositories. This is only possible if the target solution has some kind of federation support for repositories of its own type.

For a more detailed explanation of how these features can be implemented we will use OBR as an example of a target solution. An example of an infrastructure that uses this two-way federation is depicted in Figure 5.

**Sample Integration: OBR.** The OSGi Bundle Repository RFC is a draft standard for providing a common interface to distributed OSGi repositories. Its official nature, alignment to OSGi concepts and the explicit acknowledgement of federation requirements make it an ideal candidate for testing our federation approach. Here we present how we achieve two-way integration between our repository instances and federated OBR repositories.

We talk about two-way federation, as we both act as OBR providers and consumers. For external OBR repositories, we offer an OBR view that can be used by them in their standard federated dependency resolution processes. Additionally, our repository can handle a list of external OBR repositories, and can delegate dependency

resolution requests to the distributed OBR instances. Both approaches of federation are achieved through the same method: Model transformation from our generic model to the specific component model defined by OBR.

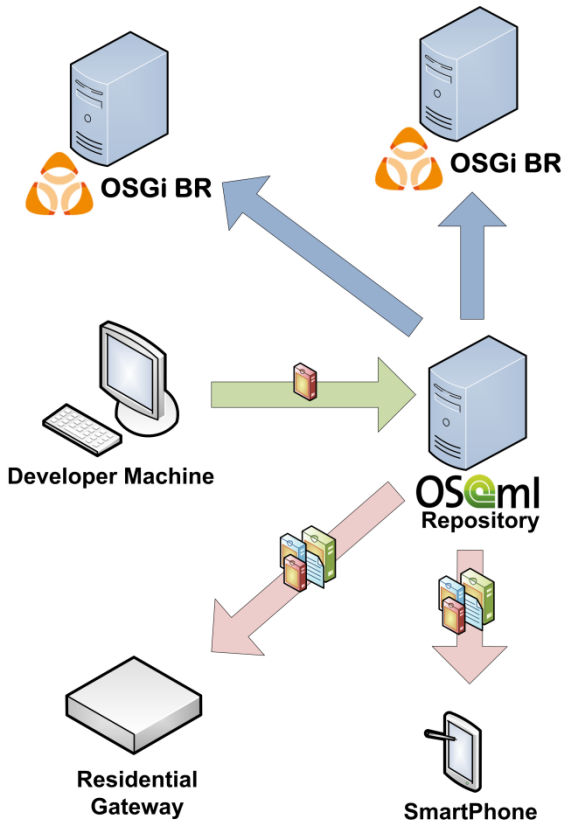


Fig. 5. Repository federation

For maximizing extensibility, the OBR model does not explicitly rely on OSGi concepts. The repository works with resources, identified by a symbolic name and a version. Additionally, a download URL is provided for each element. Each resource contains two types of declarations. First, resources offer a list of capabilities to the environment. They represent either the whole element (named *bundle*), or a software element such as a java package (named *package*). Each capability is further refined through properties, which have a name, value and value type (e.g. String or number). The second kind of elements are requirement statements, that demand the presence of resources in the resolved configuration. They model logical requirements that must be satisfied. This specification's concepts can be mapped to a subset of the Deployment Unit model.

Figure 6 presents an example mapping between both models. Every OBR concept has an equivalent definition in our abstractions. The OBR resource plus the bundle capability elements are mapped to the base Deployment Unit concept (The definition

of units as resource subclasses allows this). Additional capabilities are mapped to unit exported Resources, such as the presented java package. Resource visibility information is lost, which is not problematic for OSGi-specific elements (all of them are local), but presents the limitations of OBR for reasoning over distributed physical environments. Additionally, each OBR requirement is mapped into a logical Dependency. All the information derived from the Constraints from our model has no equivalent. This bears no impact from the OBR perspective, as our repository provides all the required information. On the other hand, the use of federated OBR repositories by our specific instance can result in lesser-quality results, in cases when physical concerns need to be evaluated.

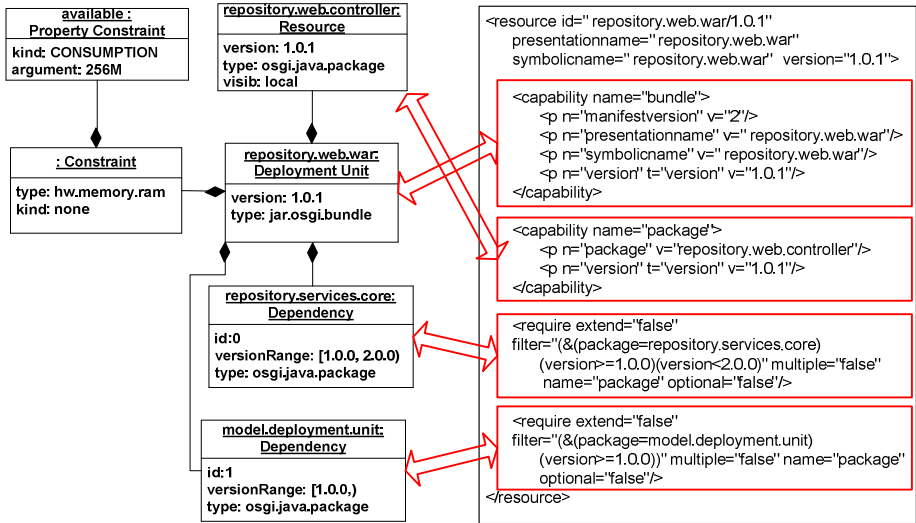


Fig. 6. Model mapping between OBR and the presented model

## 4 Validation

ITEA-OSAMI is an european project which was executed by 34 partners from both the academia and enterprise. The objective of this project was to develop open source common foundations for a distributed, dynamic service-oriented platform.

Consortium partners came from multiple domains (healthcare, personal, and mobile office), followed different software development processes, and depended on existing open source resources from different communities. This created a need for a centralized platform that eased partner integration.









































The repository presented in this article has been developed and deployed in order to address the project requirements. It has widely succeeded at this task, becoming the central point of the ITEA-OSAMI ecosystem. ITEA-OSAMI’s running instance of the repository could be publicly accessed at the time of writing of this article<sup>1</sup>. (Figure 7 shows the visual aspect of the web interface, which is listing several units already resolved).

<sup>1</sup> <http://repository.osami-commons.org>



# OS@ml *Bundle Repository*

[Main page](#)
[Find Resources](#)
[Upload Unit](#)

Search results:

Name	Version	Description	Actions
es.upm.dit.osami.repository.client	1.0.0		   
es.upm.dit.osami.model.sw	1.0.0		   
org.springframework.core	3.0.2		   
com.springsource.org.apache.commons.logging	1.1.1		   
org.springframework.cxm	3.0.2		   
org.springframework.beans	3.0.2		   
org.eclipse.core.runtime	3.5.0		   
org.eclipse.osgi	3.5.1	%systemBunde	   
org.eclipse.equinox.common	3.5.0		   
org.eclipse.core.jobs	3.4.100		   

24 Deployment units found, displaying 1 to 10.

Showing page 1 of 3  

**Fig. 7.** Screenshot of the repository

After assessing partner concerns, we provided two extensions to the repository. Regarding federation, an OBR extension was developed, as it was mandatory to support OBR-based deployment clients as well as accessing open source bundles developed at two third-party repositories. Additionally, since the beginning of the project open source license management was a potentially conflicting aspect among partners. However, these issues were addressed with the definition of a license-aware dependency Facet, based on the dependency compatibility analysis module from another project partner.

In this context, we have validated the proposed metamodel, as well as the defined architecture and extensibility capabilities. It has successfully been used by all the project partners, providing a common integration point for the developed open source software and services, both internally created and from the main open source communities.

## 5 Conclusions and Future Work

In this article we have proposed an architecture for a repository for the integration of software artifacts, with a special focus on OSGi bundles. This repository has been designed around an information model for software components, which manages to show all relevant data while hiding the undesired complexities.

We have also shown how this architecture has been created to be extensible. Using this modularity we have demonstrated how it can be easily expanded to support two important features:

- Faceted dependency resolution: Offers support for an unlimited number of conditions that dependencies are forced to respect.
- Two-way federation: Enables our solution to access contents available in other repositories and at the same time expose itself to them.

We also have developed a license compatibility Facet based upon existing open source work and the components needed to achieve federation with OBR repositories.

Finally, our work has been validated in the context of the ITEA-OSAMI European project, where it has been subjected to an intensive use by more than 30 partners from different countries and sources (universities, research centers, software and telecom companies, etc).

Concerning future developments, the most straightforward way to improve the already existing work is through the support of more repository technologies for federation and new dependency Facets. In the first field, federation with Eclipse P2 would be the most interesting repository to support, since it sees wide use in several communities. About dependency Facets, more of them could be developed, taking as an example the license check already created. These components are relatively easy to implement thanks to the infrastructure of our proposal.

Not directly related with this, but also interesting, are the possibilities for the repository to work in a cloud environment. This line of work is concerned not only with the deployment of the repository itself, but also with how it can manage the software artifacts of several types of cloud solutions (IaaS, PaaS or SaaS). To support features like these the information model would probably need to be extended and the architecture of the repository revised.

Using the capabilities of OSGi plus some extensions already in the making<sup>2</sup>, the possibility of extending OSGi for a complete PaaS solution is turning into a reality. If this possibility finally materializes, the proposed repository could be expanded to work in this kind of environment.

**Acknowledgements.** The work presented in this article has been performed in the context of the European project ITEA-OSAMI, under grant by Spanish Ministerio de Industria, Turismo y Comercio in the PROFIT program.

## References

1. Deshpande, A., Riehle, D.: The Total Growth of Open Source. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) Open Source Systems. IFIP, vol. 275, pp. 197–209. Springer, Boston (2008)
2. Evelson, B., Hammond, J.: The Forrester Wave: Open Source Business Intelligence (BI), Q3 2010, Forrester Research (2010)

---

<sup>2</sup> [http://www.osgi.org/wiki/uploads/Design/rfp-0133-Cloud\\_Computing.pdf](http://www.osgi.org/wiki/uploads/Design/rfp-0133-Cloud_Computing.pdf)

3. OSGi Alliance, OSGi Service Platform Release 4 Version 4.2 Specifications (June 2009)
4. Massol, V., Van Zyl, J., Porter, B., Casey, J., Sanchez, C.: Better builds with Maven. Merge Inc. (2006)
5. Hall, R.S.: OSGi RFC-0112 Bundle Repository (February 2006)
6. Le Berre, D., Rapicault, P.: Dependency management for the Eclipse ecosystem: Eclipse P2, metadata and resolution. In: Proceedings of the 1st International Workshop on Open Components Ecosystem. ACM (2009)
7. Rubio, D.: Pro Spring Dynamic Modules for OSGi<sup>TM</sup> Service Platforms. Apress (2009)
8. Iyengar, S.: A universal repository architecture using the OMG UML and MOF. In: Proceedings of the Second International Enterprise Distributed Object Computing Workshop, EDOC 1998 (1998)
9. Kraan, W., Mason, J.: Issues in Federating Repositories, A Report on the First International CORDRAtm Workshop. D-Lib Magazine 11(3) (2005)
10. Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., Tansley, R., Walker, J.H.: DSpace, An open Source Dynamic Digital Repository. D-Lib Magazine 9(1) (2003)
11. Van de Sompel, H., Lagoze, C., Bekaert, J., Liu, X., Payette, S., Warner, S.: An Interoperable Fabric for Scholarly Value Chains. D-Lib Magazine 12(10) (2006)
12. Van de Sompel, H., Chute, R., Hoschstenbach, P.: The aDORe federation architecture: digital repositories at scale. International Journal on Digital Libraries 9(2)
13. Object Management Group. Deployment and Configuration of Distributed Component-based Applications Specification. Version 4.0 (April 2006)



# Automated System Testing of Dynamic Web Applications

Hideo Tanida<sup>1,\*</sup>, Mukul R. Prasad<sup>2</sup>, Sreeranga P. Rajan<sup>2</sup>, and Masahiro Fujita<sup>1</sup>

<sup>1</sup>The University of Tokyo, Tokyo, Japan

<sup>2</sup>Fujitsu Laboratories of America, Sunnyvale, CA, U.S.A.

tanida@cad.t.u-tokyo.ac.jp, fujita@ee.t.u-tokyo.ac.jp,  
{mukul.prasad,sree.rajan}@us.fujitsu.com

**Abstract.** Web applications pervade all aspects of human activity today. Rapid growth in the scope, penetration and user-base of web applications, over the past decade, has meant that web applications are substantially bigger, more complex and sophisticated than ever before. This places even more demands on the validation process for web applications. This paper presents an automated approach for the system testing of modern, industrial strength dynamic web applications, where a combination of dynamic crawling-based model generation and back-end model checking is used to comprehensively validate the navigation behavior of the web application. We present several case studies to validate the proposed approach on real-world web applications. Our evaluation demonstrates that the proposed approach is not only practical in the context of applications of such size and complexity but can provide greater automation and better coverage than current industrial validation practices based on manual testing.

**Keywords:** Dynamic analysis, Validation, Web application.

## 1 Introduction

Web applications are ubiquitous today. The last decade has witnessed rapid growth in both the scope and the penetration of web applications. On one hand, the wide-scale adoption of web applications in all spheres of human activity has brought validation and quality assurance of such applications into focus. On the other hand, the development of WEB 2.0 technologies such as AJAX (Asynchronous JavaScript and XML) and Flash has resulted in feature-rich and highly interactive web applications, which are even more difficult to validate.

Current industrial practice for the functional validation of web applications still continues to largely rely on manually written test cases which exercise and check the application behavior one trace at a time. There is a growing gap between the coverage, automation and scalability of traditional testing-based validation methodologies and the validation requirements of modern WEB 2.0 applications, which has been acknowledged by validation researchers and practitioners alike. Research on automated

---

\* This author was a research intern at Fujitsu Laboratories of America, when this work was done.

model generation [9], model-based testing [10,8,7,3], and model checking [1] offers the promise to address this gap. Specifically, there has been a recent work on automated model generation [9] and model-based testing [10] of AJAX applications that looks especially promising.

This paper addresses the problem of developing a better and practical validation solution for WEB 2.0 application development. We propose a methodology for functional validation of WEB 2.0 AJAX applications that is based on an efficacious combination of some previously proposed techniques in the validation literature and our own novel extensions to these techniques. We present several case studies of applying this methodology to the validation of WEB 2.0 applications. The main objectives and contributions of this paper are as follows:

- We propose a solution for automated system testing of modern industrial-strength dynamic web applications. This solution employs a combination of automated dynamic crawling to extract a model of the navigation behavior of the web application and model checking techniques to check this model for various properties of interest.
- We extend and adapt the dynamic crawling and model checking techniques in several novel ways to fit our application domain and to ensure that the validation solution is simple, scalable, automated and applicable in an industrial context. We believe our solution is fairly complementary to current industrial practices of web application validation and at least in some respects, superior to them.
- We present several case studies of applying the proposed approach to the validation of real web applications and report on both the successes and short-comings of our proposed approach. We feel these lessons would be vital in developing and delivering the next generation of industrial practices for web application validation.

The rest of the paper is organized as follows. In the next section we survey related work. Section 4 presents our proposed validation approach. In Section 5 we describe the implementation of this approach. Section 6 presents three case studies evaluating our approach on real-world web applications, followed by a discussion of the lessons learnt in Section 7. We summarize and conclude the paper in Section 8.

## 2 Related Work

This work is aimed at system testing of dynamic web applications and specifically, validating the navigational aspects of their behavior. The vast body of research on web application validation spans several other important areas such as validation of the server-tiers of web applications, or that of security or performance aspects of the behavior. Nevertheless, these areas are beyond the scope of this paper and are therefore not surveyed in this section.

Current industrial practice for system testing of web applications primarily involves the use of capture-replay tools such as Selenium [1], WebKing [2] and Sahi [3]. Using these

---

<sup>1</sup> <http://seleniumhq.org/>

<sup>2</sup> [http://www.parasoft.com/jsp/solutions/soa\\_solution.jsp?itemId=86](http://www.parasoft.com/jsp/solutions/soa_solution.jsp?itemId=86)

<sup>3</sup> <http://sahi.co.in/w/>

frameworks, users manually exercise the application through various test scenarios, one at a time. These actions are recorded by the tool and can be replayed back at a later time, usually with user-defined assertions expressing expected behavior, inserted at various steps. This mode of validation, however, requires a substantial amount of manual effort.

Our proposed method for system testing of dynamic web applications involves extracting a state-based navigation model of the web application behavior by automatically crawling the deployed web application. This model is then checked against a temporal logic specification, represented as a set of properties, using model checking [4] techniques. There are several works which overlap with one or more aspects of our approach but differ in other respects.

**The Target Applications.** Several previous works [12][3][2] target traditional (WEB 1.0) web applications, employing some form of automatic crawling to extract a navigation model for validation. However, as also pointed out by others [10][8], the crawlers used there would not be applicable to WEB 2.0 applications. Further, the nature and scope of the model extracted from traditional web applications as well the properties to be validated on them would differ substantially from those of the dynamic web applications we target. This is also true of previous work on GUI Application testing by Memon *et al.* [13], which, while qualitatively similar in many respects, cannot be directly applied to our application domain. The tool MCWEB [1] is one of the few instances of the direct application of model checking to web application navigation behavior. However, the work was also targeted towards WEB 1.0 applications. Further, the lack of support for automated model extraction and the use of  $\mu$ -calculus for specifying properties makes the tool difficult to use for non-formalists. Our approach emphasizes automation, scalability and ease of use in an industrial setting.

**The Verification Methodology.** Almost all previous papers rely on trace-by-trace testing as the end means to validate the behavioral model. The authors of [10][8] automatically generate test-benches which exercise one trace at a time from the model. While this definitely increases the level of automation compared to the current industrial practice of manually written test-cases, the underlying validation is still test-case driven and hence the requirements and their checking very trace-specific. We submit that our approach, which is based on model checking, is much more natural, given that we have a pre-generated navigation model. Further, we can pose and check more general and global properties of the application. We present several instances of this and the advantages it provides, in Section 6.

### 3 Background

In this section we review some technologies that form the foundations of our approach for web application validation.

#### 3.1 Automated Crawling of AJAX Applications

We use the technique proposed in the CRAWLJAX work [9] as the basis for exploring the behavior of the web application under test. CRAWLJAX is a tool for automatically

exploring the dynamic state space of modern web applications. It is capable of interacting with the client-side code of a web application through programmatic interfaces that are available for most of the popular web browsers. CRAWLJAX analyzes a web page to detect widgets to click on (clickables), and systematically exercises them to explore dynamic web application behavior. Changes in the dynamic DOM tree of the page are detected and recorded as new states of the behavior. By systematically detecting new states and executing clickables on them the crawler is able to build a finite-state model of the navigation behavior of the web application. CRAWLJAX provides a set of options to configure the crawling behavior. For example, the set of widgets to click on or not click on, during crawling, can be specified. For more details about the algorithms, architecture and features of CRAWLJAX the interested reader is referred to [9]. We have extended the basic CRAWLJAX in several ways for the purposes of this work. These extensions are discussed in Section 4.

### 3.2 Model Checking

*Model Checking* [4] is a set of automated techniques for checking if the behavior of a hardware or software system satisfies a certain property. This is typically done by extracting a finite-state abstraction  $M$ , of the relevant behavior of the system under test. The property to be checked is expressed as a logical formula  $f$  in a *temporal logic*. Subsequently, model checking algorithms are applied to check if  $M$  satisfies  $f$ . A temporal logic is a formalism for expressing sequential properties of dynamic systems, for the purposes of automated reasoning through techniques such as model checking. There are several temporal logics that have been proposed in the literature, varying in their syntax, expressive power as well as the complexity of the model checking algorithms that work on them. CTL (computation tree logic) and LTL (linear temporal logic) are the two most popular ones. Our property checking approach, proposed this paper, is a simplification of traditional temporal logic model checking.

## 4 Proposed Method

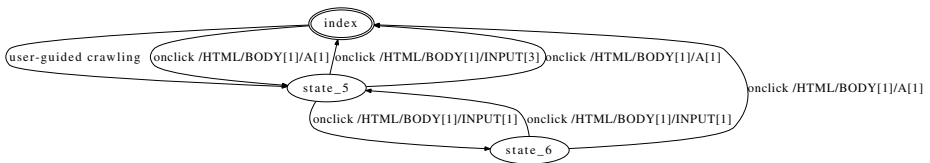
This work addresses the validation of modern, *dynamic* applications. These applications usually support a feature-rich, highly-interactive, client-side user-interface, typically by the use of technologies such as AJAX and Flash. Further, this work focuses on validating the navigational aspects of the behavior of such application (as opposed to, for example, the performance, security or concurrency aspects of the behavior).

Our overall approach is a two step process. The first step is to extract a finite-state model of the navigation behavior of the web application. This is done by automatically crawling the web application in the style of CRAWLJAX and capturing the observed behavior as a navigation model. In order to facilitate a more comprehensive, yet scalable crawling behavior, applicable to industrial-strength applications, we propose an extension to CRAWLJAX's basic crawling, called *guided crawling*. This is described in Section 4.1. The second step is to validate various functional requirements of the application against the navigation model. To do this we employ a variant of traditional temporal logic model checking, called template-based property checking. This is described in Section 4.2.

This two-step approach has several advantages. First, it allows us to isolate the relatively expensive crawling step from the actual validation and do the crawling once (or a few times) in a mostly requirement and validation independent manner. Second, it allows us to extract a compact model of the navigation behavior, by a judicious choice of the model representation as well as by discarding irrelevant application-level details during crawling. This accelerates both the crawling and the downstream model checking. Third, since all the requirements to be checked are often not known during the initial stages of validation, performing the validation offline obviates the need to repeat the expensive crawling step.

#### 4.1 Model Generation

**The Navigation Model.** The navigation model represents the structure and screen content observed by the crawler while dynamically crawling the web application. It is comprised of a *state transition graph (STG)*, representing the topology of the crawled behavior, as well as the content of each crawled state.



**Fig. 1.** Graphical view of a State Transition Graph

Fig. 1 shows an example of an STG. An STG is a labelled directed graph where each node (state) corresponds to a web page viewable on a web browser. Each state is represented by the DOM (document object model) of its corresponding web page, in the navigation model. A change in the DOM of a web page, typically through the execution of a user action (such as a button click), constitutes a new state. Edges in the STG represent transitions from one state to another and are typically labelled with the user action (e.g., a click) and the XPath of the element on which it was executed. This is the case with most edges in Fig. 1. Some edges are labelled “user-guided crawling” and correspond to a transition made by a sequence operations from a *guidance directive*. Guided crawling and guidance directives are discussed in the next section.

Note that some web applications may, theoretically, have an infinite state space (for example based on infinite different sets of data inputs). However, due to obvious practical constraints of crawling time and navigation model size we only crawl and validate a finite but behavior-rich subset of the state space of a given web application. A finite state model is also a requirement of the downstream property checking algorithms.

**Guided Crawling.** The CRAWLJAX tool [9] offers users some control over the crawled behavior by specifying the overall set of widgets to click or not click during crawling. Users can also specify one or more sets of data for each HTML `<form/>` element encountered during crawling. However, the crawling of real-life enterprise web applications often requires a more tighter control over the crawling, for example, in the following practical scenarios.

1. *User Authentication*: This is a common requirement in several web application interactions, when viewing confidential information or executing transactions. However, typical web application interactions are a complex mix of unauthenticated and authenticated behavior, with authentication being activated under specific scenarios (e.g. some applications like Amazon.com do not require a login till the checkout stage).
2. *Form Data Filling*: HTML forms are commonplace in modern web applications (a user authentication panel is a special instance of this). CRAWLJAX allows form-data filling but always fills a given form with the same data (or data-sets). For more intelligent crawling, it would be desirable to fill a given form with one of several data sets driven by the scenario being navigated.
3. *Excluding Behavior from the Model*: When crawling real web applications, the crawling time as well as the size of the crawled model needs to be managed, according to available computation resources. One strategy is to surgically exclude features and crawl scenarios outside the scope of the ensuing validation, from the crawling.

It is very difficult, if not impossible, to adequately service the above scenarios (and many others), using the default crawling controls provided by CRAWLJAX. We propose a technique called *guided crawling* to provide the user with more direct control over the crawled behavior. It allows the user to specify scenario-based desired crawling behavior. This is done by creating one or more *guidance directives*, specific to the target application being crawled. The crawling alternates between the fully automatic default crawling and the scenario-specific behavior specified by the guidance directives.

**Definition 1 (Guidance Directive).** A *Guidance Directive*  $G = (p, \mathcal{A})$  is an ordered pair that consists of a predicate  $p$  that is evaluated on a web application state, and an action sequence  $\mathcal{A}$ .  $\mathcal{A} = (\alpha_1, \alpha_2 \dots, \alpha_k)$  is a sequence of atomic actions  $\alpha_i$ . Each atomic action  $\alpha = (e, u, \mathcal{D})$  is a triple consisting of a DOM element  $e$ , a user-action  $u$  and a set of data-instances  $\mathcal{D}$  (potentially empty) associated with  $u$ .

As per Definition 1, a guidance directive  $G$ , includes the predicate  $p$  that determines when  $G$  should be activated.  $p$  is evaluated on the current state of the web application, during crawling, i.e., on the DOM of the current page loaded in the web browser. If  $p$  is true in the current state, the crawling action sequence  $\mathcal{A}$  is executed on the web application. Each atomic action  $\alpha$  in  $\mathcal{A}$  is a simple (browser-based) user action  $u$  on a particular DOM element  $e$  on the current web-page/screen. For example,  $u$  could be a click and  $e$  could be a button. Such actions have no associated data. Hence,  $\mathcal{D} = \emptyset$  (the empty set) in this case. Another example of an action would be selecting an option from a `<select/>` element or assigning a string value to an `<input/>` element etc. In these cases  $\mathcal{D}$  would be the set of data values to exercise the element with.

Algorithm 1 presents the pseudo code for our model generation, incorporating guided crawling. The main procedure, `GuidedCrawl` accepts a target web application,  $W$  and a set of associated guidance directives,  $\mathcal{G}^{set}$ . It initializes the navigation model  $M$ , loads the initial web-page (`InitPage`) of  $W$  in the web browser and invokes procedure `GuidedCrawlFromState` on it. `GuidedCrawlFromState()` does the actual crawling and recursively calls itself on new successor states. `GuidedCrawlFromState()` starts with a check

(*IsVisited(S)*) to see if state  $S$  has been visited by a previous invocation of *GuidedCrawlFromState*. This check accounts for any specified state-abstractions (explained in Section 19). If so, the crawling returns back to calling state. If not, *MarkVisited()* marks state  $S$  as visited, to exclude it from future guided crawls, and *AddState()* records  $S$  in the navigation model  $M$  as a newly discovered state. Next, the state  $S$  is analyzed to compute the set of actions (*Actions*), to execute on it, to continue the crawling. First, function *FindActions()* (line 7) computes the set of basic (non-guided) user actions, which can be executed on it, based on the clickables specified to the crawler. Next the set of guidance directives,  $G^{set}$  is processed to find additional actions to execute on  $S$  (lines 8 – 11). For each guidance directive  $G$  that can be activated on  $S$  (line 9) function *ComputeActionSequences()* computes concrete action sequences of actions from  $G$  by picking specific data values in its constituent atomic actions  $\alpha$ . All possible sequences that can be created by various choices of the specified data-values are constructed and added to the *Actions* set. Subsequently, each action  $a$  in *Actions* is fired on  $S$  (lines 13 – 18). *Execute()* Executes the action (or action sequence)  $a$  on  $W$  to discover a next state (*nextState*) and *AddTransition()* records this transition in model  $M$ . *GuidedCrawlFromState()* is then recursively called on *nextState* (line 14). *UndoTransition()* functionally reverses the transition  $S \rightarrow nextState$  on  $W$  to restore it to state  $S$ .

**Model Reduction.** The size of the navigation model has a direct bearing on the efficiency of the property checking. We employ the following two features, to derive a compact and meaningful model for validation.

1. *Specifying User Events:* CRAWLJAX provides several mechanisms for specifying the set of widgets to be exercised (or excluded) during crawling. Our guided crawling technique further supplements these mechanisms. The specification is done by the user on application-specific basis, as an input to the model generation step.
2. *State Abstraction:* Since the crawler uses the screen DOM as the unique identifier for a state, identical looking screens can often be mapped to different states in the model because of slight differences in their DOMs. This can happen because of entities such as visit counters or date/time stamps, included in the DOM or even minor differences in white-spacing or the attribute order in dynamically generated web-pages. Therefore, we have implemented a state abstraction technique which accepts a set of user-given XPath's and removes all matching elements and their descendents from the DOM tree of each state and uses the resulting abstracted DOM as the state identifier, specifically in determining equivalence of two states. This technique is implemented within the *IsVisited()* function in Algorithm 4.

Our experience regarding the specific use of these features, in the light of our case studies, is discussed in Section 6.

## 4.2 Model Validation

As mentioned in Section 2 one of the key differences between our approach and prior art in this area is that while other approaches resort trace-by-trace checking of the behavior

---

<sup>4</sup> CRAWLJAX provides a similar, albeit independently developed, mechanism called *oracle comparators* for state abstraction.

**Algorithm 1.** Guided Crawling

---

```

/* GuidedCrawl ( $W, \mathcal{G}^{set}$ ) -- main procedure */
Input :  $W$ : Web application under test
          $\mathcal{G}^{set}$ : Set of guidance directives
Output:  $M$ : Crawled navigation model

1 begin
2    $M = \emptyset$ 
3    $InitPage \leftarrow LoadBrowser(W)$ 
4    $GuidedCrawlFromState(InitPage, M)$ 
5   return  $M$ 
6 end

```

---

```

/* GuidedCrawlFromState ( $S, W, \mathcal{G}^{set}, M$ ) */
Input :  $S$ : Current state for guided crawling
          $W$ : Web application under test
          $\mathcal{G}^{set}$ : Set of guidance directives
          $M$ : Navigation model being built

1 begin
2   if  $IsVisited(S)$  then
3     | return
4   end
5    $MarkVisited(S)$ 
6    $AddState(S, M)$ 
7    $Actions \leftarrow FindActions(S)$ 
8   foreach  $\mathcal{G}(p, \mathcal{A}) \in \mathcal{G}^{set}$  do
9     | if  $p(S) = true$  then
10    | |  $Actions \leftarrow Actions \cup ComputeActionSequences(\mathcal{A})$ 
11    | end
12  end
13  foreach  $a \in Actions$  do
14    |  $nextState \leftarrow Execute(a, W, S)$ 
15    |  $AddTransition(nextState, S, M)$ 
16    |  $GuidedCrawlFromState(nextState)$ 
17    |  $UndoTransition(a, W, S)$ 
18  end
19 end

```

---

a la traditional testing, we propose to check the navigation model as a whole using the formal technique of model checking [4]. The use of a pre-generated, finite state navigation model makes the application of model checking both easy and very efficient. We claim that the expected navigational behavior of web applications can be quite naturally expressed as properties in temporal logic [4], the input language of model checkers. In the following we present a few examples of such classes of requirements and specific instances in each class.



1. *Screen Sequence/Transition Requirements:* The simplest and most common check on web applications is of the form: *A user input  $i$  with the web application on Screen  $A$  takes it to Screen  $B$ .* Here, screens  $A$  and  $B$  may be screens or pages of the web application, identified by the presence or absence of certain features, widgets or DOM elements, while input  $i$  may be a simple input like a mouseover or button/link click or a more complicated sequence of such actions interspersed with data inputs to various widgets on the screen. This kind of requirement may be further generalized in checking (for example) that Screen  $B$  follows  $A$  in one, all or none of the valid execution sequences of the web application. Some specific examples of this class of requirements could be:
  - In a web application with user authentication: *The LOGOUT screen is always preceded by the LOGIN screen*
  - On a utilities web-site under the bill-payment section: *If the CONFIRM button is clicked on the PAYMENT-DETAILS screen then the next screen is always the RECEIPT screen.*
2. *Global Navigation Structure or Usability Requirements:* This kind of requirement typically apply to the overall structure of the web application's navigation behavior. Thus, they are by their very intent, *global* and ideally suited for checking on a single, consolidated navigation model (versus conventional trace-by-trace checking). Some examples of requirements in this class are:
  - *All features of are accessible within 5 clicks, starting from the home page*
  - *The initial page is accessible from every screen*

**Temporal Property Templates.** Since each screen of the web application and all features, widgets or DOM elements on each screen are represented as part of our navigation model, it would in theory be possible to express our requirements as properties in a temporal logic [4] such CTL (Computation Tree Logic) or LTL (Linear Temporal Logic). With minor modifications to our navigation model, these properties could then be checked on it using one of the commonly used model checkers, e.g., NuSMV [11].

However, writing properties in temporal logic is quite difficult, error-prone and unintuitive for non-formalists, such as the average software developer or quality assurance engineer. Further, as demonstrated in a study by Dwyer *et al.* [5] most validation requirements observed in practice, can be captured by properties in a limited number of temporal classes. A temporal class refers to a set of temporal logic formulas that share a common temporal structure and differ only in the propositional expressions within this temporal structure. For example, the LTL temporal formulas  $G(a \wedge b)$  and  $G(\neg p_1)$  share the common temporal structure  $G(exp)$ , differing only in the value of expression  $exp$ .

**Composing Properties.** Motivated by the above arguments, our approach uses a set of temporal property templates, rather than a complete temporal logic, for specifying properties. For the purposes of this work, we used the three templates shown in Fig. 2. Here  $p, p_1$  and  $p_2$  are expressions which are evaluable in a given single state, based on the contents of that state. They are essentially assertions about the state of different DOM elements on the page or relationships between them, very similar to assertions used in

1. *Global Template*::  $G(p)$ : Globally  $p$  is true
2. *Screen Transition Template*::  $p_1, i \rightarrow p_2$ : After transiting from a state where  $p_1$  is true, with an input or a guidance-directive-driven input sequence which matches  $i$ , or with any input ( $i = \text{nil}$ ),  $p_2$  is always satisfied
3. *Precedence Template*::  $p_1 \rightarrow Pp_2$ : Need to reach a state where  $p_2$  is true, prior to reach a state where  $p_1$  is true

**Fig. 2.** Temporal property templates used

existing automatic web application system testing frameworks such as Selenium. Therefore, it is quite easy for developers in field to migrate to our methodology. Examples of such expressions include availability of a node with given type and attribute, and availability of text content which matches a given regular expression.

Our Global Template is essentially an assertion  $p$  that would be checked on each state of the STG. A counter-example to this would be a state  $S$  where  $p$  is false. The Screen Transition Template relates to a pair of neighboring states  $S_1, S_2$  and a user action  $i$  that caused the transition between them.  $i$  is a tuple of a user event (e.g., a click) and a unique identifier of the DOM element on which it was performed. This template is particularly useful to check the function of a specific widget, a very common testing scenario. A counter-example to this property would be a tuple  $(S_1, T, S_2)$ , where  $S_1$  is a state in which  $p_1$  is true,  $S_2$  is a state in which  $p_2$  is false, and  $T$  is a transition from  $S_1$  to  $S_2$  made by an input or a guidance-directive-driven input sequence which matches  $i$ . The last template, the Precedence Template, is intended to check more global relationships in the navigation structure of the web application, for example, that a `logout` event in a web application is always preceded by a `login` event. A counter-example for this template would be a sequence of transitions which starting from the initial state, reaches a state where  $p_1$  is true, without going through any state where  $p_2$  is true.

**Model Checking Algorithm.** Rather than using a general purpose model checking algorithm for one of the common temporal logics (e.g. CTL or LTL) we found it more efficient to implement the actual model checking itself through a set of state-traversal checkers, one for each template. It is easy to see that any property expressed using the templates in Fig. 2 can be checked using a single, linear-time traversal of the STG in the navigation model. This also gives a flexible and extensible model checker, to which more templates can be easily added in future or existing ones modified to fit practical situations.

## 5 Tool Implementation

We have implemented our proposed validation approach in a system, which is comprised of two components. One of the components of the system is an extended version of open-source web application crawler CRAWLJAX. The second is a custom model checker called GOLIATH, which supports checking of template-based properties on the navigation model generated by the crawler, as explained in Section 4.2.

**Model Generation.** Our principal extension to CRAWLJAX is an implementation of the guided crawling feature described in Section 4.1. Our extended crawler supports Java

APIs for instantiating an object encoding a guidance directive (as per Definition 11). The user can instantiate one or more such application specific guidance directives and add them to the driver for crawling a specific target web application.

Another significant extension is a modification to CRAWLJAX's behavior discovery mechanism, to ensure that all behavior reachable up to a specified crawling depth is included in the model, regardless of the order of firing user events. This was previously not the case.

**Model Validation.** Our model checker, GOLIATH is implemented in Ruby and accepts the crawled navigation model as well as a set of properties formulated in terms of the property templates of Fig 2, with the expressions (e.g.  $p$ ,  $p_1$  and  $p_2$  in Fig 2) specified as Ruby expressions. These expressions can refer to specific DOM elements using the standard DOM API. We use the Nokogiri 5 HTML parser in our implementation, to parse and interpret such references.

The following is a very simple instance of a supported expression, which is true if and only if there is some `<a/>` element with `id` attribute value `login`. Here `doc` refers to the document DOM object of the page.

```
doc.xpath('//a[@id="login"]').any?
```

A little more complex and useful expression instances can be constructed as the following sequence of expressions:

```
login_xpath = '//a[@id="login"]';
logout_xpath = '//a[@id="logout"]';
login_avail = doc.xpath(login_xpath);
logout_avail = doc.xpath(logout_xpath);
login_avail.any? != logout_avail.any?
```

The last expression (hereafter referred to as  $p_{not\_together}$ ) is *true* only in states where precisely one of `login/logout` button exists. Note that in Ruby, sequences of expressions can be evaluated as an expression which yields the value of the final expression. Thus, the property  $G(p_{not\_together})$ , composed using the Global Template, checks that there is no state in the extracted navigation model, where both the `login` and `logout` buttons simultaneously exist.

## 6 Case Study

The proposed method has been implemented and evaluated by applying them to applications including an industrial one as well as an open source well-known application. To assess the efficacy and utility of our approach and the corresponding implemented tool, we have conducted a number of case studies following guidelines from [6]. All experiments are performed on a workstation with Intel Xeon CPU W5590@3.33GHz.

---

<sup>5</sup> <http://nokogiri.org/>

## 6.1 Subject Web Applications

We conducted our case studies on three web applications. The first subject (ORGANIZER) is a schedule organizer application, called **MyOrganizer**, taken from a textbook [14] on AJAX-based web application development. It is composed of 8,004 lines of Java code, 2,885 lines of JavaScript code, and 1,137 lines of JSP code. The second web application (BPM) is a commercial business process manager comprised of 58,701 lines of Java code, 61,541 lines of JavaScript code, and 90,742 lines of JSP code. It uses the YUI<sup>6</sup> AJAX library. The last subject (REDMINE) is the free and open-source, project management and bug-tracking tool **Redmine** (v1.2.1). It is composed of 66,778 lines of Ruby code, 18,402 lines of JavaScript code, and 5,751 lines of RHTML code. It is implemented based on the Ruby on Rails<sup>7</sup> framework.

## 6.2 Model Generation

Appropriate guidance directives, which are the keys for meaningful and relatively quick model generations, are given to generate the corresponding screen transition diagrams for the three test subjects. The directives include operations for logging in by providing usernames and passwords whenever there is a screen image having login prompts, or operations for creation of data entries such as bug reports in Redmine at appropriate stages during crawling.

Model reduction techniques are used with all of the applications to allow generation of models in reasonable time. Namely, elements to be clicked during fully automatic crawling are specified based on validation requirements of each application. In addition, for ORGANIZER, we have employed a state abstraction technique to ignore changes on element attribute from `mouseover` events on elements.

The server-tiers of the applications are also controlled to restore their state every time the crawler goes back to the initial page to achieve extraction of models with higher accuracy. For ORGANIZER, a hook to ensure the availability of the user account to be used during model generation is used. For REDMINE, a hook to revert backend database to the state with one normal user created from its initial installation is employed.

Table 1 shows the model (screen transition diagram) generation configurations and results for the examples. Please note loops are excluded on obtaining #path and maximum depth. Although we also tried to generate tests using the approach shown in [10], it did not finish generations for neither of the examples, simply because there are so many cases from its exhaustive analysis even with the depth limit.

## 6.3 Model Checking

We prepared multiple properties for each of benchmarks. Table 2 contains number of properties for the benchmarks categorized by their temporal templates and verification results, *i.e.* satisfied or unsatisfied. The table also contains maximum / average / minimum *check count* observed during model checking. *Check counts* are provided with the definitions shown in Fig. 3 for each of temporal property templates.

<sup>6</sup> <http://developer.yahoo.com/yui/>

<sup>7</sup> <http://rubyonrails.org>

**Table 1.** Model generation configurations and results

	Crawling depth	#Guidance direc.	#State	#Transition	#Path	Max. depth	Avg. depth	Time (min.)
ORGANIZER	11	1	38	232	65	13	6.385	133
BPM	8	1	765	4039	830	41	22.161	2769
REDMINE	11	7	1580	2528	1220	15	12.937	2634

**Table 2.** Model checking configurations and results

	$G(p)$			$p_1, i \rightarrow p_2$			$p_1 \rightarrow Pp_2$		
	Sat. prop.	Unsat. prop.	#Check	Sat. prop.	Unsat. prop.	#Check max/avg/min.	Sat. prop.	Unsat. prop.	#Check max/avg/min
ORGANIZER	0	0	0	0	7	23/20.4/5	0	0	0
BPM	9	0	765	2	0	113/61.5/10	2	0	13504317615/ 6752159070/525
REDMINE	2	1	1580	3	2	4/2.2/1	10	0	294/154.4/2

1.  $G(p)$ : Number of states within the model
2.  $p_1, i \rightarrow p_2$ : Number of transitions made by inputs or guidance-directive-driven input sequences which matches  $i$ , from states which make  $p_1$  true.
3.  $p_1 \rightarrow Pp_2$ : Number of transition sequence to states which make  $p_1$  true from states which makes  $p_2$  true

**Fig. 3.** Check count for the property templates

All 7 properties for ORGANIZER including Prop. 1a shown in Fig. 4, which are all in form of  $p_1, i \rightarrow p_2$  are violated and yielded counter examples. Playing back input sequences to reach and activate the counter examples, actually results in transitions which do not meet the properties. Execution of the counter examples for 5 of the properties generate an error message dialog in client-side web browser and a record indicating “Null Pointer Exception” in the log of Java server software. On execution of counter example for another property, state transition on a user input does not occur. Execution of counter example for the other property results in a state which does not satisfy post condition  $p_2$ . They are real bugs in a program shown in the textbook.

**Prop. 1a.** `doc.xpath('//img[@id="dayAtAGlance"]').any? , //img[@id="dayAtAGlance"]:onclick -> doc.xpath('//img[@src="img/head_dayAtAGlance.gif"]').any?`

**Prop. 3a.** `G(["home", "projects", "help"].map { |c| doc.xpath('//a[@class="' + c + '"]').any? }.all? )`

**Prop. 3b.** `doc.xpath('//input[@value="Create"]').any? &&  
doc.xpath('//input[@id="issue_subject" and @value and (0<string-length(@value))']).any? &&  
doc.xpath('//input[@id="issue_parent_issue_id" and @value="10"]').any? ,  
//input[@value="Create"]:onclick ->  
doc.xpath('//a[contains(@class,"issue") and contains(text(),"10")']).any? ||  
doc.xpath('//div[@class="errorExplanation"]').any?`

**Fig. 4.** Properties used in the experiments.

All 13 properties for BPM are satisfied. *Check counts* while model checking properties in temporal form of  $p_1 \rightarrow Pp_2$ , are large due to large number of traces to navigate

back to states which make  $p_2$  true from states which make  $p_1$  true. It is impossible to track through all of the traces in test-based approaches.

In REDMINE, Prop. 3a shown in Fig. 4 which is in the form of  $G(p)$  is violated. The property is violated in a state where the web browser is showing a file in a special format. 2 properties in the form of  $p_1, i \rightarrow p_2$  are also violated in REDMINE. The violated properties including the one shown as Prop 3b, require the application to transit to a DOM page containing a reference to a ticket (bug report etc.) or to a page indicating an error, from a form with a reference to the ticket input by the user, on “Create” button click. However, the application proceeds without any error even if the referred ticket does not exist, and the resulting pages do not contain any reference to the ticket specified.

Each checking of properties in the experiments finished within 10 seconds. As you can see from the results, once the model is generated, model checking is relatively very quick, and various properties can be examined with reasonable time even for large examples.

## 7 Discussion

**Completeness.** Since our technique relies on the finite state navigation model extracted by the crawler, erroneous behaviors not included in this model cannot be exposed by the subsequent model checking step. Although we attempt to capture the largest possible set of relevant behaviors, through crawler enhancements like guided crawling, and by the judicious choice of clickable widgets and crawling depth, the model generation step is inherently incomplete and in practice limited by computation resources.

However, compared to prevailing industrial practices of using trace-by-trace testing based on manually written tests, our technique is able to automatically explore and test a significantly larger set of behaviors and thereby expose many more errors.

**Scope.** Like other black-box validation techniques, in order to detect errors, our technique requires erroneous behaviors to be propagated and observable at the user interface of the web application, i.e., on the client-tier content displayed on the web browser. In other words, the technique cannot *directly* detect problematic server-tier behaviors, such as perhaps those pertaining to security or performance aspects of behavior. However, it is an ideal fit for testing validating behavior, which typically manifests at the client-tier.

Further, compared to other black-box system testing techniques for web applications, namely those based on user-given test cases and assertions, our approach which makes use of more expressive user-given guidance directives and temporal properties can target extensive classes of behavior.

**Automation Level.** Our model generation technique require the user to specify a crawler configuration, which is composed of set of elements to be clicked during fully-automatic crawling, state abstraction configuration to ignore some part of DOM page, and a set of guidance directives to partially control the crawling behavior. Our model checking technique is supplied from the user with the properties to be checked. User inputs required for the validation is small, considering large number of execution traces covered with a single configuration, as observed in our case studies.

**Scalability.** Our case studies confirm that our technique is applicable to large real-world applications in real use. However, our experience also showed that the model generation (crawling) time far exceeds the model checking time. Most of time required for the crawling is due to the communication latency between the crawler and the target application. As the computation burden of the process is relatively small, parallelizing and distributing the crawling is an attractive option for reducing the crawling latency.

Model reduction techniques (for example those described in Section 19) are another option for pruning the state-space and hence the crawling time. We did employ these techniques in our case studies, but to a limited extent. Although more aggressive model reduction, based on the properties to be verified, could reduce the model generation time further, it needs to be implemented and confirmed through more case studies.

**Correctness.** Since our crawling only observes the client-tier of the web application, the state computed and recorded by it is actually an abstraction of the true system state (which should include the state of the server tiers as well). Thus, the STG computed by the crawler represents an over-approximation of the “true” possible set of traces of the web application. Thus, it is theoretically possible that an error trace produced by our model checking is not reproducible on the actual web application, i.e. a false positive.

However, all the counter examples reported in our case-study are confirmed to reproduce. Reproducibility of counter examples is expected to depend also on the degree of abstraction, which is expected to be useful to scalability of the technique.

**Threats to Validity.** We have discussed some of issues related to the external validity of our evaluation in the discussions above. The internal validity of our evaluation may depend on implementation of software tools used. We have minimized the chance by making use of test sets for the tools, which are completely separated from benchmarks used in the evaluation.

## 8 Conclusions and Future Work

We have proposed a new approach for the automated system testing of modern, dynamic web applications. Our method employs automatic crawling to extract a finite state navigation model of the web application behavior. The user authors a set of properties, expressing desired navigation behavior, using a simple and intuitive template-based specification language. These are then efficiently checked on the extracted model. Our experience with this approach, through several case studies, confirms that it both applicable to industrial strength applications, as well as superior to current industrial practice based on manual testing.

Future works include more verification trials with larger applications for more robust evaluations of the proposed techniques as well as their extensions.

## References

1. de Alfaro, L.: Model Checking the World Wide Web. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 337–349. Springer, Heidelberg (2001)

2. Andrews, A.A., Offutt, J., Alexander, R.T.: Testing Web Applications by Modeling with FSMs. *Software and Systems Modeling* 4, 326–345 (2005)
3. Benedikt, M., Freire, J., Godefroid, P.: VeriWeb: Automatically Testing Dynamic Web Sites. In: *Proceedings of 11th International World Wide Web Conference* (2002)
4. Clarke, E.M., Grumberg, O., Peled, D.A.: *Model Checking*. The MIT Press (1999)
5. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in Property Specifications for Finite-State Verification. In: *ICSE 1999: Proceedings of the 21st International Conference on Software Engineering*, pp. 411–420. ACM, New York (1999)
6. Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case Studies for Method and Tool Evaluation. *IEEE Softw.* 12(4), 52–62 (1995)
7. Marchetto, A., Ricca, F., Tonella, P.: A Case-Study Based Comparison of Web Testing Techniques Applied to AJAX Web Applications. *International Journal on Software Tools for Technology Transfer (STTT)* 10(6), 477–492 (2008)
8. Marchetto, A., Tonella, P., Ricca, F.: State-Based Testing of Ajax Web Applications. In: *ICST 2008: Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*, pp. 121–130. IEEE Computer Society, Washington, DC (2008)
9. Mesbah, A., Bozdag, E., Deursen, A.V.: Crawling AJAX by Inferring User Interface State Changes. In: *ICWE 2008: Proceedings of the 2008 Eighth International Conference on Web Engineering*, pp. 122–134. IEEE Computer Society, Washington, DC (2008)
10. Mesbah, A., Deursen, A.V.: Invariant-Based Automatic Testing of AJAX User Interfaces. In: *Proceedings of the 31st International Conference on Software Engineering, ICSE 2009 (May 2009)*
11. NuSMV, <http://nusmv.irst.itc.it/>
12. Ricca, F., Tonella, P.: Analysis and Testing of Web Applications. In: *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001*, pp. 25–34. IEEE Computer Society (2001)
13. Strecker, J., Memon, A.M.: *Testing Graphical User Interfaces*. In: *Encyclopedia of Information Science and Technology*, 2nd edn. IGI Global (2009)
14. Zammetti, F.: *Practical Ajax Projects with Java Technology*. Apress (2006)



**Part III**  
**Distributed Systems**

# Technologies for Autonomic Dependable Services Platform: Achievements and Future Challenges

Eila Ovaska<sup>1</sup>, Liliana Dobrica<sup>2</sup>, Anu Purhonen<sup>1</sup>, and Marko Jaakola<sup>1</sup>

<sup>1</sup> VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland

<sup>2</sup> University Politehnica of Bucharest, Spl. Independentei 313, 060042 Bucharest, Romania  
{eila.ovaska, anu.purhonen, marko.jaakola}@vtt.fi,  
liliana@aii.pub.ro

**Abstract.** A city is smart if it can provide ambient services for citizens and other end-users who have to tackle emergency situations, including small and wide scale accidents and incidents. These ambient services embody intelligence of autonomic systems based on heterogeneous execution platforms enhanced with services that provide mechanisms for self-adaptation of dependable applications. This paper aims to serve as a reference point and guide for researchers and developers interested in technologies of autonomic dependable service platforms from three viewpoints: i) architectural options, ii) ontology models for services, context and dependability, and iii) modeling methods and practices for achieving high quality service platforms and intelligent applications. Our findings are illustrated through a smart city experiment and a set of open research challenges that need to be carried out for achieving a generic solution for an autonomic dependable service platform.

**Keywords:** Self-adaptive, Service Architecture, Ontology, Dependability, Security, Context, Semantic, Modeling.

## 1 Introduction

Besides situation based information, smart cities also provide quality-critical services for their communities, i.e. high quality services for professionals such as firemen, medical service providers, police, etc. These workers are responsible for real-time reaction in emergency situations. An emergency situation – caused by an extensive accident - means that the environment immediately changes, and therefore, the type and amount of information required for decision making and actions are also changing. These changes affect on the use and availability of computing and communication resources and the applications and services they provide for problem solving.

Smart city environments heavily rely on a multitude of sensor networks, embedded systems and devices that produce a large amount of data to be analyzed and reacted on in the short run by the security and safety monitoring processes executed by critical information systems. Our focus is on heterogeneous systems that embody large scale sensor networks, embedded systems, mobile devices and enterprise systems. On the one hand, sensors interact with other nodes in various ways and communication may be periodic or ad-hoc over wired and/or wireless networks. Moreover, computers

differ in their architecture and computing resources, such as CPUs, operating systems, processing power, amount of memory, energy requirements, etc. For example, nodes that are tiny devices powered by a battery and featuring low-power wireless communication capability bring challenges for application developers. On the other hand, information systems bring tight quality requirements, which are to be fulfilled in any case, preventing human and economical damages, by reacting with adaptive and autonomous behavior to changing situations and disappearing resources.

The development of smart cities benefits from the results gained from worldwide sensor webs [3] and service oriented architectures, also applied to sensor webs [13]. However, smart city applications require more; the platform shall be able to adjust its behavior based on defined dependability requirements, users' goals, quality of services and quality of available resources. Thus, this adaptation requires intelligence that enables real-time identification, reasoning and proactive reaction on alerts.

The objective of this paper is to explore the existing technologies for developing a service platform that is able to make autonomously the needed corrective and preventive actions in abnormal situations, and thereby provide a dependable infrastructure upon which adaptive applications can easily be developed and deployed. In particular, we focus on how dependability of applications can be guaranteed in ad-hoc situations.

The main contributions of this paper are 1) the options of adaptive service architectures for autonomic dependable service platform, 2) the inventory of potential ontologies that could be exploited in the development of a dependable service platform, and 3) the approaches applicable in the development of quality critical and situation based smart city applications. In summary, self-adaptation is based on context-awareness, realized as situation based behavior that takes into account the functional and quality properties of the environment and system itself, and the needs of system's end-users.

The structure of the paper is as follows. Sections 2 and 3 explore existing self-adaptation technologies and introduce a selected set of ontologies for representing services, context and dependability. Section 4 discusses modeling methods and techniques applicable for modeling context and dependability. Section 5 describes our experiences. Section 6 summarizes the open challenges, and conclusions close the paper.

## 2 Self-adaptation Technologies

### 2.1 Self-\*ilities of Autonomic Systems

An autonomic system has six characteristics [43] also called self-\*ilities: (1) *Reflexivity*. The system must have knowledge of its capabilities, boundaries and interdependencies, and be aware of its possible configurations and their impact on particular quality requirements. (2) *Self-configuring*. The system provides increased responsiveness by adapting to dynamic changes occurred internally or in the external operating environment. (3) *Self-optimizing*. The system provides operational efficiency by tuning resources and balancing workload. (4) *Self-healing*. The system provides resiliency by discovering and preventing disruptions as well as recovering from malfunctions. (5) *Self-protecting*. The system secures its assets by anticipating, detecting and

protecting against attacks. (6) *Adapting*. The core of the system is a control loop - sensing, decision making, and acting. The adaptive mechanisms are typically inspired by work on machine learning, multi-agent systems, and control theory. Adaptive service platforms most often support self-configuring or self-optimization [33, 47]. However, recently self-healing has started to receive more attention [4, 10]. As the system is autonomic in its 'normal' operation, it should be able to survive failures and to adjust system's characteristics to altering loads and resources autonomously [7].

## 2.2 Adaptive Middleware Architectures

The quality management in service-oriented systems requires additional features of the service brokering. QoS Broker utilizes a centralized approach where requests from clients are handled by a QoS-aware broker that evaluates each request using a performance model [29]. In a quality assurance framework the brokers are created on-demand and all the service providers and consumers can have their own brokers [41]. AgFlow has a separate service composition manager that forms a composite service from proper sub-services, requested from the service broker [47].

When quality management is added to the system it may cause that all the services are required to be changed to support the new middleware. VOLARE [38] adds an adaptation middleware between Web services and the broker module that monitors the resources and context of a device, and adapts service requests accordingly. It also adapts the QoS levels advertised by service providers, to realistically reflect each provider's capabilities at any given moment. The adapted service descriptions and advertisements are syntactically identical to un-adapted versions, allowing interoperability with non-VOLARE nodes.

In the ubiquitous environments the time spent for quality management is critical. Consequently, the quality-aware service discovery can be divided into two levels [8]. First, the service provisioning level identifies the actual pool of concrete services that will be used to implement the component functionalities so that the user's end-to-end requirements are fulfilled and the service broker's utility function is maximized. At the second level the service selection determines, from a pool identified by the service provisioning, the actual concrete services which are bound to each incoming user request. The service provisioning operates at a slower pace than the service selection.

The MUSIC platform [42] supports self-adaptation in ubiquitous and service-oriented environments. It provides an adaptation planning framework for managing the frequent and unexpected changes in the execution context of mobile applications. The purpose of the adaptation planning framework is to evaluate the utility of alternative configurations in response to context changes, to select a feasible one for the current context and to adapt the application accordingly.

The control loop of quality management can be roughly decomposed into monitoring, analyzing, planning and execution [24]. *Monitoring* means collecting the data needed for QoS adaptation from the system under interest. For example, in [41] monitors are created at runtime to transparently intercept requests and responses between consumers and providers. *Analyzing* is the phase, where the collected data is combined to form proper QoS metrics, and possibly also predictions of future states are made. Predictions are used for finding out quality violations proactively [41, 38]. *Planning* is the phase, where the required action is selected and it can be an aggregate

of local and global level reasoning [31]. Local QoS requirements are filled by individual capabilities and global QoS requirements by the service composition. The optimization problem is solved using, for example, models [41, 9] or fuzzy logic [33].

*Execution* of decisions may include adaptations at two different levels [33]: 1) the resource management level, which performs application-neutral adaptation, and 2) the service management level, which is responsible of the application adaptation. The application adaptation can be about adjusting application components and configurations, or about selecting appropriate service providers for services. Sometimes it is not possible to find any service providers that fulfill the quality requirements set by the client. In that case negotiation is needed. In [8] the service provisioning level takes care of the negotiation with the service providers so that the actual service selection is made faster. In [41] the consumer broker negotiates with the provider brokers.

The existing adaptive middleware solutions seem to cover collectively all the main features required for a dependable services platform. For example, the MUSIC platform is able to take into account both context and quality issues and in addition, it is designed for ubiquitous and service-oriented environments. However, although it seems to support self-configuring and self-optimizing it is unclear how it would suit to an self-healing and self-protecting environment. The performance and dependability of the existing platforms themselves are not yet clear. Some of them have been designed for ubiquitous environments, but they have been applied only in restricted contexts. Especially, more information is needed about how these systems would be able to support the management of large amounts of data in a short time frame required by the critical services in the smart city environment.

### 2.3 Enhanced Intelligence

An intelligent system has a capability of learning and the goal of learning is to improve the system performance with respect to its environment. Machine learning paradigms are divided into three major areas [25]: (1) *Supervised learning*, where a teacher, knowing the correct input-output pairs, provides them to the system, which is learning. The system tries to emulate the teacher's behavior and also generalize. (2) *Unsupervised learning*, where's no teacher, and thereby no correct outputs exist in the learning process. (3) *Reinforcement learning*, where neither here are correct outputs known, but the system learns those by interacting with its environment – the mechanism is called rewarding. Supervised and unsupervised techniques have both training phases, although the unsupervised version has no labels – correct 'answers' of the training data – available [20]. A complete machine learning method includes steps of selecting a candidate model, and then estimating parameters for it. The estimation is done with a learning algorithm and available data. In practice, supervised learning utilizes often an error function, which should be naturally minimized. Unsupervised learning uses clustering; similarity of elements in the same cluster should be maximized, and similarity of elements in different clusters should be minimized.

In the formal model of the reinforcement learning, the system has a (discrete) state, which perceives either completely or partially, a group of actions possible in that state, and a reward which is received when a new state is entered. The system's behavior and knowledge of the environment are modeled with a function. Usually learning is not about maximization of direct reward belonging to the state transition,

but long-term performance. Reinforcement learning is applied to proactively adapting the platform for stress peaks caused by users, overwhelming data or increased attacks.

Prediction provides four kinds of improvements to self-adaptation [12]: prevents unnecessary self-adaptation, reduces disruption from incremental adaptation, enables pre-adaptation to seasonal behavior, and improves overall choice of adaptation. Smart city applications would benefit from proactive capabilities, for example, with using the sensor information for discovering activity patterns that might lead to emergency situations. In that way, it could be possible to act on the situations before they occur and possibly prevent them. The similar approach is common in smart-home systems. For example, CASAS [40] is an adaptive smart-home system that utilizes machine learning and data-mining techniques in order to detect activity patterns, generate automation policies for those patterns, and also adapt to the changes in those patterns.

Learning capabilities can be also used for self-protection. For example, an approach for wireless anomaly based intrusion detection and response system uses learning for detecting complex malicious attacks [16]. Training sets are used by the system to generate rules for the behavior to be considered normal. Those rules are used during runtime to detect complex wireless attacks and generate counter measures to protect one or more wireless resources and the privacy of their users. Fast recovery without human intervention requires proper policy management mechanisms and automated ways to learn and derive policies [18]. Unlike the current self-healing systems that most often diagnose and heal failures after they have occurred rather than anticipating failures, in consequence-oriented diagnosis and recovery the host predicts or diagnoses the possible consequences from the symptoms [14].

## 3 Ontologies

### 3.1 Service Ontologies

Ontologies are used to represent knowledge in a uniform way that machines are able to process. Ontologies provide knowledge for describing the required and provided capabilities of a service, ability and rights of achieving a service, and the quality guaranteed for a service. The eXtensible Markup Language (XML), Resource Definition Framework (RDF) [19], and Web Ontology Language (OWL) [27] schema provide a basis for service description languages and ontologies, such as Web Ontology Language for Services (OWL-S) [5], Web Service Modeling Ontology (WSMO) [45], and Internet Reasoning Service (IRS) [32], which in turn provide building blocks for service semantics. The above mentioned service ontologies describe functionality of services. Another set of ontologies focuses on service context and quality properties. However, existing service ontologies focus mainly on Web services, and none of them provides complete support for service descriptions as required in adaptive service platforms. After analysis of the existing ontologies [21], a conclusion is that the existing service ontologies have to be enhanced.

In the software architecture field, the use of viewpoints is a community-wide accepted approach to cluster stakeholder-related concerns into a single view [53]. This principle can be lent to describing semantics of services. The use of multiple views is

a necessity; the interests of stakeholders differ, application domains differ, and service functionality and quality differ according to the usage and execution contexts. Moreover, different application domains, for example, information systems and pervasive computing applications, require modeling languages that take into account the characteristics of a domain by providing a notation that can be enhanced and adapted by domain specific extensions. The approach used in software product line engineering, namely the separation of commonality and variability [54], would be a viable approach that solves the problems in separation of common and domain specific semantics, and the integrated use of the defined service ontologies.

### 3.2 Context Ontologies

Context is defined in [15]: ‘Context is any information that can be used to characterize the situation of an entity.’ Understanding of context information is heavily improved during the last five years. Recently published articles, e.g. [6, 28, 22] indicate that knowledge on specification, modeling and usage of context information might be mature enough for realizing context-aware smart space applications. Typically, context information has three dimensions: physical, computational and user context [6]. In order to assist for achieving interoperability on the levels that concern context data and change of context, the context shall [39] i) have a complete domain coverage and terminology; ii) be expressive and without semantic ambiguity; iii) be processed without complexity; and iv) be evolvable.

Three types of context modeling and reasoning approaches [6] have been identified: an object-role based model, a spatial model and an ontology based model. The object-role based approach supports various stages of the software engineering process. Its weakness is a ‘flat’ information model, i.e. all context types are represented as atomic facts. The spatial context model is well suited for context-aware applications that are mainly location-based, like many mobile applications. The main consideration of the spatial context model is the choice of the underlying location model. Relational location models are easier to build up than geographic location models. SOUPA [11], as the only standardized context ontology, provides the most promising starting point for enhanced context ontology of smart cities. Therefore, an initial version of a context ontology introduced in [37] is an enhancement of SOUPA. This context ontology defines three levels; i) the physical context deals with raw context data gathered from the environment by sensors; ii) the digital context exploits physical context information and merges it with the system’s internal context information related to applications and information; and finally iii) the situation context clusters and abstracts the digital context information in a way that it matches to the application in hand and the preferences of its user. Thereafter, the situation context is used for adapting the application according to the view of the whole context information that relates to the application.

### 3.3 QoS and Dependability Ontologies

Quality of Service (QoS) has a traditional meaning as a property of communication technologies, including throughput, latency, jitter, error rate, availability, and network

security. In service oriented architectures, QoS is defined as dependability, maintainability, usability and scalability [35]. For end-users, QoS is the degree to which an executed service meets its quality requirements. Quality characteristics are often referred as non-functional requirements, although many of them (e.g. performance and dependability) are intertwined with functionality of software. Typically, existing quality ontologies have a specific focus. For example, a quality ontology may deal with one or few quality attribute(s) in defining, managing, or matching quality properties. However, to guarantee QoS requires comprehensive support for defining and managing all the relevant quality attributes of services, at design time and at run time.

There are several studies on QoS ontologies related to quality of Web Services. In [1], an overview of resilience knowledge base (RKB) is described, in which dependability and security ontology is derived from the taxonomies [2] and developed specifically for the RKB. The ontology is represented in OWL and incorporates 166 terms related to Dependability and Security, and 23 terms related to Systems. Moreover, there are QoS attribute ontologies and QoS-aware discovery solutions based on service level agreements [30]. Some papers also discuss performance, dependability and service cost as well as mechanisms of their aggregation [46, 26]. Other dependability-related metadata included into a description are i) the development metadata, i.e. information about service developers and implementation technology, and ii) the deployment metadata, i.e. information related to the hosting organization, location, deployment environment, network connection capacity, etc.. Adding this meta-information will allow clients to decide how to use services by decreasing common mode failures. Dynamic operational state parameters, such as current service load, CPU and memory usage, network loading, etc. might also be added to an extended description. Extending a service description with dependability metadata will bring us closer to a dependable semantic service platform.

## 4 Modeling Approaches

### 4.1 Service Modeling

Service modeling can apply ontology based service engineering, software engineering or/and domain engineering modeling techniques. Knowledge engineering applies ontologies for capturing and structuring topic knowledge shared across people, organizations, computers and software. Several methods for ontology development exist, e.g. METHONTOLOGY [17]. Also a set of languages, such as XML, RDF, and OWL can be applied to represent knowledge in a machine readable format. Moreover, OWL-S as a specific service description language can be used for describing service semantics.

Semantic Web, Ontology Engineering, Semantic Annotations, Semantic Search, Intelligent Services (Modeling, Discovery and Integration) are standards from W3C ([www.w3c.org](http://www.w3c.org)) and FIPA ([www.fipa.org](http://www.fipa.org)) for describing semantics models. The use of standards and open source tools (as W3C standards and OWL in Protégé 2000 environment) helps in sharing and using ontologies. Therefore, open standards and open source tools are the key enablers of semantics modeling. The advancement in



open source tools has greatly improved the ability to test and build ontologies from scratch or/and to reuse existing ontologies.

Application programming interfaces for ontology languages provide programming language dependent means to load ontologies, manipulate the ontology classes and relations, perform reasoning, and provide persistent storage for the model. Jena and OWLS API are the most popular Java frameworks for building semantic Web applications. These tools provide an application developer with a programming language for working with ontologies. Reasoning tools, such as FaCT++, Pellet, and RacerPro, provide a standardized XML interface to description logics systems. These tools help in ontology testing and in the development of application level intelligence based on ontologies described in OWL. Domain ontology specific editors such as OWLS Editor and WSMO design studio help in creating error free semantic descriptions based on a specific ontology.

Domain specific modeling addresses the specifics of an application domain in the meta-models from which a domain specific language is derived [23]. Although UML2 is a generic modeling language, it also provides constructs to extend the language with domain specific concepts. Thus, UML2 enhanced with domain specific ontologies that extends the language with service, context and dependability ontologies makes it applicable to the development of autonomic service platforms.

## 4.2 Context Modeling

Context-aware service engineering can be classified into two classes [22]: language based approaches and model-driven approaches. Language based approaches such as context-oriented programming and aspect-oriented programming follow the separation of concerns; applications are kept context-free and context is handled as a first-class entity of the programming language while separate constructs are used to inject context-related behavior into the adaptable skeleton of an application. Context-aware aspects programming is one step further; the aspects are driven by context, i.e. a particular aspect may or may not be executed depending on the context of use.

When trying to solve the complexity of context-aware applications, the approaches for context modeling and reasoning, namely object, spatial and ontology based have the following strengths and weaknesses: (1) The object-role based approach supports various stages of the software engineering process but has an information model not suitable for modeling context information of smart cities. (2) The spatial context modeling suits well for location based applications, like mobile applications. The drawback is the effort the special context model takes to gather and keep up to date the location data of the context information. Thus, this model is suitable for those smart city applications that do not have critical performance and dependability requirements as in emergency situations. (3) Ontological context models provide clear advantage both in terms of support for heterogeneity and interoperability. User-friendly graphical tools make the design of ontological context models viable to developers that are not particularly familiar with description logics. However, there is very little support for modeling temporal aspects in ontologies. The main problem might be that reasoning with OWL poses serious performance issues.

Programming based on the spatial context models [28] uses a small set of predefined types for composing context information. Thus, it is a topographical approach

for modeling a space, i.e. the context of actors is modeled as a geometric shape based on a sequence of coordinates. This enables actors to independently define and use potentially overlapping spatial context in a consistent manner, when relationships between spatial objects are defined implicitly, i.e. as the positions of the spatial objects shapes within the coordinate system. Thus, this programming model enables efficient integration of heterogeneous systems into a global smart space. Although the programming model might be a too sophisticated and overestimated approach for developing smart cities, it is a feasible enabler for self-organized sensor networks.

As mentioned, all approaches have weaknesses that make their use such as unfeasible in autonomic dependable service platforms. However, the generic model for context monitoring and situation based adaptation of application logic [15] is part of a viable solution, as described in [37].

### 4.3 Dependability Modeling

Dependability is considered from three viewpoints; as a system property; as a service capability, and a failure free operation. Dependability of a system is its ability to deliver a service that can justifiably be trusted. Dependability of a service is its behavior as it is perceived by the service user(s). Based on the definition of failure, an alternate definition of dependability exists, which complements the other definitions in providing a criterion for adjudicating whether the delivered service can be trusted or not: the ability of a system to avoid failures that are more frequent or more severe, and outage durations that are longer than is acceptable to the user(s). The first two definitions relate to the system and software design and implementation. The third one relates to the space's ability to survive under failures. Thus, it relates to self-healing and self-protecting, the characteristics of autonomic systems.

Dependability as a general concept manages four quality attributes: reliability, availability, security and safety. Safety is not common in smart cities but extremely important in safety-critical systems, e.g. in trains and airplanes. Thus, when using sensor information for making context-aware smart city applications, we focus on reliability, availability and security. Especially, our interest is on how to deal with these quality properties in a situation based manner and how to assure that quality requirements are met when ad-hoc situation based adaptations are made.

Survivability concerns autonomic systems and is a system's capability to fulfill its mission, in a timely manner, in the presence of attacks, failures or accidents. There are two aspects of survivability: protection and adaptation. Survival by protection refers to run-time security management. Survival by adaptation is an ability of a system to adapt its behavior to the changes that occur either in the system or externally in the operating environment and users' requirements [44]. Thus being self-adaptive and self-protecting, the dependable service platform should support survivability.

Security mechanisms like access control and encryption attempt to ensure survivability by protecting applications from harmful, accidental or malicious changes in the environment. Applications can also survive by adapting themselves to the changing conditions. Survival by adaptation typically involves monitoring and changing the quality goals so that they can be reached. In order to exploit architectural design knowledge for runtime adaptation, the following should be supported; a) identification of the internal and external contexts of the system, b) reasoning the change of

context, c) reasoning the activities to be taken in order to reach the quality goals, and finally, d) reconfiguring the system in a manageable way. In proactive adaptation, all these have to be made before they occur. Thus, appropriate learning techniques are used for predicting the behavior and making system survivable by proactive actions.

Dependability modeling has four main phases. First, the semantics of dependability is described at the design time by applying the quality-driven architecture design and quality analysis methodology [36]. As a result, the sub-attributes of dependability are described as separate ontologies for defining quality requirements. Second, quality requirements are mapped to the elements of software architecture models [34]. Third, the architecture is evaluated in order to detect whether required quality is met or not. Fourth, quality of the implemented software is measured and compared to requirements. In practice, the above described approach with supporting techniques and tools can be exploited but it needs enhanced middleware services that are able to use the design knowledge, represented in the service, context and dependability ontologies, in monitoring, reasoning and adapting dependability of smart city applications.

## 5 Smart City Experiment

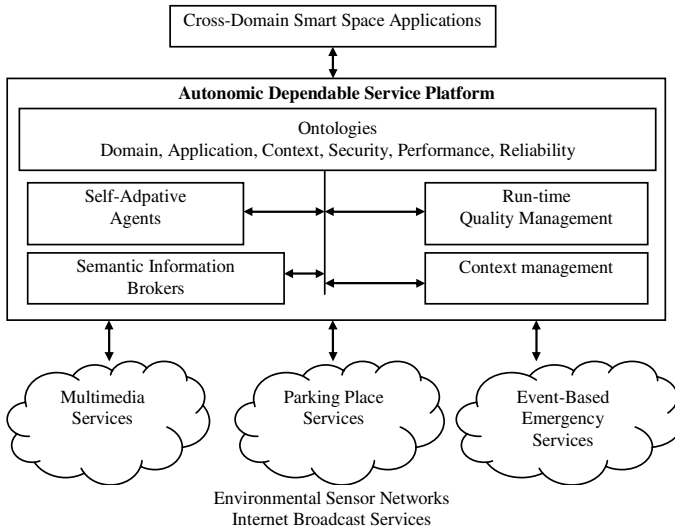
During the last year we have developed in the SOFIA/Artemis project a pilot that provides cross-domain services from multiple smart spaces. In that scenario, end-users use the heterogeneous information services provided by the smart office, smart home and smart city. The smart city provides the following services: i) multimedia entertainment services delivered from a cloud and personalized according to the end-user's profile, ii) navigation services for finding the optimum route to a target place in a city and iii) finding out a free parking place that is nearest a target place. The development of the smart city follows the design principles and practices particularly defined for smart space development [48].

### 5.1 Integration of Heterogeneous Platforms

Fig. 1 presents an overview how the autonomic dependable service platform was implemented and used for integrating existing solutions running on different platforms. On the one hand, environmental sensor networks are used for collecting data and video about the usage status and condition of roads and weather. On the other hand, the usage information of parking places is provided to smart applications through web-services and the semantic information interoperability platform (SIIP). Environmental changes are signalled to event subscribers via SIIP, too. Broadcast services and music delivered through the Internet are used as sources of multimedia services. Personalization of these content services is made by the open source content delivery platform that takes care of content management.

### 5.2 Semantic Information Interoperability Platform

The bottom part of the autonomic dependable service platform is based on the SIIP realized by Semantic Information Brokers (SIB) and agents that communicate via SIBs. SIB is essentially a blackboard that is used for publishing and subscribing



**Fig. 1.** Autonomic dependable service platform for multi-domain smart spaces

information. The smart space applications are composed of agents that produce and consume information. Information semantics is modelled with ontologies and stored into SIBs as subject – predicate – object RDF triplets. Because all kinds of information is delivered from producers to consumers via SIBs, a set of ontologies are required.

For the upper part of the SIIP we have developed several ontologies and applied them in our experiment. For example, the domain ontology includes a sensor ontology that has been applied to align heterogeneous sensor data collected from smart city and smart home environments. Metamodels are used for describing capabilities of multimedia content web services, such as news and entertainment services. However, our main focus has been on developing context, security and performance ontologies, and providing a set of autonomic agents that exploit these ontologies in order to achieve smartness. The context ontology exploits SOUPA and deals with physical, digital, situation, usage and social contexts [49]. Context monitoring, reasoning and adaptation are specific agents of the autonomous dependable service platform that are used for run-time quality management and making self-adaptive services for applications and the platform itself. Information security ontology covers threats, security goals, assets, countermeasures and metrics for managing information security at run-time [50]. Security ontology exploits reliability measures [51] as security base measures and therefore, dependability is covered by reliability and security sub-attributes. In the similar way, performance ontology deals with performance sub-characteristics and their related concepts, properties and metrics [52].

### 5.3 Dependability Metrics and Measuring Techniques

Sharing of quality information in an environment where the composition of information from heterogeneous devices and software in a dynamic manner requires a

common understanding of quality measures. In our experiment, the metrics ontology is used for defining the measures in an unambiguous way. The simplest measure is a base measure that is used to define more complex measures, i.e., derived measures, indicators and analysis models. This approach makes it possible to create and compose a new measure, i.e., analysis models, during runtime without a need to modify the existing application code.

#### 5.4 Proactive Adaptation

In order to continuously maintain an acceptable behaviour an application or a service has to be able to adapt before its operation. So far, the following experiments have been performed to validate that the SIIP is able to adapt smart city applications according to environmental changes or adapt its own behaviour to meet the requirements of the changed environment (i.e. according to the physical and digital context): (1) The navigator uses several smart city services for finding a proper parking place by automatically selecting the parking places information service with the fastest response time. (2) In case of traffic blocks caused by an accident or a roadwork, a new route is calculated and proposed to the end-user based on the information and reasoning agents running on top of the SIIP. (3) When a SIB starts to become short of memory, the application adapts and moves its operation to another SIB. (4) When context of the environment changes so that the required security level has to be increased, more secure mechanisms are taken into use in information transfer and sharing. (5) The energy consumption of a mobile device is monitored, and in case the energy level of the battery becomes low, some operations are closed and others moved to another device. (6) When behaviour of an agent becomes unacceptable, another agent with similar service capability is chosen to be used in the application. These experiments validate the automatic dependable service platform in the specific use cases.

## 6 Future Challenges

As described above there are several technologies and solutions for the development of autonomic dependable systems composed of heterogeneous subsystems such as sensors, networks, and enterprise systems. Our experiments about smart city applications provide some solutions to handle the heterogeneity of sensor technologies, proactive strategies for QoS adaptation and easiness to use and understand the sensor web and its applications. However, there are still open issues to be studied extensively in the ongoing and future research projects: (1) *Semantics*. A novel semantics modeling technique is required. The modeling approach should be i) goal oriented and stakeholder-centric, including viewpoints and views, ii) a set of core ontologies that cover the technology and application domains of smart city spaces, and iii) integrated orchestration of the developed ontologies at design time and at run-time. (2) *Proactive adaptation*. Novel reasoning techniques are required for measuring, monitoring, adapting and especially predicting of system's behavior from quality point of view. QoS-driven proactive adaptation requires innovative solutions for 1) managing semantic descriptions at run-time; 2) deriving quality indicators from basic QoS

measurements; and 3) enhanced adaptation and learning algorithms. Furthermore, the proactive QoS adaptation mechanism has to take into account the constraints of the selected technology; how to deal with context awareness and resource constraints of computing and communication environment. (3) *Dynamic semantic middleware*. There is a need for a generic solution that allows proactive service discovery, service composition and negotiation, and evolution management of cross-domain service platforms intended for heterogeneous networked systems, devices, actuators and appliances used for environmental monitoring. The biggest challenge might be the combination of run-time ontology evolution and dynamic behavior of smart environments that heavily exploit resource scarce embedded devices.

## 7 Conclusions

In this paper we explored the existing technologies applicable for use in the development of autonomic dependable service platforms that embody the technical challenges of pervasive computing environments, the business challenges of the multi-vendor product development and the quality of service challenges of trusted services that insist on a dependable and high efficient service platform.

We scoped our work by smart cities, the context where intelligence of services is benefited the most and where end-users should be supported with novel software and service engineering technologies. Moreover, we adopted an approach that exploits and enhances legacy systems because making running systems more intelligent and self-adaptive is a big enough challenge.

As a conclusion, we identified three topics that need extensive research and developments, namely i) semantics modeling, ii) proactive adaptation of architectures, and iii) middleware support for handling dynamism of self-organizing (ad-hoc) sensor networks. Our recent experiments have addressed these topics and we have identified two open issues required for smart cities: tiny and fast semantic information brokers and end-user programming tools for mashing-up of smart space applications.

**Acknowledgements.** This work is supported by the SOFIA/Artemis project, co-funded by EU, Tekes, and VTT and the Smash project, funded by VTT. The work of Liliana Dobrica was supported by Romanian Scientific Council CNCSIS – UEFISCSU, project number PNII – IDEI 1238/2008.

## References

1. Anderson, T., Andrews, Z.H., Fitzgerald, J.S., Randell, B., Glaser, H., Millard, I.C.: The ReSIST Resilience Knowledge Base. Technical Report. University of Newcastle upon Tyne (2007)
2. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. on Dependable and Secure Computing* 1(1), 11–33 (2004)
3. Balazinka, M., Deshpande, A., Flanklin, M.J., Gibbons, P.B., Gray, J., Nath, S., Hansen, M., Liebhold, M., Szalay, A., Tao, V.: Data management in the worldwide sensor web. *Pervasive Computing* 6(2), 30–40 (2007)

4. Baresi, L., Guinea, S., Pasquale, L.: Towards a unified framework for the monitoring and recovery of BPEL processes. In: TAV-WEB 2008 Workshop on Testing, Analysis and Verification of Web Services and Applications, pp. 15–19. ACM, New York (2008)
5. Barstow, A., Hendler, J., Skall, M.: OWL Web Ontology Language for Services, W3C (2004), <http://xml.coverpages.org/ni2004-01-08-a.html>
6. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Niclas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6(2), 161–180 (2010)
7. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC® Sensor Web Enablement: Overview and High Level Architecture. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) GSN 2006. LNCS, vol. 4540, pp. 175–190. Springer, Heidelberg (2008)
8. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: A Scalable Approach to QoS-Aware Self-adaption in Service-Oriented Architectures. In: Bartolini, N., Nikolettseas, S., Sinha, P., Cardellini, V., Mahanti, A. (eds.) Qshine/AAA-IDEA 2009. LNICST, vol. 22, pp. 431–447. Springer, Heidelberg (2009)
9. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: QoS-driven runtime adaptation of service oriented architectures. In: ESEC/FSE 2009, pp. 131–140. ACM, New York (2009)
10. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F., Mirandola, R.: Towards Self-adaptation for Dependable Service-Oriented Systems. In: de Lemos, R., Fabre, J.-C., Gacek, C., Gadducci, F., ter Beek, M. (eds.) Architecting Dependable Systems VI. LNCS, vol. 5835, pp. 24–48. Springer, Heidelberg (2009)
11. Chen, H., Finin, T., Joshi, A.: The SOUPA Ontology for Pervasive Computing. In: Ontologies for Agents: Theory and Experiences. Whitestein Series in Software Agent Technologies and Autonomic Computing, pp. 233–258. Springer, Heidelberg (2005)
12. Cheng, S.-W., Poladian, V., Garlan, D., Schmerl, B.: Improving Architecture-Based Self-Adaptation through Resource Prediction. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Self-Adaptive Systems. LNCS, vol. 5525, pp. 71–88. Springer, Heidelberg (2009)
13. Chu, X., Buyya, R.: Service oriented sensor web. In: Sensor Networks and Configuration, pp. 51–74. Springer, Heidelberg (2007)
14. Dai, Y., Xiang, Y., Zhang, G.: Self-healing and Hybrid Diagnosis in Cloud Computing. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 45–56. Springer, Heidelberg (2009)
15. Dey, A.K., Newberger, A.: Support for Context-Aware Intelligibility and Control. In: CHI 2009, pp. 859–868. ACM, New York (2009)
16. Fayssal, S., Alnashif, Y., Kim, B., Hariri, S.: A Proactive Wireless Self-Protection System. In: ICPS 2008, pp. 11–20. ACM, New York (2008)
17. Fernandez, M., Gomez-Perez, A., Juristo, N.: METHONTOLOGY: from ontological art towards ontological engineering. In: AAAI 1997 Spring Symposium Series on Ontological Engineering, Stanford, pp. 33–40 (1997)
18. Fuad, M.M.: Issues and Challenges of an Inductive learning Algorithm for Self-healing Applications. In: 7th Intl. Conf. on Information Technology: New Generations, ITNG 2010, pp. 264–269. IEEE Press, New York (2010)
19. Hayes, P.: RDF Semantics, W3C (2004), <http://www.w3.org/TR/rdf-schema/>
20. Jayaraj, A., Venkatesh, T., Murthy, C.S.R.: Loss classification in optical burst switching networks using machine learning techniques: improving the performance of TCP. *IEEE Journal on Selected Areas in Communications* 26(6), 45–54 (2008)

21. Kantorovitch, J., Niemelä, E.: Service Description Ontologies. In: Khosrow-Pour, M. (ed.) *Encyclopedia of Information Science and Technology*, 2nd edn., pp. 3445–3451. IGI Global (2008)
22. Kapitsaki, G., Prezerakos, G., Tselikas, N., Venieris, I.: Context-aware service engineering: A survey. *J. of Systems and Software* 83, 1285–1297 (2009)
23. Kelly, S., Tolvanen, J.: *Domain-Specific Modelling: Enabling Full Code Generation*. Wiley, New Jersey (2008)
24. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
25. Könönen, V.: Multiagent reinforcement learning in Markov games: asymmetric and symmetric approaches. Doctoral thesis, Helsinki University of Technology, Espoo, Finland (2004)
26. Lock, R., Dobson, G.: Developing an ontology for QoS. In: *Dependability interdisciplinary research Collaboration (Internal Annual Project Conference)*, Nesc (National e-Science centre), Edinburgh (2005)
27. McGuinness, D., van Harmelen, F.: *OWL Web Ontology Language Overview*, W3C (2004), <http://www.w3.org/TR/owl-features/>
28. Meier, R., Harrington, A., Beckmann, K., Cahill, V.: A framework for incremental construction of real global smart space applications. *Pervasive and Mobile Computing* 5, 350–368 (2009)
29. Menasce, D.A., Dubey, V.: Utility-based QoS brokering in service oriented architectures. In: *IEEE Intl Conf. on Web Services*, pp. 422–430. IEEE Press, New York (2007)
30. Menasce, D.A.: QoS Issue in Web Services. *IEEE Internet Computing* 6(6), 49–68 (2002)
31. Mokhtar, S.B., Georgantas, N., Issarny, V.: COCOA: CONversation-based service COMposition in pervASive computing environments with QoS support. *J. Systems and Software* 80(12), 1941–1955 (2007)
32. Motta, E., Domingue, J., Cabral, L., Gaspari, M.: IRS-II: A Framework and Infrastructure for Semantic Web Services. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) *ISWC 2003*. LNCS, vol. 2870, pp. 306–318. Springer, Heidelberg (2003)
33. Nahrstedt, K., Xu, D., Wichadakul, D., Li, B.: QoS-aware middleware for ubiquitous and heterogeneous environments. *IEEE Communications Magazine* 39(11), 140–148 (2001)
34. Niemelä, E., Evesti, A., Savolainen, P.: Modeling Quality Attribute Variability. In: *ENASE 2008*, pp. 169–176. INSTICC Press, Portugal (2008)
35. O'Brien, L., Merson, P., Bass, L.: Quality Attributes for Service-Oriented Architectures. In: *SDSOA 2007*, p. 3. IEEE Computer Society, Washington (2007)
36. Ovaska, E., Evesti, A., Henttonen, K., Palviainen, M., Aho, P.: Knowledge Based Quality Driven Architecture Design and Evaluation. *Information and Software Technologies* 52(6), 577–601 (2010)
37. Pantsar-Syvänieniemi, S., Simula, K., Ovaska, E.: Context-awareness in smart spaces. In: *IEEE Symp. on Computers and Comm.*, pp. 1023–1028. IEEE Press, New York (2010)
38. Papakos, P., Rosenblum, D.S., Mukhija, A., Capra, L.: VOLARE: Adaptive web service discovery middleware for mobile systems. *ECEASST* 19 (2009)
39. Preuveneers, D., Berbers, Y.: Internet of Things: A Context-Awareness Perspective. In: Yan, L., Zhang, Y., Ning, H. (eds.) *The Internet of Things: From RFID to the Next Generation Pervasive Networked Systems*, pp. 287–307. Auerbach Pub., New York (2008)
40. Rashidi, P., Cook, D.J.: Keeping the resident in the loop: adapting the smart home to the user. *IEEE Trans. on Systems, Man and Cybernetics* 39(5), 949–959 (2009)



41. Robinson, D., Kotonya, G.: A self-managing brokerage model for quality assurance in service-oriented systems. In: IEEE High Assurance Systems Eng. Symp., pp. 424–433. IEEE Press, New York (2008)
42. Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S., Lorenzo, J., Mamelli, A., Scholz, U.: MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) *Self-Adaptive Systems*. LNCS, vol. 5525, pp. 164–182. Springer, Heidelberg (2009)
43. Salehie, M., Tahvildari, L.: Self-Adaptive Software: Landscape and Research Challenges. *ACM Trans. on Autonomous and Adaptive Systems* 4(2), art. 14 (2009)
44. Tarvainen, P.: Adaptability evaluation of software architectures: a case study. In: IEEE Int. COMPSAC 2007. IEEE Computer Science, Washington (2007)
45. WSMO: WSMO studio (2004), <http://www.wsmostudio.org/>
46. Yang, S., Lan, B., Chung, J.Y.: Analyses of QoS Aware Web Services. In: Intl. Comp. Symp. on Web Technologies and Information Security, ICS (2006)
47. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. *IEEE Trans. Soft. Eng.* 30(5), 311–327 (2004)
48. Ovaska, E., Cinotti, T.S., Toninelli, A.: The design principles and practices of interoperable smart spaces. In: Liu, X., Li, Y. (eds.) *Advanced Design Approaches to Emerging Software Systems: Principles, Methodologies, and Tools*, pp. 18–47 (2012)
49. Pantsar-Syvänieniemi, S., Kuusijärvi, J., Ovaska, E.: Context-Awareness Micro-architecture for Smart Spaces. In: Riekkki, J., Ylianttila, M., Guo, M. (eds.) GPC 2011. LNCS, vol. 6646, pp. 148–157. Springer, Heidelberg (2011)
50. Evesti, A., Savola, R., Ovaska, E., Kuusijärvi, J.: The design, instantiation, and usage of information security measuring ontology. In: The Second International Conference on Models and Ontology-based Design of Protocols, Architectures and Services ( 2011)
51. Evesti, A., Ovaska, E.: Design time reliability predictions for supporting runtime security measuring and adaptation. In: The Third International Conference on Emerging Network Intelligence, EMERGING 2011, 6 pages. IARIA (2011)
52. Purhonen, A., Stenudd, S.: Runtime Performance Management of Information Broker-Based Adaptive Applications. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 203–206. Springer, Heidelberg (2011)
53. Dobrica, L.: Exploring Approaches of Integration Software Architecture Modeling with Quality Analysis Models. In: 2011 Ninth IEEE/IFIP Conference on Software Architecture, pp. 113–122. IEEE Computer Society, Los Alamitos (2011)
54. Dobrica, L., Ovaska, E.: Service Based Development of a Cross Domain Reference Architecture. In: Maciaszek, L.A., González-Pérez, C., Jablonski, S. (eds.) ENASE 2008/2009. CCIS, vol. 69, pp. 305–318. Springer, Heidelberg (2010)

**Part IV**  
**Data Management**

# Extracting the Main Content of Web Documents Based on Character Encoding and a Naive Smoothing Method

Hadi Mohammadzadeh<sup>1</sup>, Thomas Gottron<sup>2</sup>,  
Franz Schweiggert<sup>1</sup>, and Gholamreza Nakhaeizadeh<sup>3</sup>

<sup>1</sup> Institute of Applied Information Processing, University of Ulm, D-89069 Ulm, Germany

<sup>2</sup> Institute for Web Science and Technologies, Universität Koblenz-Landau,  
D-56070 Koblenz, Germany

<sup>3</sup> Institute of Statistics, Econometrics and Mathematical Finance, University of Karlsruhe,  
D-76128 Karlsruhe, Germany

{hadi.mohammadzadeh, franz.schweiggert}@uni-ulm.de,  
gottron@uni-koblenz.de, nakhaeizadeh@statistik.uni-karlsruhe.de

**Abstract.** This chapter presents R2L, DANA and DANAg, a family of novel algorithms for extracting the main content (MC) of web documents. The main concept behind R2L, which also provided the initial idea and motivation for the other two algorithms, is to exploit particularities of Right-to-Left languages for obtaining the MC of web pages. As the English character set and the Right-to-Left character set are encoded in different intervals of the Unicode character set, we can efficiently distinguish the Right-to-Left characters from the English ones in an HTML file. Afterwards, the R2L approach extracts areas of the HTML file with a high density of Right-to-Left characters and a low density characters from the English character set. Having recognized these areas, R2L separates only the Right-to-Left characters as a result. The first extension, DANA, improves effectiveness of the baseline algorithm by employing an HTML parser in a post processing phase of R2L for extracting the MC from areas with a high density of Right-to-Left characters. DANAg is the second extension and generalizes the idea of R2L to render it language independent.

**Keywords:** Main content extraction, R2L languages, Unicode character set, HTML documents, UTF-8 encoding form.

## 1 Introduction

Content extraction is the process of identifying the Main Content (MC) and/or removing the additional items, such as advertisements, navigation bars, design elements or legal disclaimers [5]. Estimates from 2005 [3] stated these additional items to account for 40 to 50% of the data on the World Wide Web, predicting this ratio to increase constantly in the future.

The rapid growth of text based information on the web and various applications making use of this data motivates the need for efficient and effective methods to identify and separate the main content from the additional content items. In particular, the identification of the MC is beneficial for web search engines: when crawling and indexing the web, knowing the actual main content of each web page can be exploited for

the purpose of determining more precise and descriptive index terms for the document. Furthermore, MCE is also applied in scenarios where a reduction of a document to its main content is of advantage, e.g. on devices that have limited storage or bandwidth capacity or underlie restrictions regarding the presentation of web documents [7], such as mobile phones, screen readers, etc. Furthermore, MCE can be considered as a pre-processing step for general text mining applications operating on the web. All these application scenarios have led to the development of several approaches and algorithms for main content extraction from HTML documents.

The most common traditional approach to MCE has been to hand-code rules, often implemented by regular expressions. These hand tailored rules achieve a high or even perfect accuracy on the web documents for which they have been designed. But, since different web sites have different layouts, maybe even in a variety of configurations and layouts frequently changing over time, hand coded rules are highly labour intensive and easily broken by changes to the structure of a web page [15]. This has led to an interest in finding solutions, which are generic (i.e. applicable to various types of web pages from different sites), accurate (i.e. able to extract all important content at a high precision) and efficient (i.e. capable of processing a large number of web pages at a high throughput rate) [9]. However, generic MCE approaches do not reach a perfect accuracy. Even state-of-the-art methods have been shown to still leave space for improvements [4]. From a technical point of view, most of the approaches [5][14] for MCE use HTML tags to separate the main content from the extraneous items. This implies the need to employ a parser for the entire HTML document. Consequently, the computation costs of these MCE approaches include an overhead for the parser.

In the early days of the World Wide Web, the content of the most web pages was written in English language. By now, and especially in the last decade, a large part of information is being published also in other languages, for example Spanish, German, and French, etc. Except for the non-English languages mentioned here, there are several other languages using Non-ASCII codes for their characters (Figure 1 gives an example of web pages with a Non-ASCII character set in which we have also highlighted the main content). The Unicode character set (UCS), which was introduced after ASCII and ISO-8859\*, reserves an exact interval for each language. Some of these intervals have no common character with the English character set.

The R2L [13] approach presented in this chapter exploits this fact to realize an MCE algorithm for Arabic, Farsi, Pashto, and Urdu languages. By working on the binary character encoding directly, we achieve an improvement in time performance. Moreover, our approach outperforms all other MCE algorithms also in extraction performance, i.e. detects the main content more accurately and reliably. This provided the motivation for the initial version, R2L, of our algorithms presented in this chapter. With its extensions DANA [12] and DANAg [11], we further enhance and generalize the original idea towards a better performance and a language-independent version.

Altogether, in this chapter we make three main contributions:

- We develop the idea of using character encoding for developing R2L, a new approach for MCE.



Fig. 1. A web page with an outlined main content

- We extend the R2L approach to the algorithms DANA and DANAg to further improve the extraction accuracy and develop a language independent version of the method.
- We analyse our approaches under the aspects of efficiency and effectiveness. We compare them to eleven established MCE algorithms [7,10,2,16,5,14] and show that we extend the state-of-the-art in terms of both, efficiency and effectiveness.

The remainder of the chapter is organized as follows: After discussing related work in Section 2, Section 3 introduces R2L languages and elaborates UCS and the UTF-8 encoding form. The main part of this chapter is explained in Section 4 where we introduce our algorithms in detail. Section 5 reports the empiric evaluation of R2L, DANA and DANAg as well as the comparison to other algorithms under the aspects of efficiency and effectiveness. Finally, concluding remarks are given in Section 6.

## 2 Related Work

In the last decade, many scientists and researchers have been working on main content extraction. Several algorithms have been introduced and many papers in this field have been published. All these main content extraction algorithms can be categorized into the two following groups:

## 2.1 Methods Based on the DOM Tree

One of the more prominent solutions for MCE is the Crunch framework [7] of Gupta et al. This framework applies an HTML parser to construct a DOM tree from an HTML document. Then, by navigating the DOM tree recursively, rather than using the raw HTML markup, and utilizing a number of heuristic filtering techniques, the main content of HTML web pages is extracted.

Mantratzis et al. proposed a new algorithm in [10] whose function was based on the DOM tree as well. This algorithm determines the areas with a high hyperlink density within a web document, so it can separate these areas from the main content in web pages. In doing this, they examined the DOM tree and assigned specific scores to each section based on the amount and relative location of hyperlink nodes in the DOM tree.

Debnath et al. [1] introduced the FeatureExtractor algorithm and its extension the K-FeatureExtractor. These two algorithms identify the “primary content blocks” based on several features. First, they segment the web pages into web page blocks and, second, they separate the primary content blocks from the non-informative content blocks based on their compliance with desired features, such as dominance of text or images.

## 2.2 Methods Based on HTML Source Code Elements

The work of Finn et al. [2] described the process of extracting and classifying information from HTML documents for the purpose of integrating it into digital libraries. They proposed the “Body Text Extraction” (BTE) approach, which identifies a single continuous fragment of the HTML document containing the MC. The fragment is chosen to contain a high percentage of text against low percentage of tags. The choice is formulated as an optimization problem and is based on tokenizing a web document into a binary vector of word and tag elements.

Pinto et al. [16] introduced the Document Slope Curves (DSC) method, which is an extended model of the BTE and is based on the same binary vector. In an intermediary step, DSC generates a graph by plotting the accumulated tag token count for each entry in the vector. Then, the approach extracts long and low sloping regions of this graph represent the main content (text without HTML tags). By employing a windowing technique, the approach can identify also a main content which is fragmented into several parts of an HTML document.

Gottron [5] presented two new algorithms: the Content Code Blurring (CCB) and the Adapted Content Code Blurring (ACCB) are capable of working either on characters or tokens. CCB finds the regions in an HTML document which contains mainly content and little code. In order to do this, the algorithm, determines a ratio of content to code for each single element in the content code vector (CCV) in the vicinity of each element and, by using a Gaussian blurring filter, builds a new vector, referred to as Content Code Ratio (CCR). Now a region with high CCR values indicates the main content. In ACCB, all anchor-tags are ignored during the creation of the CCV. Two parameters influence the behaviour of these two algorithms, so tuning these two parameters is important in order to produce good results [6].

Moreno et al. [14] introduced a language independent algorithm, called Density, (tested on English, Italian and German languages) for the main content extraction.

This approach, similar to CCB, has two phases. In the first step, they separate texts from the HTML tags by using an HTML parser; afterwards, the extracted texts are saved in an array of strings  $L$ . In the second step, a region in the array  $L$  that has the highest density will be determined as a main content. In addition to finding the highest density area in the array  $L$ , two parameters influence the behaviour of the algorithm. The first parameter,  $C1$ , determines minimum required length for texts in each element of the array  $L$  to be selected and inserted to the new array of String  $R$ , which is considered to keep the high density region of text. The second parameter,  $C2$ , specifies the acceptable distance between lines in  $R$  and the lines which want to be added to  $R$ .

Weninger et al. introduced content extraction via tag ratios, called CETR, in [17]. This method extracts the main content from web pages by using the HTML document's tag ratio. Their method computes tag ratios on a line-by-line basis and, afterwards, produces a histogram based on results. Finally by using the k-Means clustering method, they cluster the resulting histogram into the content and the non-content area.

### 3 R2L Languages, Unicode, and UTF-8 Encoding Form

There are different directions in writing. Languages such as English, Spanish, and German are written from left to right, while languages such as Arabic, Farsi, Pashto, and Urdu are written from right to left. Our approaches R2L and DANA are able to extract the main content of right to left language web pages. Therefore, in this section we explain some characteristics of these languages as well as their representation and encoding.

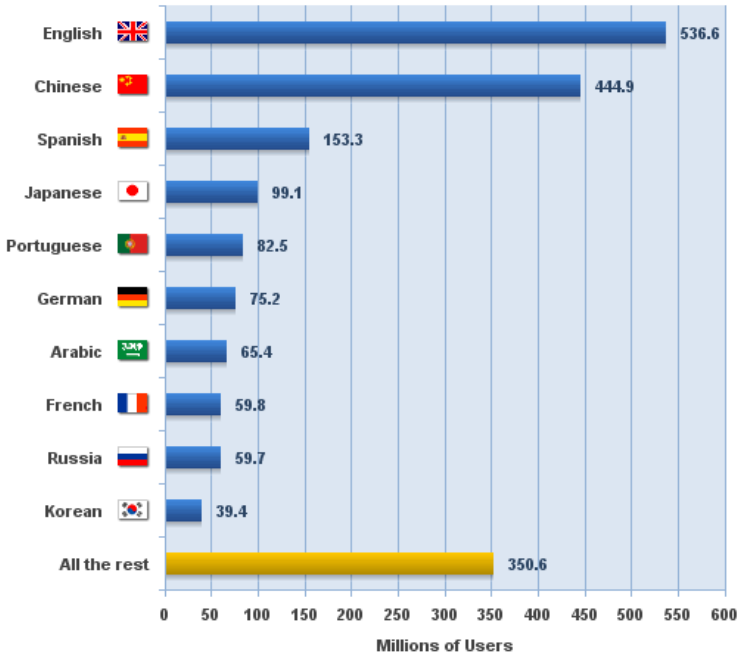
#### 3.1 Languages on the Web

Figure 2 provides statistics about the top 10 languages of Internet users<sup>1</sup>. Following English, Chinese, Spanish, Japanese, Portuguese and German speaking, users speaking Arabian – one of the four languages discussed in this chapter – rank at position 7 of this list. Concerning the statistics from the Internet World Statistics (IWS), in 2011 more than 33.5% of people in Arabian countries have access to the Internet. By exploring these statistics further, we can see around 3.4% of all users in the world coming from Arabian countries. By comparing this value with older statistics from 2000 where only 0.8% of all users in the world was from Arabian countries, we see a high growing rate of Internet use in Arabian countries. It is important to know that the population of Arabian countries is more than 350 million people and in the future, even more people in these countries will have access to the Internet. Therefore, the numbers of visitors of Arabian web pages are going to increase. This considerations motivate the importance of doing MCE research on Arabian web sites.

#### 3.2 Unicode Character Set

Before the Unicode Character Set was introduced, ASCII (developed to ISO 8859\*) and EBCDIC were used on computers. Thereby, only one byte was allocated for storing a

<sup>1</sup> <http://www.internetworldstats.com/>



**Fig. 2.** Top 10 languages on the Internet in millions of users in 2010

single character; consequently only 256 characters could be coded. By considering this limitation, rows in the interval [128, 255] in the encoding table were used by different characters of different languages. Since the introduction of UCS, where only one special number was mapped to each character, we are able to use all characters of different languages on computers. At first, from 1991-1995, only 16 bits were reserved for each character, but when the new version of UCS was introduced (July 1996), it was possible to save a character in 21 bits. The newly defined UCS encoded all characters in the interval [U+0000, U+10FFFF]. There are several encoding forms in UCS, such as UTF-8, UTF-16, and UTF-32. In each of these encoding forms, respectively, one character can be saved in one to maximally four bytes, one or two words, or 32 bits.

### 3.3 UTF-8 Encoding Form

As we mentioned in [3.2](#), UTF-8 is a variable-length encoding form in UCS. This encoding form can code all characters in UCS and represents each character in one to four bytes and has two following special characteristics (Keep in your mind that characters which are used in non-English languages need two, three, or four bytes):

- It reserves the same character codes from ASCII that makes UTF-8 backward-compatible with ASCII. Hence, every valid ASCII character (a 7-bit character set) is a valid UTF-8 character sequence and is mapped onto the following scheme. Each of these characters has a value of less than 128.



Bits	Last code point	Byte1
7	U+007F	0xxxxxxx

- It is capable of encoding up to  $2^{31}$  characters and uses the scheme in Table 1 to handle code points with up to 31 bits. Some features of these scheme are: 1) For every UTF-8 byte sequence, the first byte determines the length of the sequence in bytes. 2) The rest bytes have 10 as their two most-significant bits (bits 7 and 6), so it can be recognized whether or not a byte is the first byte in a longer byte sequence.

**Table 1.** Scheme of byte sequence in UTF-8

Rows	Bits	Last code point	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
1	7	U+007F	0xxxxxxx					
2	11	U+07FF	110xxxxx	10xxxxxx				
3	16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
4	21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
5	26	U+3FFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
6	31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

All letters of non-English-languages which we will discuss in this chapter take exactly 2 bytes, for example the Arabian character set has been represented in the interval [U+0600, U+06FF], and follow the byte sequence in row 2 in Table 1. In UTF-8, each character which needs more than one byte will be coded in such a manner that each byte of this character is greater than 127 and so it can be distinguished from one-byte characters with value less than 128, see Table 1.

To illustrate this, consider the following example. The letter ب in Arabian language (corresponding to the letter *b* of the Latin alphabet) has been defined with the value 0x0628 (with the equivalent binary value of 0B0000110000101000). According to the second row of Table 1, this value should be divided into three parts:

$$\boxed{000} \boxed{011000} \boxed{0101000}$$

Now, two right parts will be added to the bytes corresponding to the bytes in row 2 in Table 1, respectively :

$$\boxed{11000000} \boxed{10000000}$$

The result is:

$$\boxed{11011000} \boxed{10101000}$$

Note these two byte values are greater than 127. Therefore, we can easily separate one-byte characters with value less than 128 from double-byte characters by considering the first bit.

### 4 Algorithms: R2L, DANA, and DANAg

Based on the observations regarding character encoding in Section 3 we now develop our R2L [13] algorithm in Section 4.1. We then extend R2L to DANA [12] in Section 4.2 to improve its effectiveness and to a language independent version DANAg [11] in Section 4.3.

## 4.1 Algorithm R2L

The process underlying R2L can be subdivided into a preprocessing step and four main phases. The individual steps in this process work as follows.

### 4.1.1 Preprocessing Step

In the preprocessing step, all JavaScript and CSS codes and comments are removed from the HTML file. There are two reasons for this: (a) they do not directly contribute to the main text content and (b) they do not necessarily affect the content of the HTML document at the same position where they are located in the source code. Especially this latter incoherence between presentation and technical realisation in the source code could introduce inconsistencies in the downstream analysis<sup>2</sup>.

### 4.1.2 First Phase: Character Set Separation

In the following, we define two sets,  $S_1$  and  $S_2$  we will use throughout this section:

$$S_1 = \{\text{All characters belonging to UCS R2L languages}\}$$

$$S_2 = \{\text{All first 128 characters of UCS}\}$$

We know that the English characters, which are used in HTML tags, have values less than 128 and therefore can be classified to  $S_2$ . All characters of R2L languages use two bytes with a value greater than 127, and therefore they are classified to  $S_1$ . This simple rule helps us to efficiently separate R2L language characters from the first 128 characters of UCS.

In this first phase, the algorithm reads an HTML file as a stream of bytes and then by using the above rule it distinguishes whether the generated byte is a member of  $S_1$  or  $S_2$ . Now, the characters in each line<sup>3</sup> of the file are separated into two parts: characters that are a member of  $S_1$ , and the ones that are a member of  $S_2$ . By regarding to this condition, we are able to count the number of characters belonging to  $S_1$  and  $S_2$  for each line of an HTML file, which is stored in two one-dimensional arrays  $T_1$  and  $T_2$ , respectively.

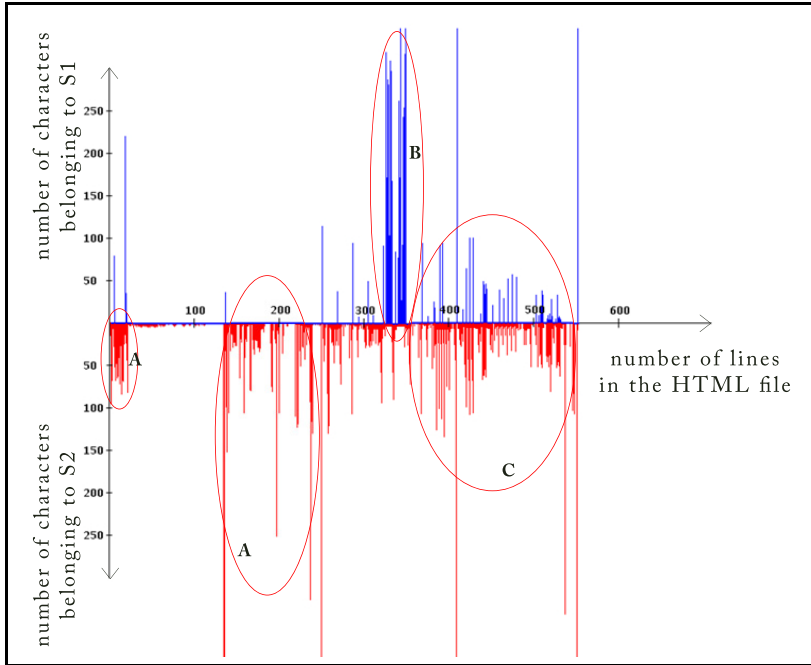
### 4.1.3 Second Phase: Smoothing

After storing the number of R2L and English characters for each line of the HTML file in the two arrays  $T_1$  and  $T_2$ , we want to recognize areas in the HTML file, where the density of R2L and English characters is high and low, respectively.

To illustrate our approach, we depict two diagrams. In Figure 3, we draw two groups of columns above and below the x-axis, with the length equal to the number of R2L and English characters, as stored in  $T_1$  and  $T_2$ , for each line of the HTML file. For example, suppose that the  $i$ -th line of an HTML file has  $y_1$  R2L and  $y_2$  English characters. Then, two lines with the length equal to  $y_1$  and  $y_2$  are drawn above and below the x-axis. Our hypothesis is that the main content is typically located above the x-axis. In Figure 3,

<sup>2</sup> This effect has already been observed in related work [5].

<sup>3</sup> We apply a preprocessing step to normalize line length and, thereby, render the approach independent from the actual line format of the source code.

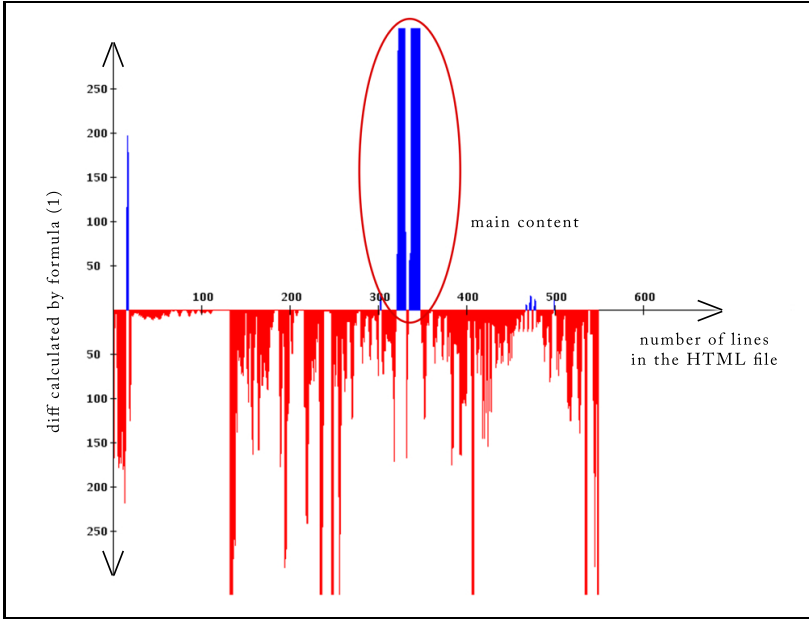


**Fig. 3.** An example plot shows the density of the main content and extraneous items

the measurement unit for the x-axis is the number of lines in the HTML file and the measurement unit for the y-axis toward up and down is the number of R2L and English characters, respectively, of each line of an HTML file. Here we interpret Figure 3 to find MC of an HTML file. There are three types of regions:

- Regions that have a low or near zero density of columns above the x-axis while having a high density of columns below the x-axis. We observed that these regions typically consist mainly of HTML tags. We outline examples for such areas with A in Figure 3.
- Further, we see some regions which have a high density of columns above the x-axis and low density of columns below the x-axis, one of them is marked with B in Figure 3. The main content, typically, will be found areas like this. In other words, some of these areas comprise the main content.
- There are some regions that have a medium density of columns above and below the x-axis. These regions form parts of navigation menus, panels, or other related link lists. Here, normally, the density of the columns below the x-axis is somehow more than the density of the columns above the x-axis because in HTML files we need to write many tags to make menus or extraneous items. One of these areas is outlined in Figure 3 and labelled with C.

Now the problem of finding MC in the HTML web pages becomes the problem of finding regions such as region B in Figure 3 comprising the main content. In the next



**Fig. 4.** Smoothed version of Figure 3 in which the MC can be identified more easily

two steps and in phase three (Section 4.1.4), we apply an elegant and simple method for finding regions such as B containing the main content in an HTML file:

- For all columns  $i$  we calculate  $diff_i$  by using the following formula 1. In this formula,  $T1_i$  and  $T2_i$  are the number of R2L and English characters of line  $i$  in an HTML file.

$$diff_i = (T1_i - T2_i) + (T1_{i+1} - T2_{i+1}) + (T1_{i-1} - T2_{i-1}) \quad (1)$$

This produces a smoothed plot as can be seen in Figure 4. Here again, if  $diff_i > 0$  we draw a line with length  $diff_i$  above the x-axis. Otherwise, we draw a line with length  $|diff_i|$  below the x-axis. Unlike Figure 3, a large part of menus and additional news in Figure 4 have been hidden.

- Now in Figure 4, we identify all regions above the x-axis and, for simplicity, we define a new set  $R = \{r_1, r_2, \dots, r_n\}$  of all such regions. Each element  $r_j \in R$  denotes only one individual line or a range of lines covering one region and  $n$  is the total number of recognized regions above the x-axis. In our example, there are several regions, one near the y-axis, two regions in the middle of the x-axis, and finally some small regions in the interval [450, 500] of the x-axis. In addition, we count the number of characters for each region and also we specify the position of regions in Cartesian coordinate. The strong hypothesis underlying R2L is that among all regions, the region with the maximum number of characters definitely belongs to the main content.

#### 4.1.4 Third Phase: Recognizing the Boundary of the Main Content Area

In this phase, all regions shaping the main content are discovered. Concerning the set  $R = \{r_1, r_2, \dots, r_n\}$  defined at the end of the previous phase, we have two possible outcomes: (1) we discovered only one region, i.e.  $n = 1$ , or (2) we have discovered several regions, so  $n > 1$ . If  $n = 1$ , then  $r_1$  is the only main content region and we are done. Otherwise, if  $n > 1$ , we start by finding the region  $r_m \in R$  which contains the highest number of characters. Again, we define a new empty set  $T$ , intended to denote the set of all regions comprising the main content at the end of this phase, and add  $r_m$  to this set  $T$ . For finding all other regions of the main content, we use Algorithm 1. In this algorithm,  $d(r_i, r_j)$  returns the distance between two regions  $r_i$  and  $r_j$  and the parameter *gap* determines the maximum allowed distance between two sequential regions of main content. In the Algorithm 1, the first loop discovers all regions in the left side of  $r_m$  comprising the main content. For example in the first iteration of *while*, Algorithm 1 evaluates the distance between sequential regions  $r_m$  and  $r_{m-1}$  and if this distance is less than or equal *gap* then  $r_{m-1}$  is added to the set  $T$ . Otherwise, the *while* loop is terminated. Therefore, the loop *while* will be terminated immediately as soon as the distance between two consecutive regions becomes greater than *gap*. In the same way, the second loop distinguishes all regions to the right side of  $r_m$  comprising the main content and adds the valid regions to the set  $T$ . The result and output of Algorithm 1 is the set  $T$  comprising all regions making the main content of selected web page. It is clear that  $T$  is a subset of  $R$ .

---

#### Algorithm 1. Finding All Regions Comprising MC.

---

```

 $T = \{r_m\}, R = \{r_1, r_2, \dots, r_n\}$ 
 $i = m$ 
while  $i > 1$  do
  if  $d(r_i, r_{i-1}) \leq \textit{gap}$  then
     $T = T \cup \{r_{i-1}\}$ 
  else
    break
  end if
   $i = i - 1;$ 
end while
 $i = m$ 
while  $i < n$  do
  if  $d(r_i, r_{i+1}) \leq \textit{gap}$  then
     $T = T \cup \{r_{i+1}\}$ 
  else
    break
  end if
   $i = i + 1;$ 
end while
return  $T$ 

```

---

Obviously, the parameter *gap* has an influence on the performance of R2L. We used a small set of test pages to empirically find a well performing default value for *gap*. For R2L a value of *gap* = 8 has proven to demonstrate good results.

#### 4.1.5 Fourth Phase: Extracting the Main Content from Selected Regions

In this phase, all Right-to-Left characters of the areas recognized in phase three are separated from all characters belonging to  $S_2$  and then are considered as the final output of the R2L algorithm. Effectively, the output then contains the MC.

### 4.2 Algorithm DANA

Here, we introduce the first extension of R2L, DANA, to improve the effectiveness of R2L. Since the R2L approach determines its output only from the Right-to-Left character set of the identified main content areas of web pages, it might miss some fractions of the MC. This happens when there are some English words or characters in the main content areas of web page. As the R2L algorithm is incapable to keep these English words in the extracted main content the recall score of R2L algorithm will not be optimal in these cases. This conceptual drawback is overcome by DANA.

DANA is divided into one preprocessing step and four phases. Empiric evaluation on our small set of test pages show that for DANA the best value for the parameter  $gap$  is 20, so all results produced by DANA are based on a value of  $gap = 20$ . The preprocessing step as well as phases one, two, and three are equivalent to the R2L approach. Thus, below we explain only the differences in phase four.

**Fourth Phase of DANA: Extracting the Main Content from Selected Regions Using a Parser.** In this phase of DANA, we feed only those HTML lines determined in the third phase of R2L as an input to an error-tolerant parser [5]. Following our hypothesis, the output of the parser is more accurate than the output of the phase four of the R2L approach, so DANA achieves overall extraction performance better than R2L. On the downside, applying the parser to selected fragments of a document causes an overhead in computation, so R2L achieves an overall better time performance than DANA. Concerning these two facts, we will see that the trade-off between efficiency against effectiveness is worth the runtime overhead.

### 4.3 Algorithm DANAg

The R2L and DANA algorithms are both language-dependent, while the second extension of R2L, called DANAg, is a generalized method which is able to run on web pages written in any language. The extraction process of DANAg is divided into one preprocessing step and four phases as well. The preprocessing step and phases two, three, and four are equivalent to the DANA approach. In the following, we explain only the differences in phase one.

#### First Phase of DANAg: Calculating the Length of Content and Code of Each Line.

In the first phase of the algorithm DANAg, our aim is to count and store the number of characters comprising both the content and the code of segments of the whole lines of the HTML file into two one-dimensional arrays  $T_1$  and  $T_2$ , respectively.

To provide two one-dimensional arrays  $T_1$  and  $T_2$ , first we count and store the number of characters of each line into one-dimensional array  $Length$ . In the second step, we feed the HTML file to our parser to extract all words representing the content of the

**Table 2.** Evaluation corpus of 2,166 web pages introduced in [13]

Web site	URL	Size	Languages
BBC	http://www.bbc.co.uk/persian/	598	Farsi
Hamshahri	http://hamshahronline.ir/	375	Farsi
Jame Jam	http://www.jamejamonline.ir/	136	Farsi
Al Ahram	http://www.ahram.org/	188	Arabic
Reuters	http://ara.reuters.com/	116	Arabic
Embassy of Germany, Iran	http://www.teheran.diplo.de/ Vertretung/teheran/fa/Startseite.html	31	Farsi
BBC	http://www.bbc.co.uk/urdu/	234	Urdu
BBC	http://www.bbc.co.uk/pashto/	203	Pashto
BBC	http://www.bbc.co.uk/arabic/	252	Arabic
Wiki	http://fa.wikipedia.org/	33	Farsi

HTML file. Afterwards, we search the extracted text, which are stored in an array of string, to find end-of-line delimiters. By this method, we are able to count and store the number of content characters for each line in a one-dimensional array  $T1$ . It is obvious that we can compute the number of characters in each line used in code elements by using Formula 2 and store results in array  $T2$ .

$$T2 = Length - T1 \quad (2)$$

Although DANAg generalizes the R2L and DANA to a language-independent approach, it is expected that incorporating the parser directly in this phase of DANAg causes a significant overhead in computation and that DANAg runs slower than DANA.

## 5 Evaluation

In this section we evaluate our MCE approaches under the aspects of efficiency and effectiveness. We look at the algorithms' runtime on real world data to measure their efficiency. Effectiveness is evaluated under the capability to precisely extract the main content from documents for which a gold standard is defined.

### 5.1 Data Sets

For evaluation we use two different corpora. The first corpus contains 2,166 web documents in Arabic, Farsi, Pashto, and Urdu (see Table 2). This dataset has been proposed in [13] and has been collected for the evaluation of MCE on Right-to-Left language web pages. The second corpus contains 9,101 web pages from different web sites (see Table 3). This dataset has been introduced in [5] and has been established for evaluation of MCE on Western language documents.

### 5.2 Evaluation Methodology

In order to calculate the accuracy of any MCE method, it is necessary to provide a manually crafted *gold standard* for the main content of all HTML files. Both corpora provide such a gold standard. For the purpose of evaluation, the output of an MCE

**Table 3.** Evaluation corpus of 9,101 web pages introduced in [5]

Web site	URL	Size	Languages
BBC	http://www.bbc.co.uk/	1,000	English
Economist	http://www.economist.com/	250	English
Golem	http://golem.de/	1,000	German
Heise	http://www.heise.de/	1,000	German
Manual	several	65	German, English
Repubblica	http://www.repubblica.it/	1,000	Italian
Slashdot	http://slashdot.org/	364	English
Spiegel	http://www.spiegel.de/	1,000	German
Telepolis	http://www.telepolis.de/	1,000	German
Wiki	http://fa.wikipedia.org/	1,000	English
Yahoo	http://news.yahoo.com/	1,000	English
Zdf	http://www.heute.de/	422	German

algorithm is compared with the gold standard of the corresponding HTML document. For comparing the *gold standard file* with the produced *cleaned file* it is essential to compute an overlap between the two of them. The establish method, introduced in [4] and used throughout several papers on MCE [5][6][14], is to use the Longest Common Subsequence [8] to find this overlap between the gold standard and the cleaned file. Now by counting the number of tokens of gold standard and cleaned files,  $g$  and  $m$  respectively, and the number  $k$  of tokens return by the LCS function we can evaluate the accuracy of the algorithm by applying the classical information retrieval performance measures – Recall, Precision, and F1-measure [4], as defined in formula [3]

$$r = \frac{\text{length}(k)}{\text{length}(g)}, p = \frac{\text{length}(k)}{\text{length}(m)}, F1 = 2 * \frac{p * r}{p + r} \quad (3)$$

### 5.3 Results

This section presents and dicusses the average F1 scores of our three algorithms explained in this chapter, R2L, DANA, and DANAg, and compares it to 11 main content extraction approaches which were introduced in Section [2] on Related Work.

#### 5.3.1 Results of R2L

Table [4] shows the F1 scores of R2L algorithm on the corpus composed of Right-to-Left language web documents. Column 3 shows F1 scores when considering the parameter *gap* set to default value of 8. We can see, that the achieved accuracy of R2L is very high in general. For many web sites, such as BBC, Hamshahri, Jame Jam it achieves nearly a perfect F1 score very close to 1 and it has very good F1 score for most other web pages.

Additionally we investigated the theoretical upper bound for MCE using R2L. Column 4 and 5 show a theoretical optimal setting for *gap*, which provides the best result that can be achieved with R2L. For example, on the Al Ahram data set, the optimal value for the *gap* parameter would be 7. In this case, we achieve an F1 score of 0.983, which is a significant improvement over the baseline of the fixed *gap* parameter. However, in most cases the optimum value for *gap* is not far from 8 and the best theoretical F1 score does not diverge much from the performance of R2L.



**Table 4.** The Average F1 Scores of R2L based on Table 2

Web site	Languages	F1 with <i>gap</i> = 8	Optimal value for <i>gap</i>	F1 with optimal <i>gap</i> in Col. 4
BBC	Farsi	0.991	8	0.991
Hamshahri	Farsi	0.991	8	0.991
Jame Jam	Farsi	0.977	3	0.987
Al Ahram	Arabic	0.929	7	0.983
Reuters	Arabic	0.936	4	0.971
Embassy of Germany, Iran	Farsi	0.954	15	0.971
BBC	Urdu	0.956	11	0.997
BBC	Pashto	0.974	8	0.974
BBC	Arabic	0.987	8	0.987
Wiki	Farsi	0.283	16	0.385

### 5.3.2 Results of DANA

Table 5 gives statistics showing the average F1 scores of DANA and other 11 algorithms on 2,166 selected web pages from 10 different web sites based on a parameter setting of *gap* to 20.

In Table 5, the bold values show the highest F1 score and the italic numbers represent the highest F1 score among all algorithms except DANA. In addition, in Table 6 we compute the processing performance in terms of data throughput (MB/s) of DANA and other methods. By looking to these two tables, the following important points can be noticed:

**Table 5.** The Average F1 Scores of DANA based on the corpus in Table 2

	Al Ahram	BBC Arabic	BBC Pashto	BBC Persian	BBC Urdu	Embassy	Hamshahri	Jame Jam	Reuters	Wikipedia
ACCB-40	0.871	0.826	0.859	0.892	0.948	0.784	0.842	0.840	0.900	0.736
BTE	0.853	0.496	0.854	0.589	0.961	0.810	0.480	0.791	0.889	<b>0.817</b>
DSC	0.871	<i>0.885</i>	0.840	<i>0.950</i>	0.896	0.824	<i>0.948</i>	0.914	0.851	0.747
FE	0.809	0.060	0.165	0.063	0.002	0.017	0.225	0.027	0.241	0.225
KFE	0.690	0.717	0.835	0.748	0.750	0.762	0.678	0.783	0.825	0.624
LQF-25	0.788	0.780	0.844	0.841	0.957	0.860	0.765	0.737	0.870	0.773
LQF-50	0.785	0.777	0.837	0.828	0.954	0.856	0.767	0.724	0.870	0.772
LQF-75	0.773	0.773	0.837	0.819	0.954	0.852	0.756	0.724	0.870	0.750
TCCB-18	<i>0.886</i>	0.826	<i>0.912</i>	0.925	0.990	<i>0.887</i>	0.871	<i>0.929</i>	<i>0.959</i>	0.814
TCCB-25	0.874	0.861	0.909	0.927	0.992	0.883	0.888	0.924	0.958	0.814
Density	0.879	0.202	0.908	0.741	0.958	0.882	0.920	0.907	0.934	0.665
DANA	<b>0.984</b>	<b>0.963</b>	<b>0.936</b>	<b>0.994</b>	<b>1.0</b>	<b>0.935</b>	<b>0.978</b>	<b>0.945</b>	<b>0.967</b>	0.674

- As can be seen from six web sites, Al Ahram, BBC Arabic, BBC Persian, BBC Urdu, Hamshahri, and Reuters, DANA achieves a F1 score higher than 0.95 and especially on BBC Urdu with an F1 score of exactly 1. No other method shows such a high effectiveness.
- In table 5 only BTE and only on Wikipedia web documents achieves an F1 score greater than DANA. Wikipedia documents have already been observed to be very

**Table 6.** Average processing performance (MB/s)

Method	Performance (MB/s)
ACCB-40	0.40
BTE	0.17
DSC	7.76
FE	14.33
KFE	11.76
LQF-25	1.25
LQF-50	1.25
LQF-75	1.25
TCCB-18	17.09
TCCB-25	15.86
Density	7.62
DANA	<b>19.43</b>
DANAg	<b>11.41</b>

difficult for MCE algorithms in other papers [5]. By looking inside the Wikipedia HTML file, we discover that there are big gaps, more than 20, between the regions composing the main content. Looking at DANA's recall of 0.5734 it can be seen that it erroneously discards large parts of the main content. In the previous section, we configured the gap parameter with a value of 20. If the gap parameter is set to 160 instead of 20, then DANA achieves a recall of 0.8364, a precision of 0.8974 and an F1 score of 0.8571. In this case, DANA outperforms all other algorithms. In our outlook at further work, we will suggest some ideas how to overcome this drawback of DANA to parametrize the gap value.

- Among all eleven algorithms only DSC and TCCB achieve F1 scores close to but never as high as DANA.
- We can see that DANA also shows good efficiency, of around 19.43 MB/S. Therefore, in comparison with the comparable methods in this chapter – DSC, TCCB-18 and TCCB-25, which have an extraction performance close to our algorithm – DANA has an acceptable efficiency. On Wikipedia, BTE achieves extraction performance better than DANA, but DANA is about 100 times faster than BTE.

### 5.3.3 Results of DANAg

Tables 7 and 8 give statistics showing the average F1 scores of DANAg and other main content extraction algorithms on both data sets. In addition, in Table 6 we compute the processing time (MB/s) of DANAg and other approaches on the data described in Table 2. By looking at the Tables 7 and 8, the following important observations can be made:

#### *Results on Arabian Language Documents*

- As can be seen in Table 7 from six web pages, Al Ahram, BBC Arabic, BBC Persian, BBC Urdu, Hamshahri, and Reuters, DANAg achieves F1 score more than 0.95 and especially on BBC Urdu with an F1 score extremely close to 1. In addition, no other method shows such a high effectiveness.
- In Table 7 only BTE and TCCB-18 achieve on wikipedia web documents and Reuters, respectively, an F1 score greater than DANAg. Wikipedia documents have already been observed to be very difficult for MCE algorithms [5].

- Among all eleven algorithms only DSC and TCCB achieve F1 scores close to but never as high as DANAg.

*Results on Western Language Documents.*

Now, we describe the results in Table 8. For better understanding, this table was divided into three parts. We explain each part of this table below:

- In the middle part of the Table 8, DANAg achieves F1 score higher than six web pages, golem, heise, republica, spiegel, telepolis, and yahoo. As can be seen, ACCB is the best algorithm, on three web pages, between all other algorithms after DANAg.
- The left side of Table 8 shows web pages that DANAg achieves F1 score less than DSC, CCB, and ACCB approaches. But as it can be seen, the differences between F1 score of DANAg and last three mentioned methods are 0.013, 0.0144, and 0.017 and it shows DANAg could be acceptable on these web pages as well.

**Table 7.** The Average F1 Scores of DANAg based on the corpus in Table 2

	Al Ahram	BBC Arabic	BBC Pashto	BBC Persian	BBC Urdu	Embassy	Hamshahri	Jame Jam	Reuters	Wikipedia
ACCB-40	0.871	0.826	0.859	0.892	0.948	0.784	0.842	0.840	0.900	0.736
BTE	0.853	0.496	0.854	0.589	0.961	0.810	0.480	0.791	0.889	<b>0.817</b>
DSC	0.871	0.885	0.840	0.950	0.896	0.824	0.948	0.914	0.851	0.747
FE	0.809	0.060	0.165	0.063	0.002	0.017	0.225	0.027	0.241	0.225
KFE	0.690	0.717	0.835	0.748	0.750	0.762	0.678	0.783	0.825	0.624
LQF-25	0.788	0.780	0.844	0.841	0.957	0.860	0.765	0.737	0.870	0.773
LQF-50	0.785	0.777	0.837	0.828	0.954	0.856	0.767	0.724	0.870	0.772
LQF-75	0.773	0.773	0.837	0.819	0.954	0.852	0.756	0.724	0.870	0.750
TCCB-18	0.886	0.826	0.912	0.925	0.990	0.887	0.871	0.929	0.959	0.814
TCCB-25	0.874	0.861	0.909	0.927	0.992	0.883	0.888	0.924	0.958	0.814
Density	0.879	0.202	0.908	0.741	0.958	0.882	0.920	0.907	0.934	0.665
DANA	<b>0.984</b>	0.963	0.936	0.994	<b>1.0</b>	<b>0.935</b>	0.978	0.945	<b>0.967</b>	0.674
DANAg	0.949	<b>0.986</b>	<b>0.944</b>	<b>0.995</b>	<b>0.999</b>	0.917	<b>0.991</b>	<b>0.966</b>	0.922	0.699

**Table 8.** The Average F1 Scores of DANAg based on the corpus in Table 3

	BBC	Economist	Zaf	Golem	Heise	Republica	Spiegel	Telepolis	Yahoo	Wikipedia	Manual	Slashdot
Plain	0.595	0.613	0.514	0.502	0.575	0.704	0.549	0.906	0.582	<b>0.823</b>	0.371	0.106
LQF	0.826	0.720	0.578	0.806	0.787	0.816	0.775	0.910	0.670	0.752	0.381	0.127
Crunch	0.756	0.815	0.772	0.837	0.810	0.887	0.706	0.859	0.738	0.725	0.382	0.123
DSC	<b>0.937</b>	0.881	0.847	0.958	0.877	0.925	0.902	0.902	0.780	0.594	0.403	0.252
TCCB	0.914	0.903	0.745	0.947	0.821	0.918	0.910	0.913	0.758	0.660	0.404	0.269
CCB	0.923	<b>0.914</b>	0.929	0.935	0.841	0.964	0.858	0.908	0.742	0.403	0.420	0.160
ACCB	0.924	0.890	<b>0.929</b>	0.959	0.916	0.968	0.861	0.908	0.732	0.682	<b>0.419</b>	0.177
Density	0.575	0.874	0.708	0.873	0.906	0.344	0.761	0.804	0.886	0.708	0.354	<b>0.362</b>
DANAg	0.924	0.900	0.912	<b>0.979</b>	<b>0.945</b>	<b>0.970</b>	<b>0.949</b>	<b>0.932</b>	<b>0.952</b>	0.646	0.401	0.209

- In the right side of Table 8, we see three web pages, manual, slashdot, and wikipedia which DANAg and other algorithms could not achieve considerable F1 score. At the moment, we are working on some extensions to overcome this drawback of DANAg.

We can see that DANAg also shows good efficiency, of around 11.41 MB/s. Therefore, in comparison with the comparable methods in this chapter – DSC, TCCB-18 and TCCB-25, which have an extraction performance close to our algorithm – DANAg has an acceptable efficiency.

As explained in 4.3, applying the parser in the first phase of DANAg causes an overhead in computation and as it shown in Table 6 DANAg has a lower efficiency in comparison to DANA (19.43 MB/s).

## 6 Conclusions and Future Work

In this chapter, we presented a novel main content extraction algorithm, R2L, and its two extensions DANA and DANAg.

The first proposed algorithm, R2L, very accurately extracts the main content from web documents with an F1 score  $> 0.929$ . This algorithm has two further technical advantages: 1) It is DOM tree and HTML-format independent; therefore, errors or non-standard compliant HTML documents do not pose a problem. 2) We do not need to use a parser for our algorithm. This improves runtime efficiency over many of the previous MCE methods which employed the DOM tree structure or used other output of HTML parsers for their purpose.

On the contrary, the R2L algorithm is not able to achieve an F1 score closer to 1 in the case of documents where there are some Non-R2L characters among words in the main content area. To overcome this problem we introduced DANA which feeds entire lines of an HTML file with an outline of the identified MC to an HTML parser. The output of our parser is exactly the main content of selected regions highlighted in the third phase of DANA.

DANA, the first extension of R2L, determines the main content with previously unseen accuracy. Achieving an average F1 score  $> 0.935$  on the test corpus used in this chapter, it outperforms all previous methods. Also, DANA succeeded to achieve F1 score greater than 0.96 on over six web sites and a perfect value of 1 on BBC Urdu.

The second extension of R2L, DANAg, is a language-independent version of DANA, with considerable effectiveness. Results show that DANAg determines the main content with high accuracy on many standard data sets. Achieving an average F1 score  $> 0.90$  on the test corpora used in this chapter, it outperforms the state of the art methods in MCE.

In the future and as a next research step, we will extend DANAg to use machine learning methods to group several areas in an HTML file contributing to the main content of web pages. This allows for discarding the parameter setting for gaps between main content blocks and to overcome the problem observed on certain documents in the evaluation corpora.

**Acknowledgements.** We would like to thank Dr. Norbert Heidenbluth for helping us to prepare figures and diagrams. We would like also to thank Dr. Koen Deschacht and the University of K.U.LEUVEN for providing us with the Content Extraction Software.

The research leading to these results has received partial funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257859, ROBUST.

## References

1. Debnath, S., Mitra, P., Giles, C.L.: Identifying Content Blocks from Web Documents. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 285–293. Springer, Heidelberg (2005)
2. Finn, A., Kushmerick, N., Smyth, B.: Fact or fiction: Content classification for digital libraries. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries (2001)
3. Gibson, D., Punera, K., Tomkins, A.: The volume and evolution of web page templates. In: WWW 2005: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, pp. 830–839. ACM Press, New York (2005)
4. Gottron, T.: Evaluating content extraction on HTML documents. In: ITA 2007: Proceedings of the 2nd International Conference on Internet Technologies and Applications, pp. 123–132 (September 2007)
5. Gottron, T.: Content code blurring: A new approach to content extraction. In: DEXA 2008: 19th International Workshop on Database and Expert Systems Applications, pp. 29–33. IEEE Computer Society (September 2008)
6. Gottron, T.: An evolutionary approach to automatically optimise web content extraction. In: IIS 2009: Proceedings of the 17th International Conference Intelligent Information Systems, pp. 331–343 (2009)
7. Gupta, S., Kaiser, G., Neistadt, D., Grimm, P.: DOM-based content extraction of HTML documents. In: WWW 2003: Proceedings of the 12th International Conference on World Wide Web, pp. 207–214. ACM Press, New York (2003)
8. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. *Commun. ACM* 18(6), 341–343 (1975)
9. Liu, C., Liao, B.: Gaussian smoothing-based web content extraction. *International Journal of Advancements in Computing Technology* 3(8), 255–262 (2011)
10. Mantratzis, C., Orgun, M., Cassidy, S.: Separating XHTML content from navigation clutter using DOM-structure block analysis. In: HYPERTEXT 2005: Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia, pp. 145–147. ACM Press, New York (2005)
11. Mohammadzadeh, H., Gottron, T., Schweiggert, F., Nakhaeizadeh, G.: Extracting the main content of web documents based on a naive smoothing method. In: KDIR 2011: International Conference on Knowledge Discovery and Information Retrieval, pp. 470–475. SciTePress (2011)
12. Mohammadzadeh, H., Gottron, T., Schweiggert, F., Nakhaeizadeh, G.: A fast and accurate approach for main content extraction based on character encoding. In: TIR 2011: Proceedings of the 8th International Workshop on Text-based Information Retrieval, DEXA 2011, pp. 167–171. IEEE Computer Society (2011)
13. Mohammadzadeh, H., Schweiggert, F., Nakhaeizadeh, G.: Using utf-8 to extract main content of right to left language web pages. In: Cuaresma, M.J.E., Shishkov, B., Cordeiro, J. (eds.) ICISOFT 2011 - Proceedings of the 6th International Conference on Software and Data Technologies, Seville, Spain, July 18-21, vol. 1, pp. 243–249. SciTePress (2011)

14. Moreno, J., Deschacht, K., Moens, M.: Language independent content extraction from web pages. In: Proceeding of the 9th Dutch-Belgian Information Retrieval Workshop, pp. 50–55 (2009)
15. Pasternack, J., Roth, D.: Extracting article text from the web with maximum subsequence segmentation. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 971–980. ACM, New York (2009), <http://doi.acm.org/10.1145/1526709.1526840>
16. Pinto, D., Branstein, M., Coleman, R., Croft, W.B., King, M., Li, W., Wei, X.: QuASM: a system for question answering using semi-structured data. In: JCDL 2002: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 46–55. ACM Press, New York (2002)
17. Weninger, T., Hsu, W.H.: Text extraction from the web via text-tag-ratio. In: TIR 2008: Proceedings of the 5th International Workshop on Text Information Retrieval, pp. 23–28. IEEE Computer Society (September 2008)

# Facilitating Structuring of Information for Business Users with Hybrid Wikis

Florian Matthes, Christian Neubert, and Alexander Steinhoff

Technische Universität München, Institute for Informatics  
Boltzmannstr. 3, 85748 Garching, Germany  
{matthes,neubert,steinhoff}@in.tum.de  
<http://wwwmatthes.in.tum.de>

**Abstract.** The flexibility and general applicability of wikis make them a valuable tool for information management and collaboration in modern enterprises. Since information in traditional wikis is only stored in form of unstructured text pages, the way it can be accessed by users is limited to fulltext search and navigation along the links in wiki pages. Some wikis try to overcome this limitation by allowing users to enrich text with semantic annotations, usually being defined in semantic web ontologies. While this provides database-like querying capabilities, it requires that business users learn a specific syntax and understand modelling concepts they are not familiar with.

Our approach, called Hybrid Wikis, introduces a small set of simple structuring concepts, namely attributes, type tags, and constraints. Furthermore, we propose to apply user interface elements that users understand, such as forms, spreadsheet-like tables, and faceted search interfaces, even though this limits expressive power. We illustrate our approach using an example scenario and highlight key implementation aspects of a Java-based hybrid wiki system. The paper concludes with practical experiences we gained in two usage scenarios, an overview of related work, and an outlook on future research.

## 1 Motivation and Problem Statement

To keep pace with the growing amount of digital information that has to be managed, enterprises have to adopt new tools and methods [1]. In the recent past, wikis are increasingly used as lightweight shared knowledge repositories that allow to collaboratively gather and consolidate information that was previously scattered across emails, files on personal computers and paper documents [2]. Having this information integrated in a central place, being able to search it and to connect related pieces of information with hyperlinks is in fact a major advance.

However, with a growing knowledge base soon the demand arises to access information in more structured ways that classical wikis do not support. For example, it is not possible to query a wiki for a company's research projects that started in the year 2010 or to export data about these project to a spreadsheet. This means that even if only rudimentary structured querying functionality is required, the enterprises have to resort to separate applications, usually specialized to manage information of particular domains (like employees, projects or customers) or they have to develop customized solutions. In both cases the advantages of storing information in a central repository are lost.

From a technical point of view, semantic wikis are promising tools to tackle this problem. They allow to combine textual contents with structured data. Typically, users have to provide this data in the form of semantic annotations to wiki pages or parts thereof. The structured part of the information in the wiki can then be queried similar to the contents of a database. However, in practice they are rarely used as a general purpose tool that dynamically adapts to new needs. In contrast, semantic wikis often are pre-configured by experts to solve particular problems. Although they can theoretically be used to structure arbitrary information, there are several barriers users are facing when editing content:

- Usually, a special syntax has to be used to add semantic annotations. This makes it difficult and cumbersome to edit structured content.
- Users are not familiar with the modelling concepts.
- It is not obvious for users how they can benefit from providing semantic annotations.

This paper describes a novel approach to mitigate these problems. In Section 2, our approach of so-called *hybrid wikis* is presented and illustrated using an example scenario. The main modelling concepts are described and the limitations of the approach are discussed. Important and interesting technical aspects of the implementation are covered in Section 3. Section 4 contains two case studies demonstrating the practicability of the approach. In Section 5, we give an overview of related work and highlight some examples of semantic wikis that use different approaches to facilitate data entry. The paper concludes with a short summary and an outlook on further research and planned improvements of the prototype.

## 2 Hybrid Wikis

In this section, we present the concept of *hybrid wikis*, our approach to the problem of facilitating the management of structured data for non-expert users is presented. The term ‘hybrid’ expresses that wikis combine a subset of the features of semantic wikis and classic text-based wiki software. Before we describe the structuring mechanisms that are available in the current software implementation on a conceptual level, we first give an overview of our goals, assumptions and motives that guided its development. Finally, we demonstrate how the modelling concepts can be applied in practice using a small example scenario.

### 2.1 General Principles and Design Rationale

In the development of hybrid wikis, our primary goal is to lower the barriers that non-expert users encounter when they try to structure the wiki contents as it has been described in Section 1. This means that for using the wiki neither special knowledge of wiki syntax or modelling concepts should be required nor should the user be forced to learn a query language to utilize the structured part of the wiki contents. We try to find ways to enable all users to enter structured data, in contrast to a two-phase process



where unexperienced users enter textual content that is later enriched with annotations by experts.

From our point of view, all attempts to translate between the expressivity of classic, established semantic web formalisms (like RDF<sup>1</sup> or OWL<sup>2</sup>) and the user by the means of new user interfaces are unsatisfactory. Our approach primarily focuses on the user and accepts that there are limitations in the complexity of modelling concepts users can be expected to understand. We start from lightweight structuring concepts and metaphors that users are familiar with and then, in a second step, we examine how the data the users provide by these simple means can be exploited by the system to offer features that usually require a formally defined data model.

Therefore, we rely mainly on simple keyword-like annotations of wikipages, dynamically compiled and easily extensible forms for data entry, and the presentation of data in automatically generated tabular views. In turn, we try to avoid the notion of semantic annotations being something that is optionally appended to pure text content and that is defined in a separately maintained ontology or schema. Instead, we attempt to allow the user to implicitly provide semantics by filling data in particular fields of a form or a table, by optionally creating new such fields on demand, and by the way the data is queried and displayed in different contexts.

Since we do not require that users explicitly maintain a data model, we focus on dynamically mining the wiki to ‘guess’ the model and provide users with input options to guide them towards a consistent data model and vocabulary. However, advanced users can impose a schema and define certain integrity constraints.

For querying and browsing we provide a general search interface that allows a faceted drill-down that is based on the structured contents of the wiki pages. Furthermore, contents can be accessed in a spreadsheet-like tabular form. Our assumption is that users feel familiar with this representation and in consequence are less hesitant to manipulate the data.

## 2.2 Structuring Concepts

Hybrid wikis were implemented as an extension to the wiki component of the commercial enterprise collaboration platform *Tricia*<sup>3</sup>. Wiki pages basically consist of a name and some rich-text content. Pages can be organized using text labels (*tags*) and page/sub-page relationships. We added two means for structuring the information on a wiki page: *attributes* and so-called *type tags*. Both can be used independently or in combination. They are described in the following.

**Attributes.** In their simplest form, attributes are key-value pairs that can be added to wiki pages. They consist of an attribute name – the key – and a value. A value has a specific data type. Per default, a value is a short text literal. Alternatively, the value’s data type can be date, number or link. A link value represents a reference to another wiki page. Attributes do not represent metadata but constitute the structured part of the

<sup>1</sup> <http://www.w3.org/RDF>; visited on November 27th 2011.

<sup>2</sup> <http://www.w3.org/2004/OWL>; visited on November 27th 2011.

<sup>3</sup> <http://www.infoasset.de>; visited on November 27th 2011.

Wikis » sebis Public Website » Research » Hybrid Wikis

Last editor: Alexander Steinhoff - Feb 11

## Hybrid Wikis

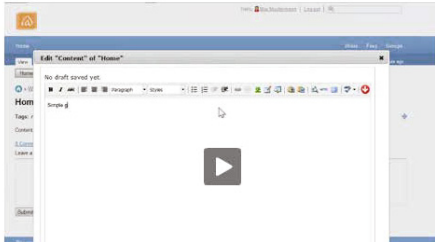
Tags: [earn](#) [hybrid wiki](#) [project](#) [research](#) [semantic](#) [social software](#) [wiki](#) [wiki-based-approach](#)  
[edit tags](#)

### Objective

Hybrid Wikis provide a lightweight semantic extension to the collaboration platform [Tindia](#) which comprises a traditional wiki as a core component. In contrast to more heavyweight approaches as semantic wikis, the Hybrid Wiki approach does not focus on annotating wiki content with semantic information corresponding to a fixed ontology, which could be regarded as a top-down (i.e., ontology first) approach. In contrast, it allows a kind of ontology or datamodel to emerge from the content by enabling users to easily add structured content to any wiki page in the form of arbitrary named attributes or tags. Guidance is provided by suggesting terms that are frequently used by other users and the ability to derive new pages from existing ones. This bottom-up approach is complemented with the ability to establish certain constraints such as mandatory attributes or inheritance relationships between types. By applying these to stable parts of the schema that emerged over time further processing of the structured content – for example the generation of visualizations – becomes possible.

### Screencast

Watch this video to see Hybrid Wikis in action:



Types: <a href="#">research project</a> <a href="#">edit tags</a>	
<b>Contact</b>	<a href="#">Christian Neubert</a>
<b>Team members</b>	<a href="#">Christian Neubert</a> <a href="#">Thomas Büchner</a> <a href="#">Christian M. Schweda</a> <a href="#">Sabine Buckl</a> <a href="#">Alexander Steinhoff</a> <a href="#">Sascha Roth</a>
<b>Project start</b>	2008
<b>Research area</b>	SoSo EAM
Sponsors:	
Name:	
Partners:	
Budget:	
Project end:	
Hosted at:	
<b>References</b>	
"Project" of	<a href="#">Rachelorthesis Martin Stange</a>

Fig. 1. Type tags and attributes in the context of a wiki page

content. For the user, the attributes are presented in a box containing a list of key-value pairs at the right border of the page (see Figure 1). The appearance is inspired by a kind of templates widely used in Wikipedia<sup>4</sup> – and thus the MediaWiki software<sup>5</sup> – to structure the contents of pages describing objects of the same type, like for example cities, countries or planets. Since the box forms the structured part of the page, it can also be compared to the so-called fact box of Semantic MediaWiki, that summarizes the facts being expressed by annotations in the text.

However, in hybrid wikis neither does the attribute box reflect facts defined somewhere else nor is the selection of attributes specified by a template. This means that new attributes can be added to individual pages simply by adding a new entry to the list of attributes. It is possible to assign multiple values for one attribute. The values are ordered and can be a mix of different data types, i.e., literals, links, dates, and numbers.

To avoid redundancy, and thus inconsistencies, and to facilitate navigation in the wiki, references from other pages are shown in an additional ‘references’ section of the attribute box. For example, if a page for a country  $S$  references the capital  $M$ , i.e., it contains an attribute with one value being a link to  $M$ , the page for  $M$  will show an additional attribute entry ‘capital of  $S$ ’ having the value  $M$ . The references section of the attribute box is similar to the incoming and outgoing links as displayed in the KiWi system<sup>6</sup>.

<sup>4</sup> <http://www.wikipedia.org>; visited on November 27th 2011.

<sup>5</sup> <http://www.mediawiki.org>; visited on November 27th 2011.

<sup>6</sup> <http://www.kiwi-project.eu>; visited on November 27th 2011.

**Type Tags.** Type tags allow the user to make a statement about the type of the object being described on the page. They are shown at the top of the attribute box as shown in Figure 1. Like for the normal tags used in the wiki (i.e., arbitrary text labels assigned to pages for categorization), users may choose an unlimited number of terms they consider appropriate.

These tags can then be used to generate lists of pages describing objects of the same type. While this is possible with standard tags as well, for example by searching for all pages tagged ‘university’, the results are more precise when type tags are used: users might use the same tag ‘university’ to categorize pages describing for example people working at a university, software products targeted at an academic audience or the concept of a university. Thus, type tags allow the user to make an important distinction when tagging wiki pages by stating that the tagged page describes an individual instance of the respective type – in contrast to vaguely relating it to a broad topic.

Retrieving a list of pages for a specific type tag can be achieved by simply clicking on the tag. If the pages contain attributes, they are displayed in a tabular view (as shown in Figure 2). The table can be sorted by all columns. By using the faceted search functionality of the Tricia platform type tag and attribute filters can be combined. In this way it is possible to define very specific subsets of the wiki contents. The results can optionally be presented in tabular form and they can be embedded in any wiki page.

It is important to note that on the one hand it was consciously avoided to force users to explicitly create relationships between type tags and attribute names as it would be done when defining a template. On the other hand, experienced users can assign a list of attribute names to a type tag. These attribute names are then highlighted to the user whenever the respective type tag is used. It is further possible to specify integrity constraints like the type of value, the number of values or allowed value ranges for each attribute.

Even if no explicit relations between type tags and attributes are defined, the system analyzes the data in the wiki to infer implicit relations based on their usage. This is described in the following section.

### 2.3 Input Support

In order to encourage users to structure the content they create, we strived for a convenient user interface: When viewing a wiki page, the user is offered a selection of names of attributes she might want to add. These suggestions are shown together with the already assigned attributes at the bottom of the attribute box so that the user only has to provide proper values (see Figure 1). The list of suggestions consists of the most frequent attribute names used on similar pages. Type tags are the most important indicator for this similarity.

In the simplest case, exactly one type tag is assigned to a page. The system then determines the set of attributes used in combination with this type tag on other pages and displays the most frequent attribute names. In the case of multiple type tags, not only the attribute frequency is considered but also if an attribute occurs on pages having all or several of the type tags assigned. Such attributes are preferred in the list of suggestions. If no type tags are present on the given page, similar pages are determined by taking into account only the attribute names used.

Wikis » sebis Public Website » Type Tags » research project

Wiki Pages with Type Tag **research project** in sebis Public Website

Showing 1 to 10 of 11 entries

	Team members (11)	Research area (11)	Project start (11)	Contact (10)	Partners (4)	Sponsors (4)	Project end (2)	Name (1)
<a href="#">EAM for the German Public Sector</a>	<a href="#">Sabine Bueckl</a> , <a href="#">Prof. Dr. Florian Matthes</a>	EAM	2010	<a href="#">Sabine Bueckl</a>	Bundesministerium des Inneren (BMI)			
<a href="#">EAM in the Context of Mergers and Acquisitions</a>	<a href="#">Christopher Schulz</a> , <a href="#">Andreas Freitag</a>	EAM	2009	<a href="#">Christopher Schulz</a> , <a href="#">Andreas Freitag</a>				
<a href="#">EAM Pattern Catalog</a>	<a href="#">Sabine Bueckl</a> , <a href="#">Dr. Josef Lanke</a> , <a href="#">Christian M. Schweda</a>	EAM	2008	<a href="#">Sabine Bueckl</a> , <a href="#">Christian M. Schweda</a>			2010	
<a href="#">EAMML - Enterprise Architecture Management Method Library</a>	<a href="#">Christian M. Schweda</a>	EAM	2009	<a href="#">Sabine Bueckl</a>		Bausparkasse Schwäbisch Hall BOC Deutschland	2010	
<a href="#">Hybrid Wikis</a>	<a href="#">Christian Neubert</a> , <a href="#">Thomas Büchner</a> , <a href="#">Christian M. Schweda</a> , <a href="#">Sabine Bueckl</a>	SoSo EAM	2008	<a href="#">Christian Neubert</a>				

Fig. 2. Tabular view of wiki page attributes

This makes it in effect very convenient to create a new page of a type previously used somewhere else in the system. Once a type is assigned, providing the attributes is hardly more demanding for the user than filling out a form. From this point of view, type tags can be considered a more flexible alternative to templates.

Similar to the attribute suggestions, the system recommends additional type tags based on the type tag combinations on other pages. If the user assigns a certain tag that is never used without another one – which is usually more general – the system can assign the other tag automatically. This depends on the number of other wiki pages supporting the assumption that there is really a type-subtype relation between the two tags.

Finally, the input fields for attribute names, attribute values and type tags display autocomplete suggestions as the user is typing. When an attribute value is typed, the respective attribute name is factored in to improve the quality of the suggestions. This makes it comfortable to contribute structured content and additionally fosters consistent usage of terms.

## 2.4 Limitations

Since the expressivity of established semantic web technologies is sacrificed in favour of a better user experience, modelling capabilities of hybrid wikis are constrained. In particular it is not possible to explicitly define type-subtype relationships between type tags. If a type tag is the generalisation of another type tag in the system, it has to be

manually ensured that each page with the more specific type tag also has the more general type tag assigned. However, the system supports the user in maintaining these implicit relationships (see Section 2.3).

It is not possible to specify any semantic relation for attributes. Since no reasoning capabilities are provided by the system, adding properties like symmetry or transitivity to an attribute would have no effect. Yet, attributes containing links are also visible on the target pages as a reference. This means that relationships between two pages are always owned by one of them and it is often not trivial to determine which one should be the owning side (does capital city link to country or country to capital?). This depends on the multiplicity of the relation and in particular cases on the access rights of the pages. However, in order to lower the barriers for entering structured data, the fact that this decision possibly has important implications is not explicitly communicated to the user. In contrast, it is relied on the ability to invert these links later when it becomes necessary.

Finally, querying capabilities are limited in so far that results can only be filtered according to the very attributes of pages. It is for example not possible to express a query targeting all pages having the attribute ‘Owned by’ set to a page with the type tag ‘Company’. Requiring that the attribute points to a specific page is of course possible. From a technical point of view, attribute (and type tag) filters could be flexibly combined using boolean operators. The current implementation supports only conjunction of filters to keep the user interface simple. From our experience, this is sufficient for browsing the wiki contents.

## 2.5 Example Scenario

In the following, we illustrate the modelling capabilities of hybrid wikis taking the example of a small company’s intranet wiki. Among other things, the wiki is used for gathering the knowledge about projects and people relevant for the company. We assume that while there may be many pages holding information about a person or project, there is one dedicated page for each such entity that can be considered the primary page which holds the basic information about it and optionally links to pages with more specific information. We further assume that in the beginning, there is only little content in the wiki and no type tags and attributes are used.

As the number of projects increases over time, there is a growing demand for a more structured view on project related wiki content. Attributes for project start and end dates are thus added to the respective pages and the type tag ‘Project’ is assigned. Having marked all project pages with the type tag, an overview table of all projects is instantly available showing sortable columns for the date attributes. Attribute values can be changed directly in this overview table. By this means, consolidating the project data, i.e., adding missing information or standardising the representation of attribute values, is facilitated.

Let’s assume many of the employees have created profile pages for themselves in the wiki to provide some information about their specific skills and experiences. Some of them independently start to add attributes to the project pages to express their relationship with these projects. For example, they state that they are members of the project team or project managers. In the beginning, people use different terms to describe their

roles. As these inconsistencies become visible in the overview table, they are quickly harmonized by the wiki users. As a result, if somebody now creates a new wiki page and assigns the type tag ‘Project’, she is offered to provide values for the attributes ‘Start date’, ‘End date’, ‘Project manager’ and ‘Team’. If she now adds a link to her profile page to the ‘Project manager’ attribute, this reference will be automatically visible on her page in the ‘References’ section of the attribute table: A new entry ‘Project manager of’ with a single value being the link to the project is displayed.

Starting from this basic schema, the company can further adapt it to new needs: Besides adding more attributes, the existing ones can be refined. The attribute ‘Project manager’ can be made mandatory for the type tag ‘Project’ so users are additionally reminded to provide this attribute. If it is omitted, the page is flagged as invalid. It can be further specified that only a single link to a page having the type tag ‘Person’ is accepted as a value.

New type tags can be added to distinguish different types of projects like ‘Research project’ or ‘Internal project’. On the one hand, this allows the generation of lists and tables containing only the respective subset of the projects. On the other hand, the system is supported in offering the user more relevant attribute suggestions. For example, the attribute ‘Field of research’ could be suggested for a research project whereas attributes only relevant for internal projects are not shown. It is also possible to add a temporal dimension to the project types by adding type tags like ‘Project in preparation’, ‘Current project’ or ‘Completed project’. Using type tags instead of status attributes has the advantage that again attribute constraints can be related to the types: Each page of a ‘Completed project’ can be required to contain a value for the ‘End date’ of the project.

### 3 Implementing Hybrid Wikis

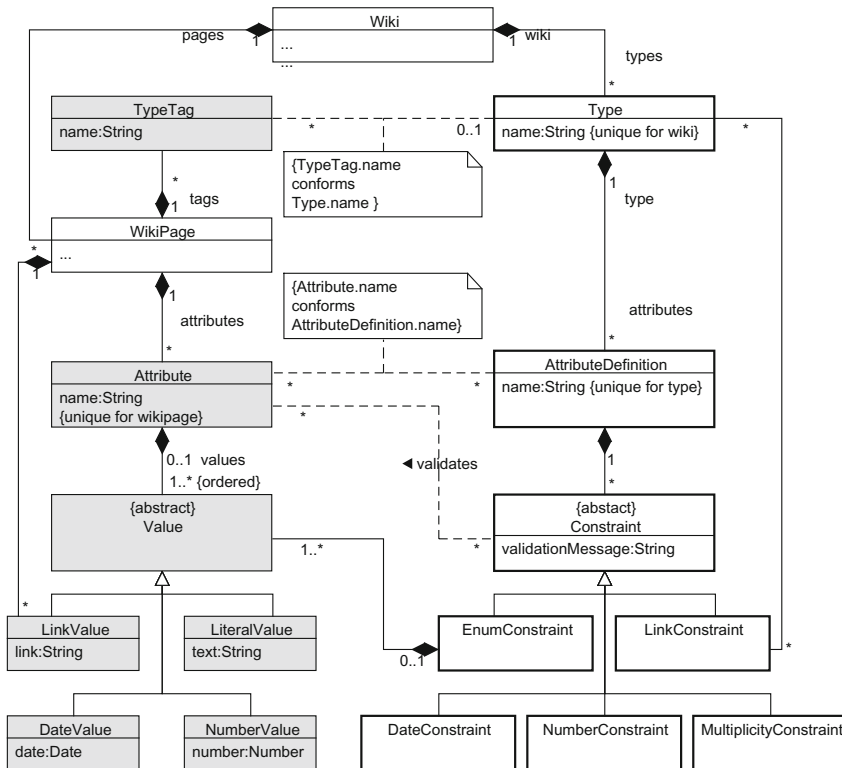
The concept of hybrid wikis is developed by members of our chair since 2009. The system is built on the experiences made with classic and semantic Wiki technology as well as integrated Enterprise 2.0 platforms and it is currently realized based on the modelling framework provided by Tricia enabling the model-driven development of web cooperation systems [3]. Hybrid wikis extend the Tricia wiki functionality by the mechanisms described in Section 2.

The data model underlying the implementation of hybrid wikis is shown in Figure 3. Wiki and wiki page are built-in concepts provided by Tricia, the other elements are additionally provided by hybrid wikis. Gray colored elements refer to concepts used for the structuring of individual wiki pages, elements with thick borders represent concepts defining a schema, i.e., types, attribute definitions, and constraints.

The illustrated concepts are currently mapped to a SQL database<sup>7</sup> by means of Tricia’s object relational mapping mechanism. In the following we will explain the data model of hybrid wikis and discuss some impacts on the system behaviour.

For each wiki page, multiple attributes can be assigned. Each of the attributes has a non-empty list of ordered values being either literal, link, date or number. A wiki page can have multiple tags assigned. A tag is a simple character sequence and either a

<sup>7</sup> SQL is the default storage implementation, but other implementations are also supported, e.g., NoSQL storages.



**Fig. 3.** The Data model of Hybrid Wikis

normal tag or a type tag. WikiPage, TypeTag, Attribute and Value constitute the set of concepts which are provided for structuring information in hybrid wikis.

Type, AttributeDefinition, and Constraint are additional concepts to enable users to define types and attributes more precisely and specify validation rules. A type contains the set of attributes that the users are urged to provide in combination with the respective type tag. These attributes can be further used to restrict the range of attribute values with integrity constraints.

A type is loosely coupled to a type tag by name, the same applies to attribute and attribute definition. The fact that an attribute definition for an attribute  $A$  is bound to a type for a tag  $T$  means that on pages with type tag  $T$  an attribute suggestion  $A$  is shown, which takes precedence over the default attribute suggestion mechanism. In consequence, it is not required that a page having both, tag  $T$  and attribute  $A$ , assigned exists in the system. Thus, new attributes can be seeded top-down over typed pages by means of suggestions.

Furthermore, for individual attribute definitions, validation rules can be specified by using constraints. The EnumConstraint defines a finite set of values – of any



type – that may be assigned to an attribute. The `MultiplicityConstraint` allows to specify how many values an attribute should have, e.g., at-least-one, at-most-one, exactly-one. The `LinkConstraint` checks if the values of an attribute are hyperlinks. Additionally, it can be defined that these hyperlinks have to be links to wiki pages with specific types. `DateConstraint` and `NumberConstraint` check if the values of an attribute are of type date or number respectively.

The validation rules of an attribute constraint apply when a wiki page is displayed. This means that if a wiki page has an attribute  $A$ , a type tag  $T$ , and a type for  $T$  exists having an attribute definition matching the name of  $A$ , the constraints will be applied to  $A$  and its current values. In case of a validation failure, the `validationMessage` of the constraint is shown in the context of the attribute. The situation that two type tags specifying contradicting attribute constraints are assigned to a wiki page is currently not prevented by the system and users have to solve such conflicts manually.

Beside showing validation messages for wiki page attributes, the integrity constraints are also used to improve the input support for attribute values. For instance, if an enumeration constraint applies to an attribute, in the value auto-completion control only the elements are offered which are specified for this constraint, e.g., {‘open’, ‘closed’}.

Since constraints in hybrid wikis never prevent users from entering values that contradict the specified validation rules, we call them *soft constraints*. The system assists users in systematically searching for contradictions. As a side-effect, it is possible to import information from arbitrary data sources to hybrid wikis, e.g., from a Microsoft Excel data sheet, without being restrained by hard integrity constraints. Any conflicts with the schema can be fixed after the import.

Since the main purpose of Tricia is to enable users to find relevant information within enterprises quickly, the content of all elements (e.g., files, wikis, blogs) is indexed by means of the Lucene information retrieval software library<sup>8</sup>. In addition to simple text queries, querying for metadata like tags or date of last modification is supported in this way. For the implementation of hybrid wikis, Lucene plays a critical role in the dynamic generation of attribute suggestions and overview tables. Lucene supports both features by providing flexible filtering capabilities together with a very fast access to particular fields of the indexed entities. For example, to display a table of all pages having a specific type tag, Lucene can efficiently determine the respective set of pages (filtered by the access rights of the user) and provide the set of attribute names used on each page. From the analysis of the frequency of attribute names the selection and order of the table columns is then determined.

It is possible to import and export hybrid wiki data by using the standardized Eclipse Modeling Framework (EMF) exchange format. Models can be imported and exported as ecore-files, model instances as xmi-files. Furthermore, the contents of hybrid wikis can be imported from and exported to spreadsheets (e.g., Microsoft Excel). Additionally, it is possible to visualize the contents of hybrid wikis by using an external service (System Cartography Service<sup>9</sup>) that is accessed via RESTful API calls.

<sup>8</sup> <http://lucene.apache.org>; visited on November 28th 2011.

<sup>9</sup> <http://wwwmatthes.in.tum.de/wikis/sebis/syca>; visited on November 27th 2011.



## 4 Practical Experiences

Hybrid wikis are currently evaluated in several industrial and research projects. In the following, we briefly describe the usage scenarios of hybrid wikis' technology in two selected cases. The first is taken from one of our research projects *Wiki4EAM*<sup>10</sup> where hybrid wikis are used for the collaborative documentation of system landscapes in enterprises in order to evolve models representing these landscapes bottom-up. The second scenario is the application of hybrid wikis as a simple issue tracker in a German software development company, the InfoAsset AG<sup>11</sup>. For both cases we sketch the models being developed bottom-up via attribute and type suggestions as well as the (soft-)constraints which were defined afterwards.

### 4.1 Wiki4EAM Community

In the Wiki4EAM community, founded in December 2010 at TUM, members share their experiences regarding the use of hybrid wikis in the context of enterprise architecture management (EAM). A prerequisite for adequate management of the enterprise architecture (EA) is to capture its current state in a model. Since the knowledge of all different elements to be considered in the model (e.g., business processes, applications, organizational units) is spread over all the different stakeholders in the company, the documentation of the current state of the EA remains a challenging task. Hybrid wikis enable these stakeholders to document the particular parts of the EA they are responsible for by means of (hybrid) wiki pages. Thereby, the data model emerges bottom-up by creating, editing, linking, and structuring these particular pages. A preliminary data analysis in two German companies from December 2010 to January 2011 showed the following model evolution:

- Company A: 12 concepts (type tags) with 63 attributes (textual and links), 100 wiki pages, 2 constraints were created within two weeks by four participating stakeholders (editors).
- Company B: 18 concepts with 60 attributes (textual and links), 120 wiki pages, 28 constraints were created within one month by five participating stakeholders.

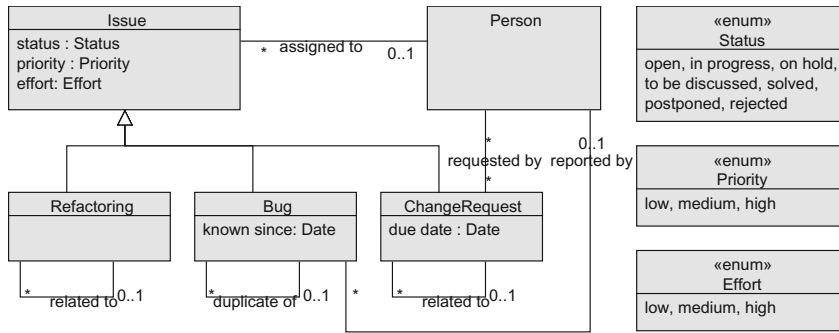
Although these numbers are not a founded empirical evaluation, they allow to assume that in a relatively short time period models can emerge bottom-up by the collaborative documentation of the particular elements when lightly structured wiki pages are used. Indeed, feedback from the Wiki4EAM community members indicates that hybrid wikis are well suited in cases where the target model is not completely known. They also confirmed that attribute suggestions facilitate the evolution of the information model.

### 4.2 InfoAsset Bugtracker

The InfoAsset AG providing the Tricia platform for commercial use, uses hybrid wikis among others things for its issue management. For each issue a hybrid wiki page with

<sup>10</sup> <http://www.matthes.in.tum.de/wikis/sebis/wiki4eam>; visited on November 27th 2011.

<sup>11</sup> <http://www.infoasset.de>; visited on November 27th 2011.



**Fig. 4.** Emerged Concepts of the InfoAsset Bugtracker

tag ‘unprocessed’ and type tag ‘issue’ is created. All unprocessed issues are shown in a list on the Tricia developer dashboard, that is built only using standard features of hybrid wikis. This list is dynamically generated by using an embedded query which retrieves all wiki pages having the tag ‘unprocessed’ and type tag ‘issue’ assigned. When a new entry appears in this list, i.e., the underlying embedded query yields a new search hit, the developers are informed through an RSS-Feed. A responsible person processes these new entries and categorizes them by assigning additional type tags. Currently two kinds of additional type tags are mainly used: ‘Bug’ and ‘ChangeRequest’. Furthermore, a person is assigned who is responsible for processing this issue by setting the attribute ‘assigned to’ to the respective name and to delete the tag ‘unprocessed’ in order to remove this issue from the dashboard list. The responsible person is in turn informed by an RSS-Feed. The complete data model is shown as a UML class diagram in Figure 4.

The schema emerged bottom-up by management of the particular wiki pages. Only the following attribute constraints were defined afterwards:

- Enumeration types for the attributes ‘Status’, ‘Priority’, and ‘Effort’ were created. In the beginning these values were plain strings.
- The cardinality for the relationship ‘assigned to’ between ‘Issue’ and ‘Person’ instances was set to ‘exactly-one’.

Note, that the cardinality in the UML diagram of Figure 4 is given with 0..1. This is due to the fact that new reported issues do not have a responsible person assigned. For these instances a warning message is shown indicating that there should be at least one person defined.

## 5 Related Work

Another way to reuse content structures among wiki pages is provided by wiki templates. In [4], Haake et al. discuss the need for structure in wikis based on templates, in [5] different wiki templating approaches are compared to each other. Although hybrid wikis explicitly do not support templates, attribute and type tag suggestions, based on

a statistical analysis of their usage, enable authors to reuse well established type structures, similar to templates.

The *Semantic MediaWiki* project is the most prominent example in the category of semantic wikis. This project adds database-like structuring and querying capabilities on top of an existing wiki, without requiring users to develop or adhere to a rigid database schema when authoring content [6]. This project also includes various features for browsing, searching, and aggregating the wiki's content as well as embedding queries in the wiki pages. Additionally, Semantic Media Wiki provides a model export to the standardized format OWL/RDF. In contrast to hybrid wikis, meta information can only be added to the pages by directly editing the markup in wiki syntax.

This issue is addressed by *Semantic Enterprise Wiki (SWM+)* [12], a set of open-source extensions to the Semantic MediaWiki. In this approach, meta data (properties) can be defined by means of a graphical user interface, called 'semantic toolbar', so users are not forced to write semantic annotations manually. Categories can be used for the classification of pages. They are shown together with the properties as a list in the semantic toolbar. The approach differs from hybrid wikis as follows:

- A separate annotation mode (semantic toolbar) is needed to enter structured information.
- Users have to create and edit both, the content of the wiki page and the semantic annotations. Users also have to be aware of content and annotations being synchronized, which is an laborious and error-prone task.
- No attribute (property) suggestions based on the types (categories) are provided.

*AceWiki* is a semantic wiki using controlled natural language for ontology management [7]. In [8], a lightweight approach for editing ontologies is introduced. Like hybrid wikis, both approaches try to facilitate structured data entry. However, in *AceWiki* this is achieved by natural language processing, in the second case by introducing a lightweight ontology editor.

The open source project *TWiki* [13] and its fork *Foswiki* [14] try to combine the advantages of wikis and database systems by allowing the user to attach data records to wiki pages. Although the schema can be changed at runtime, the effort is considerably higher than for hybrid wikis and the usage of a special wiki syntax is required.

The research prototype *SnoopyDB* [9] focuses particularly on recommending attributes to users when they are entering data. Types cannot be specified at all but all attributes are suggested based on attributes of other pages. This makes it very difficult to generate tabular overviews of pages covering certain kinds of things.

*MokiWiki* [10] is a plain modelling wiki, used to model the constituent parts of an enterprise collaboratively. With hybrid wikis we focus on facilitating collaborative data and information management within enterprises. Information models emerge bottom-up as a by-product [15].

<sup>12</sup> <http://wiki.ontoprise.de>; visited on November 27th 2011.

<sup>13</sup> <http://twiki.org>; visited on November 27th 2011.

<sup>14</sup> <http://foswiki.org>; visited on November 27th 2011.

<sup>15</sup> While hybrid wikis are not mainly built for modelling purposes, the models emerging through data management in hybrid wikis can also be used to represent the particular enterprise constituents (c.f. [11] and Section 4).

In [12], the prototype system *Social Infobox* is introduced. The system supports property tagging, a method to freely add attribute-value pairs to wiki-like resources without an explicitly defined schema. A user study showed that property names were shared among different resources which the authors interpret as the emergence of implicit types. Furthermore, the system provides property suggestions based on an analysis of attribute cooccurrences. Both ideas, data-driven bottom-up schema evolution and attribute suggestions, are similar to hybrid wikis but there are two important differences:

- By using type tags, hybrid wikis enable users to *explicitly* make a statement about the type. Since attributes can be used independently, this does not limit the flexibility.
- Beside the analysis of attribute cooccurrences, attribute suggestions in hybrid wikis are additionally based on the combinations of type tags and attributes used in the system. This means, that not only the frequency of occurrence is considered, but additionally, attributes are preferred when they occur together with many or all of the type tags.

In the Enterprise 2.0 Tool Survey that we conducted in 2008 [13], the functional capabilities of integrated Enterprise 2.0 platforms are compared using a multi-dimensional classification and evaluation framework. Especially the update of the survey in 2010<sup>16</sup> includes functional aspects regarding the ‘Structuring of Content’ as well as ‘Templates for Structured Content’. The fact that leading social software vendors include capabilities for structuring contents indicates an increasing demand in enterprises.

## 6 Summary and Outlook

In this article, we introduced hybrid wikis, a lightweight approach for data and information management that facilitates structuring of contents for business users. The main purpose of hybrid wikis is to avoid requiring users to learn complex semantic languages for adding structure to wiki pages. Instead, we provide an easy way for structured data entry with a small set of modelling concepts, namely attributes (key-value-pairs) and type tags. Furthermore, attribute suggestions encourage users to employ these structuring elements. Suggestions and auto-completion foster a common vocabulary, so users immediately benefit from the structure through better search capabilities. By means of these mechanisms, complex structures emerge bottom-up during management of particular data sets. To define validation rules we introduced the concepts of types and attribute constraints. They can additionally be used to define advanced rules and parts of the data model in a top-down manner.

Furthermore, we described the practical experiences made with hybrid wikis in two concrete projects, one from the industry and one as part of our research endeavours. For the latter, we sketched the models emerging bottom-up by means of attribute suggestions as well as the constraints additionally defined top-down afterwards.

We see potential to improve and validate our approach in the following ways:

- Compare hybrid wikis to similar wiki-based approaches (c.f. Section 5). For instance, we will analyze the revision history in order to compare the evolution of structure compared to hybrid wikis.

---

<sup>16</sup> <http://www.matthes.in.tum.de/wikis/enterprise-2-0-survey/home>, visited on November 27th 2011.

- Support advanced visualizations of structured data, which also can be used for navigating the wiki contents.
- Improve the way relationships between pages are handled (c.f. Section 2.4).

## References

1. Edmunds, A., Morris, A.: The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management* 20, 17–28 (2000)
2. Stocker, A., Tochtermann, K.: Exploring the value of enterprise wikis: A multiple-case study. In: *KMIS* (2009)
3. Büchner, T., Matthes, F., Neubert, C.: Data Model Driven Implementation of Web Cooperation Systems with Tricia. In: Dearle, A., Zicari, R.V. (eds.) *ICOODB 2010*. LNCS, vol. 6348, pp. 70–84. Springer, Heidelberg (2010)
4. Haake, A., Lukosch, S., Schümmer, T.: Wiki-templates: adding structure support to wikis on demand. In: *WikiSym 2005: Proceedings of the 2005 International Symposium on Wikis*, pp. 41–51. ACM Press, New York (2005)
5. Di Iorio, A., Vitali, F., Zacchirola, S.: Wiki content templating. In: *WWW 2008: Proceeding of the 17th International Conference on World Wide Web*, pp. 615–624. ACM, New York (2008)
6. Krötzsch, M., Vrandečić, D., Völkel, M.: Semantic MediaWiki. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006*. LNCS, vol. 4273, pp. 935–942. Springer, Heidelberg (2006)
7. Kuhn, T.: *Acewiki: A natural and expressive semantic wiki*. Technical Report arXiv:0807.4618, Department of Informatics University of Zurich (2008); Comments: To be Published as: *Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. *CEUR Workshop Proceedings*
8. Paschke, A., Coskun, G., Luczak-Rösch, M., Oldakowski, R., Harasic, M., Heese, R., Schäfermeier, R., Streibel, O.: *Realizing the corporate semantic web: Concept paper*. Technical report, Freie Universität Berlin, Berlin (2009)
9. Gassler, W., Zangerle, E., Tschuggnall, M., Specht, G.: Snoopydb: narrowing the gap between structured and unstructured information using recommendations. In: *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, HT 2010*, pp. 271–272. ACM, New York (2010)
10. Ghidini, C., Kump, B., Lindstaedt, S., Mahbub, N., Pammer, V., Rospocher, M., Serafini, L.: MoKi: The Enterprise Modelling Wiki. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) *ESWC 2009*. LNCS, vol. 5554, pp. 831–835. Springer, Heidelberg (2009)
11. Buckl, S., Matthes, F., Neubert, C., Schweda, C.M.: A wiki-based approach to enterprise architecture documentation and analysis. In: *The 17th European Conference on Information Systems (ECIS) – Information Systems in a Globalizing World: Challenges, Ethics and Practices*, Verona, Italy, June 8–10, pp. 2192–2204 (2009)
12. Hamasaki, M., Goto, M., Takeda, H.: Social infobox: collaborative knowledge construction by social property tagging. In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, CSCW 2011*, pp. 641–644. ACM, New York (2011)
13. Büchner, T., Matthes, F., Neubert, C.: A concept and service based analysis of commercial and open source enterprise 2.0 tools. In: *KMIS*, pp. 37–45 (2009)

**Part V**  
**Knowledge-Based Systems**

# Statistical and Possibilistic Methodology for the Evaluation of Classification Algorithms

Olgierd Hryniewicz

Systems Research Institute, Polish Academy of Sciences, Newelska 6, Warsaw, Poland  
hryniewi@ibspan.waw.pl

**Abstract.** In the paper we consider the problem of the statistical evaluation and comparison of different classification algorithms. For this purpose we apply the methodology of statistical tests for testing independence in the case the multinomial distribution. We propose to use two-sample tests for the comparison of different classification algorithms. In the paper we consider only the case of the supervised classification when an external ‘expert’ evaluates the correctness of classification. The results of the proposed statistical tests are interpreted using possibilistic methodology based on indices of dominance introduced by [7].

**Keywords:** Classification, Accuracy, Statistical Tests of Independence, Multinomial Distribution, Comparison of Algorithms, Possibility and Necessity Indices.

## 1 Introduction

Statistical algorithms used for classification (discrimination) and clustering of observations (data points, data records) are considered as a part machine learning. Classification algorithms in machine learning are considered as the algorithms of supervised learning. On the other hand, data clustering algorithms in machine learning are used as the algorithms of unsupervised learning. In this paper we will discuss the problem of the evaluation of the quality of the algorithms used for classification, usually understood as the accuracy of classification, from a statistical point of view. A natural measure of such quality is the percentage of correctly classified objects, usually called *classification accuracy*. This measure is used by all authors of papers devoted to classification problems, both developers of new algorithms, and users of existing algorithms who apply them for solving practical problems.

Quality of classification measured by the *accuracy* index may not be sufficient for the comparison of algorithms. Consider for example a decision support system that classifies patients to different classes of illness. It is usually not unimportant if all false classifications are evenly distributed over all possible classes or if they are concentrated in one class. When we have only two classes or when this distinction is not important we can use indices whose background can be found in medical sciences, namely the indices of *sensitivity* and *specificity*. Let us assume that considered objects can be assigned to two disjoint classes called ‘positive’, and ‘negative’. By *sensitivity* (also known in machine learning as *recall*) we understand the conditional probability that the object which should be classified to the ‘positive’

class has been correctly assigned to this class. By *specificity* (also known in machine learning as *recall of negatives*) we understand the conditional probability that the object which should be classified to the ‘negative’ class has been correctly assigned to this class. For good classification rules the values of these indices should be both close to one. In machine learning some functions of these indices (e.g. *F-measures* or *ROC diagrams*) are used. For more information see e.g. Chapter 7 in [2].

When the number of possible classes is larger than two we have to take into account statistical relationship between errors of different kind. Some measures proposed for the evaluation of algorithms in the case of multiple classes, like e.g. the *error correlation EC*, have probabilistic interpretation, but the majority of them are based on some heuristics. For more information on this subject see e.g. Chapter 11 in [15]. The lack of statistical interpretation is of lesser importance if we deal with only one set of data. However, automatic classifiers may be used in situations when analyzed data sets may belong to different populations. For example, in automatic inspection of production processes classifiers are designed at the outset of the process using some training data, and then used using different set of ‘test’ data acquired from a production process. In such cases possible classification algorithms should be compared using statistical methods. We believe that without statistical interpretation we are not able to present sound comparisons of different algorithms.

Application of statistical tests for the evaluation of classification algorithms has been proposed in [10]. In this paper, which presents an extended analysis of some problems considered in that paper, we propose to use some known statistical tests to evaluate and compare the quality of classification algorithms. In the second section of the paper we consider the problem of the comparison of algorithms. We consider two important practical cases. First is typical to the problem of supervised learning when the quality of classification of compared algorithms is evaluated using classification provided by an expert. In the second case, typical for algorithms related to unsupervised learning, we deal only with purely random data yielded by the compared algorithms.

The main problem with the application of different statistical tests is related to their interpretation. In the third section of the paper we propose a new application of *possibilistic measures* for the comparison of classification algorithms. This measures are based on the possibilistic interpretation of statistical tests proposed in [9], and provide the user with information about possibility or necessity of preferring one algorithm over another one. The paper is concluded in the fourth section where problems for future considerations are also formulated.

## 2 Statistical Tests for the Comparison of Classification Algorithms

Let us assume that we have to classify  $n$  objects into  $K$  disjoint classes using two algorithms, say  $A$  and  $B$ . In this paper we restrict ourselves to the case when the classification algorithm classifies each object to only one of possible classes. We do not impose any restriction on the type of the algorithm used for this purpose. This can be artificial neural network classifier, set of classification rules, vector supporting machine classifier, Bayes naïve classifier or any other algorithm that can be proposed



for this purpose. An expert may act as one of these algorithms. In this case we are able to evaluate the correctness of the classification of each considered object by the second algorithm, as in the case of classical supervised learning. Thus, in this case we can use our statistical test to evaluate the quality of the classification algorithm. However, when we are not able to evaluate the correctness of the classification, we can only compare the performance of considered algorithms. This situation is typical when the algorithms are built using the methodology of unsupervised learning (e.g. using methods of data clustering).

When we compare two classification algorithms using *the same dataset* of  $n$  objects the results of the comparison may be presented in the form of a two-way contingency table, such as Table 1.

**Table 1.** Data for the comparison of algorithms using the same dataset

Alg.A/Alg.B	1	...	$j$	...	$K$	Total A
1	$n_{11}$	...	$n_{1j}$	...	$n_{1K}$	$n_{A1}$
...	...	...	...	...	...	...
$i$	$n_{i1}$	...	$n_{ij}$	...	$n_{iK}$	$n_{Ai}$
...	...	...	...	...	...	...
$K$	$n_{K1}$	...	$n_{Kj}$	...	$n_{KK}$	$n_{AK}$
Total B	$n_{B1}$	...	$n_{Bj}$	...	$n_{BK}$	$n$

By  $n_{ij}$ ,  $i=1,\dots,K$ ;  $j=1,\dots,K$  in this table we denote the number of observations that have been classified by the algorithm  $A$  to the  $i$ th class, and by the algorithm  $B$  to the  $j$ th class. We assume that the results of classification by the algorithm  $A$  are described by the set  $(n_{A1}, n_{A2}, \dots, n_{AK})$ , and that the results of classification by the algorithm  $B$  are described by the set  $(n_{B1}, n_{B2}, \dots, n_{BK})$ . The data can come from classifications performed on a test sample, combined results of cross-validation experiments or classification obtained in the learning process (training sample). However, in the latter case the results are of rather limited interest, as all good classification algorithms perform rather well on training data.

We are interested in the verification of the statistical hypothesis that the probability distributions are such that these sets of data are *strongly dependent*. This strong dependence means that both compared algorithms provide the same or nearly the same results of classification. When one of the compared algorithms is just an expert, the measure of such dependence is also the measure of the correctness of classification. Otherwise, the strength of dependence is the measure of the equivalence of the compared algorithms.

When we assume that the classification by the algorithm  $A$  and the classification by the algorithm  $B$  are *independent* then the data presented in Table 1 are distributed according to the *multiple hypergeometric* distribution. Probability of observing the two-way contingency table  $\{n_{ij}\}$  with the fixed values of marginal observations  $n_{Ai}, n_{Bj}, i = 1, \dots, K, j = 1, \dots, K$  is given by the formula:

$$P(\{N_{ij}\} = \{n_{ij}\}) = \frac{\prod_{i=1}^K n_{Ai}! \prod_{j=1}^K n_{Bj}!}{n! \prod_{i=1}^K \prod_{j=1}^K n_{ij}!} \tag{1}$$

The probability distribution given by (1) can be used for the construction of the test of independence. The general idea of this test, known as Fisher's exact test, is simple. We have to generate all possible contingency tables, such as Table 1, with the fixed margins equal to the margins observed for the considered table  $n_{Ai}, n_{Bj}, i = 1, \dots, K, j = 1, \dots, K$ . For all these tables we have to calculate, using (1), their probabilities. The sum of those probabilities whose values do not exceed the probability of the observed table is equal to the  $p$ -value (significance) of the test. Low values of this characteristics, say less than 0,05 (or 5%), indicate that the observed table does not support the hypothesis of the independence between classifications obtained using both compared algorithms, and thus, supports the alternative hypothesis of dependence.

Despite its simple and intuitive description the implementation of this algorithm is very difficult as the computational volume grows exponentially with the increasing values of  $K$  and  $n$ . Till the publication of the network algorithm in [13] computations were possible only for small tables. This algorithm, presented in the form of the FORTRAN code in [14] allows to compute  $p$ -values of Fisher's exact test for tables with larger values of  $n$ , provided that the table contains many cells with very low (i.e. equal to zero or close to zero) values. Fortunately, this is the case when we analyze good classification algorithms with a low percentage of false classifications.

In the case of large samples with significant percentage of false classifications we can use the well known Pearson's chi-square asymptotic test for independence. The chi-square statistic is given by

$$\chi_I^2 = \sum_{i=1}^K \sum_{j=1}^K \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}}, \quad (2)$$

where

$$\hat{n}_{ij} = \frac{n_{Ai} n_{Bj}}{n} \quad (3)$$

is the expected number of observations in the  $ij$ th cell when both classifications, i.e. by the algorithm A and the algorithm B, are statistically *independent*. When the total number of observations  $n$  is large (greater than 100), and the expected number of observations in every cell is larger than 5, the chi-square statistic, defined by (2), is distributed according to the chi-square distribution with  $(K-1)^2$  degrees of freedom. Thus, the  $p$ -value of this test is computed by solving, with respect to  $p$ , the following equation:

$$\chi_I^2 = \chi_{(K-1)^2, 1-p}^2, \quad (4)$$

where  $\chi_{(K-1)^2, 1-p}^2$  is the quantile of the  $1-p$  order from the chi-square distribution with  $(K-1)^2$  degrees of freedom. When the total number of observations  $n$  is large we can consider the expectations calculated according to (3) as close to the theoretical expected values of observations. Then, we can use the rule proposed in [16] which states that if  $r$  is the number of cells with the expectations less than 5, then the lowest expectation could be as small as  $5r/K^2$ . When the chi-square test of independence is

used for the evaluation of classification algorithms Yarnold’s rule could be very useful in practice.

Let us apply the methodology explained above for the analysis of the classical linear discrimination algorithm (LDA) applied for a well known benchmark test – the famous Fisher’s Iris test. The results of classification using the LDA algorithm implemented in the statistical package STATISTICA and the Iris data set are displayed in Table 2.

**Table 2.** Classification of the Iris data with the LDA algorithm

Expert \ LDA	Iris-Setosa	Iris-Versicolor	Iris-Virginica
Iris-Setosa	50	0	0
Iris-Versicolor	0	48	2
Iris-Virginica	0	1	49

The probability of the observation of this table, when the hypothesis of independence is true, is extremely low (3,1E-65). Thus, the  $p$ -value for Fisher’s exact test of independence in the case of these data is equal to 0. It means that the results of classification provided by the expert are, as expected, strongly dependent. This supports the opinion that the LDA algorithm for this data set is very efficient.

Now, let us consider the application of another algorithm, namely Classification Regression Tree (CRT). The results of the application of this algorithm implemented in the statistical package STATISTICA are presented in Table 3.

**Table 3.** Classification of the Iris data with a CRT algorithm

Expert \ LDA	Iris-Setosa	Iris-Versicolor	Iris-Virginica
Iris-Setosa	50	0	0
Iris-Versicolor	0	48	2
Iris-Virginica	0	4	46

The probability of the observation of this table, when the hypothesis of independence is true, is also extremely low (1,6E-61). Hence, the  $p$ -value for Fisher’s exact test of independence in the case of these data is equal to 0. Therefore, from a statistical point of view both algorithms are fully efficient. We have to note, however, that the difference between the numbers of observed false classification (3 by the LDA algorithm, and 6 by the CRT algorithm) in the case of a relatively small sample (150 observations) may be considered as random.

The Iris data are well separable, and classification algorithms usually perform very well on this benchmark. Let us consider now another example, presented in the paper [4], where number of false classifications, even on a training data set, is quite large. The results of the classification using a proposed in this paper Complete Gradient Clustering Algorithm (CGCA) are presented in Table 4.

**Table 4.** Classification of the wheat kernel data with the CGCA algorithm

Expert \ CGCA	Kama	Rosa	Canadian
Kama	59	2	9
Rosa	3	67	0
Canadian	3	0	67

The application of Fisher’s exact test gives in this case also a extremely low  $p$ -value (0,897E-74) showing the great strength of dependence between the results of classification provided by the expert and the evaluated algorithm.

The numerical examples presented above show that the proposed statistical methodology is computationally demanding, and its results are difficult to interpret. The ratio of observed  $p$ -values provides some information about the superiority of one algorithm over another one, but this interpretation does not have any sound statistical basis.

Consider now the situation when all false classifications are equally important. In this case we can put them together in one class of incorrectly classified objects. Let  $(n_1, n_2, \dots, n_K, n_{K+1})$  be the vector describing the results of the application of the considered classification algorithm. First  $K$  components of this vector represent the numbers of cases of the *correct* classification to  $K$  considered classes. The last component gives the total number of incorrectly classified objects.

Let us assume now that observed values of  $(n_1, n_2, \dots, n_K, n_{K+1})$  represent a *sample* from an unknown multinomial distribution, defined by the probability mass function

$$MB(p_1, \dots, p_K, p_{K+1}) = \frac{n!}{n_1! \dots n_{K+1}!} \prod_{i=1}^{K+1} p_i^{n_i} \tag{5}$$

where  $\sum_{i=1}^{K+1} n_i = n$ , and  $\sum_{i=1}^{K+1} p_i = 1$ , that describes a hypothetical population of objects classified in a similar way to that used for the classification of the considered sample.

Now, let us suppose that we have to compare *two* classification algorithms, whose results of application are given in the form of two vectors  $(n_1, n_2, \dots, n_K, n_{K+1})$ , and  $(m_1, m_2, \dots, m_K, m_{K+1})$ , respectively. First, let us consider the case that both algorithms are compared using *the same set* of observations. Thus, the sample sizes in both cases are equal and both observed vectors are statistically *dependent*. In such case in order to compare the considered algorithms we have to know the results of the classification of each object, and then to use statistical methods devised for the analysis of pair-wise matched data. Theoretically, it is possible if we construct Fisher’s test of independence using three-dimensional contingency table. Taking into account computational problems with classical Fisher’s exact test it seems to be rather impossible to propose a test which compares classification algorithms taking into account their efficiencies.

The situation is different if we want to compare two algorithms without taking into account their efficiencies understood as probabilities of yielding correct classifications. In such a case the comparison can be done relatively easily when the data from a classification experiment performed on the *same* sample are given in the form presented in Table 5.

**Table 5.** Comparison of dependent test data

	Alg.1 -correct	Alg.1 - false
Alg. 2-correct	$k_{11}$	$k_{12}$
Alg. 2 - false	$k_{21}$	$k_{22}$

In this table  $k_{11}$  denotes the number of objects classified correctly by both algorithms,  $k_{12}$  denotes the number of objects classified correctly by the Algorithm 1 but incorrectly by the Algorithm 2,  $k_{21}$  is the number of objects classified correctly by the Algorithm 2 but incorrectly by the Algorithm 1, and  $k_{22}$  is the number of objects classified incorrectly by both algorithms.

Let us notice that the only information about the differences between both algorithms are contained in  $k_{12}$  and  $k_{21}$ . We can verify two hypotheses related to these values. First hypothesis is that the probabilities of incorrect classification that generate observations  $k_{12}$  and  $k_{21}$  are the for both compared algorithms are the same, and this hypothesis is tested against the alternative that they are simply different. In this case we have to apply the so called two-sided statistical test. We may also consider testing this statistical hypothesis against the alternative hypothesis that one particular algorithm is better than a second one. In this case we have to apply the so-called one-sided statistical test.

When both compared probabilities are equal it is known, see e.g. [1] for more information, that the number of incorrect classifications by only one algorithm  $k_{21}$  (or  $k_{12}$ ) is described by the Binomial probability distribution with the parameters  $k=k_{12}+k_{21}$  and  $p=0,5$ . Let us assume now that we observe  $k_{12}^*$  and  $k_{21}^*$  incorrectly classified (only by one algorithm!) objects. The probability of observing  $k_{21}$  false classification given  $k^* = k_{12}^* + k_{12}^*$  can be calculated from the following formula

$$P(k_{21} | k^*) = \binom{k^*}{k_{21}} \left(\frac{1}{2}\right)^{k_{21}} \left(\frac{1}{2}\right)^{k^* - k_{21}} \tag{6}$$

In order to verify the hypothesis of equal probabilities of misclassification we have to calculate, according to (6), the probabilities of all possible pairs  $(k_{21}, k^*)$ . In the case of the two-sided test the sum of those probabilities that do not exceed the probability of the observed pair  $(k_{21} = k_{21}^*, k^*)$  gives the value of the significance (known also as the  $p$ -value) of the tested hypothesis. When this value is greater than 0,05 it is usually assumed that the hypothesis of the equal probabilities should not be rejected. In the case of the one-sided test we consider only these pairs  $(k_{21}, k^*)$  that are less or equally probable that the observed pair  $(k_{21}^*, k^*)$ , and support the one-sided alternative. Thus, the  $p$ -value in the case of the one-sided alternative is smaller than in the case of the two-sided alternative. Hence, it is easier to reject the hypothesis that one algorithm is not worse than the other one than to reject the hypothesis that they are statistically equivalent.

When the number of objects  $k^*$  that are incorrectly classified only by one algorithm is sufficiently large (in practice it is required that the inequality  $k^* > 10$  must be fulfilled) the following statistic

$$T = \frac{(k_{12} - k_{21})^2}{k_{12} + k_{21}} \tag{7}$$

is approximately distributed according to the chi-square distribution with 1 degree of freedom. This statistic is used in the well known McNemar’s test of the homogeneity of proportions for pair-wise matched data.

Let us consider again the example of Fisher’s Iris data. We use this benchmark set for the comparison of two algorithms: LDA (Linear Discrimination Analysis) and CRT (Classification Regression Tree) – both implemented in a popular statistical software such as e.g. STATISTICA. For more information about these algorithms see e.g. [11]. Close examination of the classifications given by both algorithms results in the data presented in Table 6.

**Table 6.** Comparison of algorithms (LDA vs. CRT) – Iris data set

CRTLDA	LDA – correct	LDA - false
CRT - correct	143	1
CRT - false	4	2

The  $p$ -value for these data, computed according to the algorithm given above, is equal to 0,375. Therefore, the obtained statistical data do not let us to reject the hypothesis that the probabilities of incorrect classification are in case of these two algorithms the same despite the fact that the CRT algorithm gives twice as many false classification in comparison to the LDA classifier.

Now, let us use the data that are less separable than the Iris data set. This situation is in the case of wheat kernel data considered in [4]. We will use these test data for the comparison of two algorithms: the Bayesian algorithm proposed in [12] and the classical Quadratic Discrimination Algorithm (QDA) algorithm described in [11]. The results of the comparison are presented in Table 7.

**Table 7.** Comparison of algorithms (Bayes vs. QDA)– Wheat kernels data set

QDA\Bayes	Bayes - correct	Bayes - false
QDA - correct	85	9
QDA - false	5	6

The  $p$ -value in this case is equal to 0,424. Therefore, the obtained statistical data do not let us to reject the hypothesis that the probabilities of incorrect classification are in the case of these two algorithms the same despite the fact that one of the compared algorithms (QDA) seems to be significantly better (nearly 30% lower probability of incorrect classification).

When we do not have an access to individual results of classification we can compare algorithms using *independent* samples described by the multinomial distributions. Let the data be described by (5), and  $\sum_{i=1}^{K+1} n_i = n$  and  $\sum_{i=1}^{K+1} m_i = m$  be the sample sizes which in general, as we compare the classifications of different samples, do not have to be equal. Moreover, note that in the case when one of these algorithms is a perfect classifier (e.g. a domain expert) we have  $n_{K+1} = 0$  (or  $m_{K+1} = 0$ ). If the results of the application of the first algorithm are described by the multinomial distribution  $MB(p_1, \dots, p_K, p_{K+1})$ , and the results of the application of the second algorithm are described by the multinomial distribution  $MB(q_1, \dots, q_K, q_{K+1})$  their performance can be compared by testing the statistical hypothesis

$$H_0 : p_1 = q_1, \dots, p_K = q_K, p_{K+1} = q_{K+1} . \tag{8}$$

To test this hypothesis we may apply the methodology of two-way contingency tables. Test data in the case of the accumulation of all falsely classified object into one (K+1) class are presented in Table 8.

**Table 8.** Independent test data

Alg./Class	1	...	j	...	K	K+1	Total
Alg. 1	$n_{11}$	...	$n_{1j}$	...	$n_{1K}$	$n_{1K+1}$	$N$
Alg. 2	$n_{21}$	...	$n_{2j}$	...	$n_{2K}$	$n_{2K+1}$	$M$
Total	$c_1$	...	$c_j$	...	$c_K$	$c_{K+1}$	$N+M$

When the hypothesis  $H_0$  given by (8) is true, the conditional distribution of observed random vectors  $(n_1, n_2, \dots, n_K, n_{K+1})$ , and  $(m_1, m_2, \dots, m_K, m_{K+1})$ , given the vector of their sum  $(c_1, c_2, \dots, c_K, c_{K+1})$ , is given by the multivariate hypergeometric distribution [5]

$$P(\mathbf{n}; \mathbf{m} | \mathbf{c}, H_0) = \frac{m!n!}{N!} \prod_{i=1}^{K+1} \binom{c_i}{n_i} \tag{9}$$

This probability function is used for the construction of the multivariate generalization of Fisher’s exact test that is used for the verification of (4). Let  $\mathbf{n}^*$ ,  $\mathbf{m}^*$  and  $\mathbf{c}^*$  be the observed data vectors. The  $p$ -value (significance) of the test is computed from the formula [5]

$$(p - value) = \sum_{\Gamma} P(\mathbf{n}, \mathbf{m} | \mathbf{c}^*, H_0), \tag{10}$$

where

$$\Gamma = \left\{ (\mathbf{n}, \mathbf{m}) : P(\mathbf{n}, \mathbf{m} | \mathbf{c}^*, H_0) \leq P(\mathbf{n}^*, \mathbf{m}^* | \mathbf{c}^*, H_0) \right\} \tag{11}$$

The  $p$ -values of this test can be computed by the tools of statistical packages such as SPSS or SAS. However, in the case of many categories and large (or even moderate) samples the computation time may be prohibitively long.

It can be shown that the test of the equality of two sets of multinomial probabilities is formally equivalent to the test of independence of categorical data, considered in the first part of this section. Hence, in the case of sufficiently large sample sizes with all cells having at least 5 observations, for testing (8) one can use Pearson’s chi-square test of independence. These assumptions are usually fulfilled in testing classification algorithms, except for situations when tested data allows building perfect or nearly perfect classifiers. However, in such cases the problem of choice of the best classifiers does not exist.

The  $\chi^2$  statistic in the considered case can be written as

$$\chi^2 = \sum_{i=1}^{K+1} \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i} + \sum_{i=1}^{K+1} \frac{(m_i - \hat{m}_i)^2}{\hat{m}_i} \tag{12}$$

where

$$\hat{n}_i = \frac{nc_i}{N}, \tag{13}$$

and

$$\hat{m}_i = \frac{mc_i}{N}. \tag{14}$$

The  $p$ -value for this test is obtained by solving, with respect to  $p$ , the equation

$$\chi^2 = \chi_{K,1-p}^2, \tag{15}$$

where  $\chi_{K,1-p}^2$  is the quantile of order  $1-p$  in the chi-square distribution with  $K$  degrees of freedom. Also in this case the  $p$ -values of Pearson’s chi-square test of independence can be computed using the tools available in various statistical packages.

In order to illustrate the application of the proposed tests in the evaluation of classification algorithms tested on samples of  $N=100$  objects each which are classified into  $K=3$  classes. Suppose that we want to compare three algorithms A, B, and C, together with a “perfect” algorithm represented by an expert E. All compared algorithms have their basic and ‘improved’ versions indexed by subscripts 1 and 2, respectively. In order to make the evaluation simple we assume that all incorrect (false) classifications are assigned to the additional fourth class. Suppose that the results of this experiment are presented in Table 9.

Algorithms A, B and C in their both versions are characterized by the *same* total percentages of incorrect classification equal to 10% and 5%, respectively. However, the distribution of incorrectly classified objects depends upon the used algorithm. We face this situation when the algorithms are “aimed” at correct classification of chosen classes (e.g. Bayes classifiers).

**Table 9.** Results of an experiment with independent samples

Alg.\Class	1	2	3	4
Expert	20	30	50	0
A <sub>1</sub>	18	27	45	10
A <sub>2</sub>	19	29	47	5
B <sub>1</sub>	10	30	50	10
B <sub>2</sub>	15	30	50	5
C <sub>1</sub>	20	30	40	10
C <sub>2</sub>	20	30	45	5

In the case of the algorithm A incorrectly classified objects are distributed proportionally to the actual sizes of classes. For the algorithm B all incorrectly classified objects are assigned to the class with the lowest number of actual observations. Finally, in the case of the algorithm C all incorrectly classified objects are assigned to the class with the highest number of actual observations.



In Table 10 we present the  $p$ -values of both considered tests when the performance of each classification algorithm is compared to the classification given by the expert.

**Table 10.** Comparison of different algorithms with the expert

	Fisher's	Chi-square
A <sub>1</sub> vs. E	0,008	0,015
B <sub>1</sub> vs. E	0,002	0,004
C <sub>1</sub> vs. E	0,006	0,011
A <sub>2</sub> vs. E	0,177	0,162
B <sub>2</sub> vs. E	0,132	0,126
C <sub>2</sub> vs. E	0,165	0,154

In the case of basic versions of all algorithms the results of classification are statistically significantly different than the classification provided by the expert. The worse classification is provided by the algorithm A. In the sample analyzed by this algorithm all falsely classified objects are evenly distributed over all classes. The best performance is observed in case of the algorithm B characterized by the largest percentage-wise differences between levels of the accuracy of classification in different classes. In the case of the 'improved' versions of the considered algorithms the data do not let us to reject the hypothesis that the results of classification are statistically equivalent to the results of classification provided by the expert.

Now, let us apply the proposed methodology for the comparison of basic and 'improved' versions of our hypothetical algorithms. The results of this comparison are presented in Table 11.

**Table 11.** Comparison of different versions of algorithms

	Fisher's	Chi-square
A <sub>1</sub> vs. A <sub>2</sub>	0,640	0,613
B <sub>1</sub> vs. B <sub>2</sub>	0,470	0,446
C <sub>1</sub> vs. C <sub>2</sub>	0,599	0,581

The results of this comparison are somewhat unexpected for a non-statistician. Despite seemingly large improvement (reduction of the percentage of incorrect classifications from 10% to 5%) the compared results statistically do not differ. The reason for this behavior is, of course, a small sample size. What is also interesting that the difference is the least significant (the highest  $p$ -value in the test of equality) in the case of evenly distributed misclassifications. The lowest  $p$ -value (but still very high using statistical standards) is for the case of algorithm B which assigns all incorrectly classified objects to the class with the smallest number of observations.

Now, let us consider an example of the application of this methodology to real data. Suppose, that we have been provided with two algorithms for the classification of vehicle silhouettes data (data provided by Turing Institute, Glasgow, and available at the UCI web-site). One of these algorithms implements the Bayesian algorithm proposed in [12], and the second one implements a classical CRT algorithm described in [3]. The algorithms have been tested on two *independent* samples, and the results of this comparison are presented in Table 12.

**Table 12.** Comparison of algorithms - Vehicle Silhouettes data set

Alg.\Class	1	2	3	4	5
Bayes	55	48	112	90	141
CRT	46	55	86	84	175

The  $p$ -value obtained as the solution of (15) for these data is equal to 0,079. According to the classical statistical approach this result does not let us claim that the Bayes algorithm is better than the CRT. Note however, that similar results obtained on the *same* sample would probably indicate the superiority of the Bayes algorithm.

### 3 Possibilistic Evaluation of Test Results

In the previous section we have proposed statistical tests for the evaluation of classification procedures. The results of the proposed test procedures have been expressed in terms of significance, known also as the test volume or the  $p$ -value. Examples given in this section show that the results of statistical tests interpreted in a traditional way are not well suited for finding if one classification algorithm is better than the other one. Therefore, there is a need to present an additional indicator that can be used to show to what extent one algorithm is better than the other one despite the fact that they are statistically equivalent. This goal can be achieved using the methodology proposed in the *theory of possibility*. In the papers [8] and [9] the possibilistic interpretation of statistical tests has been proposed. This interpretation gives a decision maker the evaluation of test’s result using notions of *possibility* or *necessity* of making certain decisions.

Statistical decision problems are described by setting a certain hypothesis  $H$  (usually called the null hypothesis), and an alternative hypothesis  $K$ . In the context of decision-making we usually choose this hypothesis which is better supported by statistical data. Hryniewicz [9] proposes to consider these two hypotheses separately. Suppose that the significance of  $H$  is given by the  $p$ -value of the test, and is equal to  $p_H$ . The value of  $p_H$  shows to what extent the statistical evidence supports the null hypothesis.

In [9] it was proposed to evaluate the null hypothesis  $H$  by a fuzzy set  $\tilde{H}$  with the following membership function

$$\mu_H(x) = \begin{cases} \min[1, 2p_H] & x = 0 \\ \min[1, 2(1 - p_H)] & x = 1 \end{cases} \quad (16)$$

This membership function may be interpreted as a *possibility distribution* of the truth of  $H$ . If  $\mu_H(1)=1$  holds, it means that it is quite *plausible* that the considered hypothesis is not true. On the other hand, when  $\mu_H(0)=1$ , we would not be surprised if  $H$  were true.

The same can be done for the alternative hypothesis  $K$ . The statistical test of this hypothesis may be described by another  $p$ -value, denoted by  $p_K$ . When  $K = \text{not } H$  we have  $p_K = 1 - p_H$ . However, in a general setting this equality usually does not hold. The

alternative hypothesis  $K$  is now represented by a fuzzy set  $\tilde{K}$  with the following membership function

$$\mu_K(x) = \begin{cases} \min[1, 2p_K] & x = 0 \\ \min[1, 2(1-p_K)] & x = 1 \end{cases} \quad (17)$$

In order to choose an appropriate decision, i.e. to choose either  $H$  or  $K$ , Hryniewicz [9] proposes to use three measures of possibility defined in [6].

For two fuzzy sets  $\tilde{A}$  and  $\tilde{B}$ , described by their membership functions  $\mu_A(x)$  and  $\mu_B(y)$ , respectively, the *Possibility of Dominance (PD)* measure is defined in [6] in the following way

$$PD(\tilde{A} \geq \tilde{B}) = \sup_{x, y: x \geq y} \min[\mu_A(x), \mu_B(y)] \quad (18)$$

The second index is called the *Possibility of Strict Dominance (PSD)*, and for two fuzzy sets  $\tilde{A}$  and  $\tilde{B}$  is given by the expression

$$PSD(\tilde{A} > \tilde{B}) = \sup_x \left\{ \inf_{y: x \leq y} [\min(\mu_A(x), 1 - \mu_B(y))] \right\} \quad (19)$$

Positive, but smaller than 1, values of this index indicate certain weak evidence that  $\tilde{A}$  strictly dominates  $\tilde{B}$ .

Third measure is named the *Necessity of Strict Dominance*, and for two fuzzy sets  $\tilde{A}$  and  $\tilde{B}$  has been defined in [6] as:

$$NSD(\tilde{A} > \tilde{B}) = 1 - \sup_{x, y: x \leq y} [\min(\mu_A(x), \mu_B(y))] \quad (20)$$

The *NSD* index represents a *necessity* that the fuzzy set  $\tilde{A}$  strictly dominates the set  $\tilde{B}$ .

In the considered statistical problem of testing a hypothesis  $H$  against an alternative  $K$  these indices have been calculated in [8], and are given by the following formulae

$$PD(\tilde{H} \geq \tilde{K}) = \max[\mu_H(0), \mu_K(1)], \quad (21)$$

$$PSD(\tilde{H} > \tilde{K}) = \min[\mu_H(0), 1 - \mu_K(0)], \quad (22)$$

$$NSD(\tilde{H} > \tilde{K}) = 1 - \max[\mu_H(1), \mu_K(0)]. \quad (23)$$

The value of *PD* represents the *possibility* that according to the observed statistical data the choice of the null hypothesis is not a worse decision than choosing its alternative. The value of *PSD* gives the measure of *possibility* that the data support

rather the null hypothesis than its alternative. Finally, the value of  $NSD$  gives the measure of *necessity* that the data support the null hypothesis rather than its alternative.

It has been proved that

$$PD \geq PSD \geq NSD. \quad (24)$$

It means that according to the practical situation we can choose the appropriate measure of the correctness of our decision. If the choice between  $H$  and  $K$  leads to serious consequences we should choose the  $NSD$  measure. In such a case  $p_H > 0,5$  is required to have  $NSD > 0$ . When these consequences are not so serious we may choose the  $PSD$  measure. Finally, the  $PD$  measure, which is always positive, gives us the information of the possibility that choosing  $H$  over  $K$  is not a completely wrong decision.

In some cases considered in this paper the alternative hypothesis has been formulated as the complement of the null hypothesis, Thus, we have the equality  $p_K = 1 - p_H$ . In this case we have

$$PD(\tilde{H} \geq \tilde{K}) = \mu_H(0) = \min(1, 2p_H), \quad (25)$$

$$PSD(\tilde{H} > \tilde{K}) = NSD(\tilde{H} > \tilde{K}) = \max(2p_H - 1, 0). \quad (26)$$

Let us apply these results for the comparison of different algorithms using the test result presented in Table 6 for the comparison of the LDA and CRT algorithms used for the classification of the Iris data. For this statistical test we have  $p_H = 0,375$ , and  $p_K = 0,625$ . Hence, we have  $PD = 0,750$ , and  $PSD = NSD = 0$ . Therefore, there is only a certain possibility that these two algorithms are equivalent, but the measure of the necessity of such claim is equal to zero.

The possibilistic comparisons are not necessary when null and alternative hypotheses are, as in the cases considered above, complementary. In such case strong evidence in favor of the null hypothesis means automatically weak support of its complementary alternative.

## 4 Conclusions

In the paper we have considered the problem of the evaluation and comparison of different classification algorithms. For this purpose we have applied the methodology of statistical tests for the multinomial distribution. We restricted our attention to the case of the supervised classification when an external 'expert' evaluates the correctness of classification. The results of the proposed statistical tests are interpreted using the possibilistic approach introduced in [9]. The results presented in this paper can be extended to the case of imprecise data. In this case the applicability of the proposed possibilistic measures is even much stronger when we omit, for example, the assumption that there exists an 'expert' who indicates only one 'true' class. In such cases we have to use the methodology of fuzzy statistics, whose

overview can be found e.g. in [7]. We will face such problems, for example, when we will adapt the methodology presented in this paper for the case of the evaluation of fuzzy classifiers.

**Acknowledgements.** The author expresses his thanks to Dr. P. A. Kowalski and Dr. S. Łukasik for providing solutions for some practical examples of classification problems.

## References

1. Agresti, A.: *Categorical Data Analysis*, 2nd edn. J. Wiley, Hoboken (2006)
2. Berthold, M., Hand, D.J. (eds.): *Intelligent Data Analysis. An Introduction*, 2nd edn. Springer, Berlin (2007)
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees*. CRC Press, Boca Raton (1984)
4. Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P.A., Łukasik, S., Żak, S.: Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images. In: Piętka, E., Kawa, J. (eds.) *Information Technologies in Biomedicine. AISC*, vol. 69, pp. 15–24. Springer, Heidelberg (2010)
5. Desu, M.M., Raghavarao, D.: *Nonparametric Statistical Methods for Complete and Censored Data*. Chapman & Hall, Boca Raton (2004)
6. Dubois, D., Prade, H.: Ranking Fuzzy Numbers in the Setting of Possibility Theory. *Information Science* 30, 183–224 (1983)
7. Gil, M.A., Hryniewicz, O.: Statistics with Imprecise Data. In: Meyers, R.A. (ed.) *Encyclopedia of Complexity and Systems Science*, pp. 8679–8690. Springer, Heidelberg (2009)
8. Hryniewicz, O.: Possibilistic Interpretation of the Results of Statistical Tests. In: *Proceedings of Eight International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2000, Madrid*, pp. 215–219 (2000)
9. Hryniewicz, O.: Possibilistic decisions and fuzzy statistical tests. *Fuzzy Sets and Systems* 157, 2665–2673 (2006)
10. Hryniewicz, O.: Possibilistic methodology for the evaluation of classification algorithms. In: *Proceedings of the 6th International Conference on Software and data Technology, ICSoft 2011, Seville (July 2011)*
11. Krzanowski, W.J.: *Principles of Multivariate Analysis: A User's Perspective*. Oxford University Press, New York (1988)
12. Kulczycki, P., Kowalski, P.A.: Bayes classification of imprecise information of interval type. *Control and Cybernetics* 40, 101–123 (2011)
13. Mehta, C.R., Patel, N.R.: Network algorithm for performing Fisher's exact test in  $r \times c$  contingency tables. *Journ. Amer. Stat. Assoc.* 78, 427–434 (1983)
14. Mehta, C.R., Patel, N.R.: ALGORITHM 643: FEXACT: a FORTRAN subroutine for Fisher's exact test on unordered  $r \times c$  contingency tables. *ACM Transactions on Mathematical Software (TOMS)* 12, 154–161 (1986)
15. Nisbet, R., Elder, J., Miner, G.: *Statistical Analysis and Data Mining. Applications*. Elsevier Inc., Amsterdam (2009)
16. Yarnold, J.K.: The Minimum Expectation in  $X^2$  Goodness of fit test and the Accuracy of Approximations for the Null Distribution. *Journ. Amer. Stat. Assoc.* 70, 864–886 (1970)

# What Else Can Be Extracted from Ontologies? Influence Rules

Barbara Furletti and Franco Turini

Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy  
{furletti,turini}@di.unipi.it

**Abstract.** A method for extracting new implicit knowledge from ontologies by using an inductive/deductive approach is presented. The new extracted knowledge takes the form of If-Then rules annotated with a weight. Such rules, termed Influence Rules, specify how the values of the properties bound to a collection of concepts may influence the values of the properties of another concept. The technique, that combines data mining and link analysis, is completely general and applicable to whatever domain. The paper reports the methods and the algorithms supporting the process of mining the rules out of the ontology, and discusses its application to real data from the economic field.

**Keywords:** Ontology mining, Knowledge discovery, If-Then rules.

## 1 Introduction

Knowledge extraction from databases is a consolidated practice that continues to evolve in parallel with the new data management systems. It is based not only on querying systems, but above all, on complex reasoning tools. Today, with the coming of the Web 2.0 and the semantic web, new methods for representing, storing and sharing information are going to replace the traditional systems. Roughly speaking, ontologies “could substitute” in many applications the Data Bases (DBs). Consequently, the interest is moving toward the research of new methods for handling these structures and to efficiently obtain information from them besides what is obtained by using the traditional reasoning systems.

In this paper we aim at contributing to this topic by handling the problem of extracting interesting and implicit knowledge from ontologies, in a novel way with respect to the traditional reasoners methods. By getting hints from the semantic web and data mining environments, we give a Bayesian interpretation to the relationships that already exist in an ontology in order to return a set of weighted If-Then rules, that we refer to as Influence Rules (IRs).

The idea is to split the extraction process in two separate phases by exploiting the ontology peculiarity of keeping metadata (the schema) and data (the instances) separate. The deductive process draws inference from the ontology structure, both concepts and properties, by applying link analysis techniques and producing a sort of implications (rules schemas) in which only the most important concepts are involved. Then an inductive process, implemented by a data mining algorithm, explores the ontology instances for enriching the implications and building the final rules.

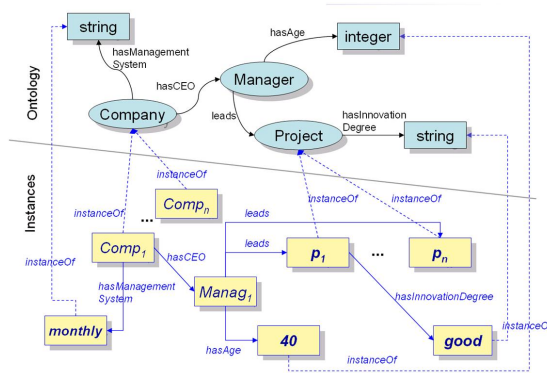


Fig. 1. Fragment of an ontology schema and its instances

For example, let us suppose we have a fragment of ontology as depicted in Fig. 1 that describes companies and the business environment. *Company*, *Manager* and *Project* are concepts, continuous arrows represent properties of the ontologies while the dotted ones are used for connecting instances to the classes they belong to. Starting from this ontology and the corresponding instances we are able, at the end of the process, to produce IRs as the following one:

$$\text{Manager.hasAge} < 45 \xrightarrow{w=0.80} \text{Project.hasInnovationDegree} = \text{good}$$

Both the premise ( $\text{Manager.hasAge} < 45$ ) and the consequence ( $\text{Project.hasInnovationDegree} = \text{good}$ ) are expressions binding the data-type property of a class to a specific value, while the weight ( $w$ ) measures the strength of the influence. This rule must be read as:

“In 80% of the cases, whenever a manager of a company is less than 45 years old, then the project he manages has a good degree of innovation”.

What we want to prove, besides the correctness and feasibility<sup>1</sup> of the project, is that the approach allows us to extract “higher level” rules w.r.t. classical knowledge discovery techniques. In fact, ontology metadata gives a general view of the domain of interest and supplies information about all the elements apart from the fact that they are included as instances in the collected data. The technique is completely general and applicable to every domain. Since the output is a set of “standard” If-Then rules, it can be used to integrate existing knowledge or for supporting any other data mining process.

The paper includes the following sections:

Sect. 2 proposes some related works that try to combine ontologies and data mining in different ways.

Sect. 3 gives a short overview of the technical background about theories and algorithms used in the rest of the paper.

Sect. 4 is the core section in which the Ontology Miner strategy is described.

Sect. 5 shows a case study where our strategy is applied to an actual problem.

<sup>1</sup> The term feasibility has to be intended as the “capability of being done”.

Before concluding this paper, in Sect. 6 we present the new version of the system and some comments about an experiment. Sect. 7 contains the conclusions and discusses some future promising work.

## 2 Ontologies and Data Mining

When speaking about ontologies and data mining (DM), we enter into a domain in which DM techniques and domain ontologies are either combined for improving existing knowledge discovery tools and processes or for supporting decision systems. Ontologies and DM are related in different ways depending on the perspective from which the two field are viewed: is the ontology that improves DM or is DM that operates on ontologies?

Actually, both the perspectives are interesting and three significant research lines can be identified: 1) using ontologies for driving DM; 2) using DM for building ontologies; 3) using ontologies for describing DM processes.

Great efforts are currently spent by researchers in these fields, for example Geller and colleagues [12] describe the use of taxonomies for improving the results of association rule mining. The goal is to produce association rules with higher support from a large set of tuples about demographic and personal interest information. Since the collection of people interests tends to be too abstract for actual applications, they use a hierarchy of concepts for raising data instances to higher levels during a pre-processing step, before running the DM algorithm.

A similar approach has been described in [3] where an ontology in the domain of super market products is used for extracting constraint-based multi-level association rules. In this case the use of an actual ontology (instead of a simple taxonomy) permits the definition of constraints and the use of concepts at different levels of abstraction. In this case the objective is to drive the extraction of rules that fit the user request and need and identify possible target items for seasonal promotions.

On the other hand, since the construction of an ontology is a complex and creative work for the domain experts, DM techniques are often of great help. See for example [9], where the Quinlan's C4.5 algorithm is used for building an ontology starting from the generated decision tree. The ontology is constructed by means of a mapping function from the tree elements: root node, internal nodes and decision branches are mapped into OWL classes, while the leaves (which permit the identification of the association rules) are coded as individuals.

A more structured work is presented in [15], where the authors describe how to enrich an existing seed ontology by using text mining techniques, especially by mining the domain specific texts and glossaries/dictionaries in order to find groups of concepts/terms which are related to each other. Even if the extraction of new concepts or instances from text is automatic, the enrichment of the seed ontology is manually done by the experts. The advantage here is the discovering of many important concepts and interesting relationships directly from the data in an automatic way.

Other contributions in this field are described in [8] and [20].

In [8], the authors describe the implementation of an unsupervised system that combines a syntactic parsing, collocation extraction and selectional restriction learning.



The system, applied to a set of data (in this case to a molecular biology corpus of data), generates a list of labeled binary relations between pairs of ontology concepts. They demonstrate that the system can be easily applied in text mining and ontology building applications.

In [8] a method is sketched for extending existing domain ontologies (or for semi-automatic generating ontologies) on the basis of heuristic rules applied to the result of a multi-layered processing of textual documents. The rules, extracted by using essentially statistical methods, are used for deriving ontology classes from linguistic annotation. The new classes can be added to already existing ontologies or can be used as starting point for a new ontology.

Ontologies are frequently employed also in context-aware systems. As for example in [18], they are used for describing both contexts and the DM process in a dynamic way. In particular the authors split the context aware DM into two parts: the representation of the contexts through the ontology and a framework which is able to query the ontology, invoke the mining processes and coordinate them according to the ontology design.

In the light of the above classification, our work can only partially be seen as a contribution to the line one, because what we do is to move from Knowledge Discovery in Databases to Knowledge Discovery in Ontologies by using a combination of DM and Link analysis methods. Indeed, the analysis of the T-Box of an ontology is used to prepare the process of actual mining out of the A-Box (the instances).

### 3 Technical Background

Our methods combines in a novel way link analysis and DM techniques in order to extract knowledge from ontologies. In this section we introduce the link analysis method we customized and the corresponding extension to the ontology domain. For what it concerns the DM, we used PATTERNIST, a pattern discovery algorithm developed by colleagues at the CNR in Pisa. PATTERNIST is the result of a research activity that has now come to the implementation of a more sophisticated (and documented) system: ConQueSt [7].

#### 3.1 Link Analysis

In this paper we exploit the peculiarities of HITS (Hypertext Induced Topic Selection) [13], the Kleinberg's algorithm for ranking web pages, to provide a sort of "authority measure" to the ontology concepts. HITS rates web pages based on two evaluation concepts: authority and hub. The authority estimates the content value of the page, while the hub estimates the value of its links to other pages. In other words, a hub is a page with outgoing links and authority is a page with incoming links. Kleinberg observed that there exists a certain natural type of balance between hubs and authorities in the web graph defined by the hyperlinks, and that this fact could be exploited for discovering both types of pages simultaneously.

As shown in the Pseudocode 1, HITS works as an iterative algorithm applied to the subgraph  $G_\sigma$  of the web graph, derived from a sort of text matching procedure (for further details see the procedure Subgraph in [13]) of the query terms  $\sigma$  in the search

topic. For this reason it is query-dependent. The core of the algorithm starts from  $G_\sigma$  and computes hub ( $y^{<p>}$ ) and authority ( $x^{<p>}$ ) weights by using an iterative procedure qualified to mutually reinforce the values. It becomes natural to express the mutually reinforcing relationship between hubs and authorities, as: “If  $p$  points to many pages with high  $x$ -values, then it should receive a large  $y$ -value, and if  $p$  is pointed to by many pages with large  $y$ -values, then it should receive a large  $x$ -value”.  $\mathcal{I}$  and  $\mathcal{O}$  operations have been defined for updating the weights.

$\mathcal{I}$  updates the authority  $x$ -weights as:

$$\mathcal{I} : x^{<p>} \leftarrow \sum_{q:(q,p) \in E} y^{<q>}$$

$\mathcal{O}$  updates the hub  $y$ -weights as:

$$\mathcal{O} : y^{<p>} \leftarrow \sum_{q:(p,q) \in E} x^{<q>}$$

Since the two operations are mutually recursive, a fixed point is needed for guaranteeing the termination of the computation. Even if the number  $k$  of iterations is a parameter of the algorithm, it is proven that, with arbitrarily large values of  $k$ , the sequences of vectors  $x_1, x_2, \dots, x_k$  and  $y_1, y_2, \dots, y_k$  converge to the fixed points  $x^*$  and  $y^*$  (Theorem 3.1 in [13]).

As one can guess, and as it happens for the main information retrieval methods, linear algebra supplies “tools” of support for formalizations and proofs.

### Pseudocode 1: HITS

#### Iterate( $G_\sigma, k$ )

$G_\sigma$ : a collection of  $n$  linked pages.

$k$ : a natural number.

$z$ : the vector  $(1, 1, 1, \dots, 1) \in \mathbb{R}^n$ .

Set  $x_0 := z$ ; Set  $y_0 := z$ ;

For  $i = 1, 2, \dots, k$

    Apply the  $\mathcal{I}$  operation to  $(x_{i-1}, y_{i-1})$ , obtaining new  $x$ -weights  $x'_i$ .

    Apply the  $\mathcal{O}$  operation to  $(x'_i, y_{i-1})$ , obtaining new  $y$ -weights  $y'_i$ .

    Normalize  $x'_i$ , obtaining  $x_i$ . Normalize  $y'_i$ , obtaining  $y_i$ .

End

Return  $(x_k, y_k)$ .

First, it is possible to represent the graph  $G_\sigma$  in matrix form with the help of an adjacency matrix  $A$ . Then, one can easily observe that the iterative and mutual call of  $\mathcal{I}$  and  $\mathcal{O}$  can be (re)written as:

$$\begin{aligned} x_i &= A^T y_{i-1} \\ y_i &= A x_i \end{aligned} \tag{1}$$

Stated that, it is easy to trace the computation of  $x^*$  and  $y^*$  back to the mathematical computation of the principal eigenvectors of a matrix  $A^T A$  and  $A A^T$ , respectively. From [1] after  $k$  iterations, we obtain

$$\begin{aligned} x^{(k)} &= (A^T A)^{(k-1)} A^T \mathbf{u} \\ y^{(k)} &= (A A^T)^{(k)} \mathbf{u} \end{aligned} \tag{2}$$

where  $\mathbf{u}$  is the initial seed vector for  $x$  and  $y$ . Equation 2 is the recursive formula for computing the authority and hub vectors at a certain iteration.

For our purposes we customized the HITS algorithm. A short description of HITSx-ONTO algorithm is presented in following Sect. 3.2

### 3.2 HITSxONTO Algorithm

HITSxONTO [10,11], the core algorithm, is the customized version of HITS for handling ontologies. Like HITS, it is based on the concepts of authority and hubness, and its purpose is to measure the importance of the ontology concepts, basing only on the ontology structure (the TBox). In other words, it tries to deduce which concepts can be considered particularly “important” (authorities) and which ones give a particular importance to other concepts (hubs). In this context we are interested in concepts, object properties and in the *isA* relation. This last element is used for constructing the matrix associated to the ontology that points out direct, indirect and hidden connections. The datatype properties, instead, are not relevant in the ranking procedure.

The main algorithm variant w.r.t. HITS concerns the pre-processing phase, that is the preparation of the input and the general adaptation to the ontology. Passing from the web to the ontology environment we adopt the following association: an ontology concept is seen as a web page, and an object property is seen as a hyperlink. HITSxONTO is iterative as HITS, and follows the same core steps.

## 4 Ontology Mining Strategy

As introduced in Sect. 1, the objective of this method is to extract hidden information from an ontology by operating on the structure and on the instances, separately. The strategy (sketched in Fig. 2 and detailed below) is composed by four main steps, each one dedicated to a particular phase of the extraction.

### [Step 1] Identification of the Concepts

This step consists in the analysis of the ontology schema and the extraction of the most relevant concepts.

For the extraction, we exploit the possibility of representing the ontology as a graph with its associated Adjacency Matrix (AM). The AM points out the existence of a direct link between two concepts. Starting from the AM and exploiting the ontology hierarchical structure (defined by the *isA* property) we compute a Weighted Adjacency Matrix (WAM). It is an  $n \times n$  matrix where each entry  $w_{ij}$  has the following meaning:

$$w_{ij} = \begin{cases} k & \text{if } k \text{ edges from } i \text{ to } j \text{ exist} \\ 0 & \text{otherwise} \end{cases}$$

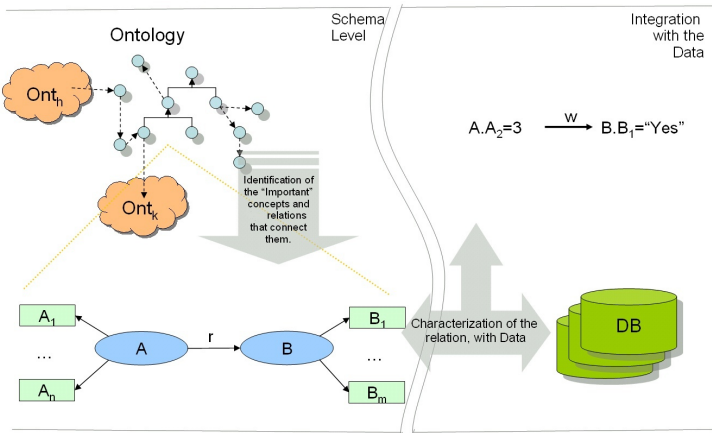


Fig. 2. Steps of analysis

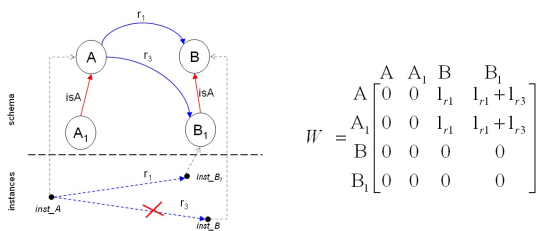
In this matrix we store multiple and hidden connections between concepts that is, the connections among sub-concepts, or parent concepts and sub-concepts that are not directly defined by an explicit link. In other words, we refer to the connections that exist but that are not explicitly represented by an arc in the ontology-graph. The following Example 1 shows a typical case, while the other cases of inheritance are described in [10,11].

**Example 1. Hidden Connections**

Consider the fragment of ontology depicted in Fig. 3:  $A$  and  $B$  are main concepts, while  $A_1$  and  $B_1$  are sub-concepts of  $A$  and  $B$ , respectively.  $r_1$  and  $r_3$  are object properties and the arrows labelled with  $isA$  identify the hierarchy. Thanks to these last connections,  $A_1$  inherits from  $A$  the status of being domain of the properties  $r_1$  and  $r_3$ , while  $B_1$  inherits from  $B$  the status of range of the property  $r_1$ .

It is easy to see that  $A$  is connected to  $B$  and  $B_1$  thanks to direct links ( $r_1$  and  $r_3$ ), but  $A$  has actually a “double” connection with  $B_1$ : one thanks to the direct link  $r_3$  and the other induced by  $r_1$  and the inheritance property.  $A_1$  has no physical connections with other concepts, nevertheless it inherits from  $A$  a simple connection to  $B$  and a double connection to  $B_1$ . Instead,  $B$  does not inherit the range status of  $B_1$  induced by  $r_3$ . In fact, given instances  $inst\_A \in A$  and  $inst\_B \in B$ , they cannot be connected by means of  $r_3$ . The associated WAM  $W$  highlights, for each concept, the number of direct and hidden connections. Since  $isA$  is a hierarchic relation and not an object property, both the  $[A_1, A]$  and  $[B_1, B]$  matrix entries are set to 0. As stated before, the contribution of this relation is used for the identification of the hidden connections. □

In order to extract the relevant concepts, we analyse only the schema of the ontology. The idea is to adopt a link analysis method as the one used in the semantic web environment. While HITS works with web pages and hyperlinks, HITSxONTO works on concepts and object properties. Running HITSxONTO with the WAM as input, we obtain two lists of concepts, ranked on authority and hub principles.



$$W = \begin{matrix} & \begin{matrix} A & A_1 & B & B_1 \end{matrix} \\ \begin{matrix} A \\ A_1 \\ B \\ B_1 \end{matrix} & \begin{bmatrix} 0 & 0 & 1_{r1} & 1_{r1} + 1_{r3} \\ 0 & 0 & 1_{r1} & 1_{r1} + 1_{r3} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Fig. 3. Hidden connections

The most relevant concepts are those that exceed the thresholds for acceptance fixed by the user. Since the threshold strongly depends on the ontology size and connectivity, it has to be empirically fixed.

[Step 2] Influence Rule Schema Building

In this step we construct the schemas of the rules, that is we identify the implicant and the implicated concepts, and the direction of the implication. Each rule schema is created by using the potential implicant concepts, and connecting them with the potential implicated concepts reachable directly or indirectly via object properties. An IR Schemas has the following format:

$$\langle \text{Implicant} \rangle \rightarrow \langle \text{Implicated} \rangle$$

where, Implicant is a concept belonging to the hub-set of concepts and Implicated is a concept belonging to the authority-set of concepts. The following Example 2 clarifies the point.

Example 2. Building the IRs Schema

Suppose to have an ontology that describes companies and the economic environment, and suppose to obtain, from Step 1, the following two lists of candidates concepts:

**Implicant Set** = {ManagementTeam, Company, ...}

**Implicated Set** = {CapitalizationStrategy, DiversificationOfProduction, LevelOfCompetition, ...}

The Implicant and the Implicated sets are composed by concepts that obtained a hub value and an authority value greater than the fixed thresholds respectively. Let us also suppose that in the ontology a connection (a direct object property, an inherited object property or an indirect path of object properties) from Company to LevelOfCompetition exists. Under these hypothesis the following new schema can be built:

$$\text{Company} \rightarrow \text{LevelOfCompetition}$$

This schema is the starting point for the construction of IRs where the concept LevelOfCompetition depends on the concept Company. The characterization of the schema is realized by associating the appropriate attributes defined as datatype properties of the concept in the ontology. □

<sup>2</sup> The appropriate attributes are determined by adopting a particular strategy that uses a DM method on the ontology instances, as described in step 3.

**[Step 3] Characterization of the Influence Rules Schemas**

In this step we create the IRs starting from the schemas built in the previous step. In particular we associate the appropriate attributes to the concepts that form the schema, and a weight for the implication that identifies the strength of the rules.

To do that, we analyse the ontology instances associated to the set of concepts which the domain of interest is composed of, and we extract the frequent items by using the algorithm PATTERNIST cited at the beginning of Sect. 3. The frequent items give us three important information:

1. The pairs of <concept.attribute> that appear together more frequently in the set of instances.
2. The values associated to the attributes.
3. The support of the frequent item sets, that corresponds to the percentage of the instances that include all items in the premise and consequence in the rule.

We then collect, from the frequent itemsets, the values and the weights for the Influence Rules schemas.

It is important to notice that we consider the support as the appropriate measure for weighting the rules. Other measures, like the confidence, could be a refinement in specific fields, although the support remains the more intuitive measure. Example 3 clarifies the point.

*Example 3. Characterizing the IRs Schema*

Starting from the result of the previous example 2, let us suppose that the concepts involved in the schema have the datatype properties reported in Tab. 1. In this step 3, we run PATTERNIST on the set of instances of the ontology under analysis.

**Table 1.** Description of the datatype properties associated to the concepts of the example

Concept	Datatype Prop.	Type	Options
Company	hasName	String	–
	hasDimension	Enumerated	{Small, Medium, Big}
	hasFoundationYear	Integer	–
LevelOfCompetition	hasLevel	Enumerated	{Low, Medium, High}
	hasDescription	String	–
	hasType	Enumerated	{TypeA, TypeB}

The result is a set of frequent items. Let us suppose that the frequent items are the following two:

**FI1.** {LevelOfCompetition.hasType = TypeA, Company.hasFoundationYear = 1989} (supp=0.6)

**FI2.** {LevelOfCompetition.hasLevel = High, Company.hasDimension = Big} (supp=0.8)

Merging FI1 and FI2 according to the schemas extracted in step 2 we obtain the following two influence rules.

**IR1.** Company.hasFoundationYear = 1989  $\xrightarrow{w=0.6}$  LevelOfCompetition.hasType = TypeA

**IR2.**  $\text{Company.hasDimension} = \text{Big} \xrightarrow{w=0.8}$   
 $\text{LevelOfCompetition.hasLevel} = \text{High}$

The rules can be read respectively as:

*“In 60% of the cases, if the company has been founded in 1989 than its level of competition is of TypeA”, and*

*“In 80% of the cases, if the company is big than its level of competition is high”. □*

#### [Step 4] Validation

The Validation is needed to guarantee that the IRs are consistent and do not conflict with each other. The best way for validating the rules is to ask a domain expert, nevertheless some *ad-hoc* procedures can be implemented with reference to the domain under analysis and the foreseeable use.

The first two steps are essentially deductive, they are a sort of “top-down” approach that starts from the theory and tries to find a model. The third one is an inductive step, a sort of “bottom-up” approach; we move from the observations (the instances) to the results (the IRs).

The methodology we propose can be employed in different DM or non-DM applications that make use of additional information in the form of rules, or for enriching pre-existing knowledge repository and structures [1].

To complete the discussion, in the next section we show an actual application that uses the IRs in another DM process.

## 5 Case Study

The IRs extraction process described in Sect. 4 has been used on both real data and ontologies in the European project MUSING [14]. MUSING, “**M**ulti-industry, **S**emantic-based next generation business **I**ntelli**G**ence” aims at developing a new generation of Business Intelligence (BI) tools and modules based on semantic knowledge and content systems. It integrates Semantic Web and Human Language technologies and combines declarative rule-based methods and statistical approaches for enhancing the technological foundations of knowledge acquisition and reasoning in BI applications.

One of the services developed during the project is the Online Self Assessment. By analysing the answers to a questionnaire that describes the economic plan of a company, the tool supplies an evaluation of the quality of the company and of the credit worthiness. The system is based on a predictive model that uses both historical and external knowledge provided by an expert in the domain. The predictive model is implemented by using YaDT-DRb [4], a variant of the famous Quinlan’s C4.5 [17] algorithm, modified for using the external knowledge. As usual for this kind of algorithms, the historical data are used for constructing and training the classification models. The external knowledge instead, is new data-independent knowledge provided by an expert and used for integrating the training set and for driving the construction of the models. This technique is documented in our previous works [21]. The new information is provided in form of If-Then rules that we call Expert Rules (ERs).

In the project, data and metadata are described and stored by using a set of ontologies.

Starting from this scenario, the extraction of IRs out of an ontology is applied to the MUSING ontology (in particular to the subset of ontology that describes the qualitative questionnaire), and the IRs are used to enrich the set of Expert Rules (ERs) provided by an expert in economics.

Below the details and the results of the IR extraction procedure are given.

**Knowledge Repository** - Data and metadata reside in the MUSING ontologies. The questionnaire adopted in the Online Self Assessment service is described by the so called BPA ontology. A fragment of the integrated ontologies is depicted in Fig. 4. The concepts that belong to upper or related ontologies are labelled with the corresponding prefix (i.e. *psys*, *ptop* or *company*), while for the concepts that belong to the BPA ontology the prefix is missing for saving space. The black continuous arrows represent the *isA* relationships, while the blue broken-line arrows represent the object properties. Not all the relationships nor the object properties and labels have been drawn for the picture clarity sake.

**The Data** - The dataset used to train and test the models has been provided by the Italian bank Monte dei Paschi di Siena (MPS). The data set, composed of 6000 records contains the following information:

- 13 Qualitative Variables representing a subset of the questions included in the Qualitative Questionnaire performed by MPS to assess the credit worthiness of a third party, and in particular utilised to calculate the Qualitative Score of a Company.
- The Qualitative Score (target item of the classification task).
- 80 Financial/Economic indicators calculated from the Balance Sheets and representing a part of the information utilised to evaluate the probability of the default of a company.

**Extraction of the Relevant Concepts** - The HITSxONTO algorithm has been applied to the MUSING ontologies yielding a list of 552 ranked concepts. The computation ends after four iterations, returning a list of 5 concepts with hub score greater than 0 and a list of 14 concepts with authority score greater than 0. This is because the ontology is large and not strongly connected.

**Construction of the IRs Schemas** - Considering all the concepts in the lists as candidates, we obtain 2097 IRs Schemas with exactly one implicant and one implicated.

**Characterization of the IRs** - After a suitable filtering procedure we apply PATTERNIST to a set of 5757 instances of questionnaires. Having set the minimum support to 20%, PATTERNIST returns a set of 56 frequent itemsets (pairs of concepts). The result of the characterization of the IRs Schemas by using the set of frequent itemsets, is the following set of 14 IRs:

1. `ResearchAndDevelopment.isACompanyInvestment=1`  $\xrightarrow{26\%}$   
`PreviousAchievements.hasPrevAchievements=1.`
2. `ResearchAndDevelopment.isACompanyInvestment=1`  $\xrightarrow{30\%}$   
`CapitalizationStrategy.isTheIncreasingForeseen=2.`
3. `ResearchAndDevelopment.isACompanyInvestment=2`  $\xrightarrow{28\%}$   
`PreviousAchievements.hasPrevAchievements=2.`
4. `StrategicVisionAndQualityManagement.hasRate=2`  $\xrightarrow{28\%}$   
`CapitalizationStrategy.isTheIncreasingForeseen=2.`



5. CapitalizationStrategy.isTheIncreasingForeseen=2  $\xrightarrow{36\%}$   
PreviousAchievements.hasPrevAchievements=2.
6. ManagementTeam.hasYearOfExperience=1  $\xrightarrow{32\%}$   
PreviousAchievements.hasPrevAchievements=2.
7. ResearchAndDevelopment.isACompanyInvestment=2  $\xrightarrow{31\%}$   
PreviousAchievements.hasPrevAchievements=1.
8. StrategicVisionAndQualityManagement.hasRate=2  $\xrightarrow{42\%}$   
PreviousAchievements.hasPrevAchievements=1.
9. CapitalizationStrategy.isTheIncreasingForeseen=2  $\xrightarrow{48\%}$   
PreviousAchievements.hasPrevAchievements=1.
10. ManagementTeam.hasYearOfExperience=1  $\xrightarrow{54\%}$   
PreviousAchievements.hasPrevAchievements=1.
11. ResearchAndDevelopment.isACompanyInvestment=2  $\xrightarrow{54\%}$   
CapitalizationStrategy.isTheIncreasingForeseen=2.
12. StrategicVisionAndQualityManagement.hasRate=2  $\xrightarrow{60\%}$   
CapitalizationStrategy.isTheIncreasingForeseen=2.
13. ManagementTeam.hasYearOfExperience=1  $\xrightarrow{62\%}$   
StrategicVisionAndQualityManagement.hasRate=2.
14. ManagementTeam.hasYearOfExperience=1  $\xrightarrow{73\%}$   
CapitalizationStrategy.isTheIncreasingForeseen=2.

To correctly interpret these rules, please refer to the description of the qualitative questionnaire and its codification, reported in [10].

For example, the meaning of the last IR (IR\_14) is:

*In 73% of the cases, if the management team has more than 10 years of experience in the industrial sector, then the company does not foresee to increase its capital.*

This IR\_14, in agreement with what we just stated, belongs to the following schema:

ManagementTeam  $\rightarrow$  CapitalizationStrategy

which is one of the 2097 schemas extracted in the previous phase. Here it is clear that the schema provides the structure of a set of future IRs; it defines the direction of the implication and what are the involved concepts. The frequent itemset, instead, identifies the interesting datatype properties (related to the considered concepts) and assigns the weight (i.e. the support), making one of the possible instances compatible with that schema.

## 6 New Developments

The successful results obtained in the MUSING project and in the economic domain encouraged us to further work on the system and to carry on new experiments. In particular, the extension covers two aspects:



```

PreviousAchievements.hasPrevAchievements=2,
StrategicVisionAndQualityManagement.hasRate=1,
ResearchAndDevelopment.isACompanyInvestment=2  $\xrightarrow{1\%}$ 
CapitalizationStrategy.isTheIncreasingForeseen=2 (c=98%)

```

*“In the 1% of the cases, if the company owner/CEO has no relevant past experiences, the rate of the strategic vision and quality of management is Excellent and the company does not invest in R&D, then the company is not foreseeing to increase its capitalisation.”*

These two IRs have a very low probability but an high confidence, and they can be considered important for an analyst interested in the behavior of a company towards the strategies of management, the investment in the R&D, and the way to finance them.

```

ManagementTeam.hasYearOfExperience=3  $\xrightarrow{1\%}$ 
PreviousAchievements.hasPrevAchievements=2 (c=64%)

```

*“In the 1% of the cases, if the years of experience of the management team are less than 5, then company owner/CEO has no relevant past experiences”.*

This is a really rare case, but maybe it should be taken into consideration because of its not negligible confidence value.

## 7 Conclusions and Future Works

In this paper we have presented how we handled the problem of extracting interesting and implicit knowledge out of an ontology, presenting the results in form of influence rules. Our idea was to drive the extraction process by using the ontology structure, and to exploit the instances only in a second step. The main problem was to understand if and how to use traditional methods for DM in the context of the ontology. Obviously, the traditional systems can be used only as models, but they are not directly applicable to the ontologies. By decomposing the problem into sub-problems, we succeeded in finding a methodology taking inspiration from consolidated theories and recent developments.

Besides the theoretical results, we tested the system in an concrete setting exploiting our involvement in a European industrial research project: MUSING. In this way, we had at our disposal an integrated framework and a real set of data. Our analysis tool mainly solves, in this domain, the problem of the availability of the expert knowledge. In fact, in the economic field, obtaining a cognitive net of relationships from experts is a hard task, either for the complexity of the matter, or for the lack of specific studies (very often these rules are based on the expert believes or his/her own experience).

In the paper, we focused on the economic domain using the IRs for augmenting a set of “similar” (for meaning, structure and objective) rules. Nevertheless, it is important to point out that the system is fully general and can be used in several domains i.e. in all the domains that can be described by an ontology and where instances are available.

To demonstrate this, we further tested the system in the domain of intrusion detection. For this new case study, that is documented in [11], we used (i) the ontology created by Pinkston et al. to detect different attacks [16] (57 classes, 21 object

properties and 47 datatype properties), and (ii) a set of instances regarding intrusion detection taken from the UCI KDD Archive [19] (27 variables and 311029 records).

Under appropriate system settings, we obtained a set of 70 IRs. Among these rules, some of them result particular informative because they illustrates how dangerous an attack can be. See for example the following rules:

**IR<sub>A</sub>:** *In 52% of the cases, when the attack is classified as a 'smurf', the number of the hot indicator is '0' (i.e. the attack is not dangerous).*

**IR<sub>B</sub>:** *In 20% of the cases, when the connection status is 'normal', the number of bytes from the source to the destination is between 1 and 1032, the number of bytes from the destination to the source is greater than 100 and there were no connections to the same host as the current connection in the past two seconds, there were no files created.*

We believe that one of the many advantages of our approach is the interpretability of the rules, thanks to the semantics provided by the ontology. From the ontology we get not only a list of concepts but also their description and the meaning of their relationships. The process of interpretation, however, is still not fully automatic and, as happened in MUSING, the support of the domain expert is valuable. Thus the challenge is to exploit the information (explicit and hidden) in the ontology and from other sources as much as possible, in order to automate the interpretation process, especially because the support of experts is not always available.

Moreover, the new extension further enriches the system, making the IRs much more informative and interesting than before.

**Acknowledgements.** This paper was supported by MUSING Project (IP FP-027097) which provided an useful and convenient framework.

## References

1. Baglioni, M., Bellandi, A., Furletti, B., Spinsanti, L., Turini, F.: Ontology-based business plan classification. In: Proceedings of EDOC 2008, pp. 365–371 (2008)
2. Baglioni, M., Furletti, B., Turini, F.: DrC4.5: Improving C4.5 by means of prior knowledge. In: Proceedings of SAC 2005, pp. 474–481 (2005)
3. Bellandi, A., Furletti, B., Grossi, V., Romei, A.: Pushing Constraints in Association Rule Mining: An Ontology-Based Approach. In: Proceedings of the IADIS International Conference WWW/INTERNET (2007)
4. Bellini, L.: YaDT-DRb: Yet another Decision Tree Domain Rule builder. Master's Thesis, University of Pisa, Italy (2007)
5. Berners-Lee, T., Fischetti, M.: Weaving the Web. Harper, San Francisco (1997)
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. In: The 7th International World Wide Web Conference, Brisbane, Australia (1998)
7. Bonchi, F., Giannotti, F., Lucchese, C., Orlando, S., Perego, R., Trasarti, R.: Conquest: a constraint-based querying system for exploratory pattern discovery. In: ICDE (2006)
8. Ciaramita, M., Gangemi, A., Ratsch, E., Saric, J., Rojas, I.: Unsupervised Learning of Semantic Relations for Molecular Biology Ontologies. In: Ontology Learning and Population: Bridging the Gap between Text and Knowledge (2008)
9. Elsayed, A., El-Beltagy, S.R., Rafea, M., Hegazy, O.: Applying data mining for ontology building. In: Proceedings of the 42nd Annual Conference on Statistics, Computer Science, and Operations Research (2007)

10. Furletti, B.: Ontology-Driven Knowledge Discovery. Ph.D. Thesis: IMT-Lucca (2009), <http://www.di.unipi.it/furletti/papers/PhDThesisFurletti2009.pdf>
11. Furletti, B., Turini, F.: Knowledge Discovery in Ontologies. To Appear in: IDA Journal 16(3) (2012)
12. Geller, J., Zhou, X., Prathipati, K., Kanigiluppai, S., Chen, X.: Raising data for improved support in rule mining: How to raise and how far to raise. *Intelligent Data Analysis* 9(4), 397–415 (2005)
13. Kleinberg, J.: Authoritative sources in a hyperlinked environment. In: *ACM-SIAM Symposium on Discrete Algorithms*, pp. 668–677 (1998)
14. MUSING Project (2006), <http://www.musing.eu/>
15. Parekh, V., Gwo, J., Finin, T.: Mining Domain Specific Texts and Glossaries to Evaluate and Enrich Domain Ontologies. In: *Proceedings of the International Conference of Information and Knowledge Engineering* (2004)
16. Pinkston, J., Undercoffer, J., Joshi, A., Finin, T.: A Target-Centric Ontology for Intrusion Detection. In: *Proceedings of the Workshop on Ontologies in Distributed Systems* (2003)
17. Quinlan, J.: *C4.5: programs for machine learning*. Morgan Kaufmann P. (1993)
18. Singh, S., Vajirkar, P., Lee, Y.: Context-Based Data Mining Using Ontologies. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) *ER 2003. LNCS*, vol. 2813, pp. 405–418. Springer, Heidelberg (2003)
19. UCI KDD Archive. Data of The Third International Knowledge Discovery and Data Mining Tools Competition (KDD CUP 1999) (1999), <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
20. Vela, M., Declerck, T.: Heuristics for Automated Text-Based Shallow Ontology Generation. In: *Proceedings of the International Semantic Web Conference, Posters & Demos* (2008)
21. W3C Community, *OWL Web Ontology Language Overview* (2004), <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
22. W3C Community, *RDF Vocabulary Description Language 1.0: RDF Schema* (2004), <http://www.w3.org/TR/rdf-schema/>

# Author Index

- Bertolino, Antonia 20  
Bugliesi, Michele 151
- Calabrò, Antonello 20  
Cortesi, Agostino 151  
Cuadrado, Félix 166
- Dobrica, Liliana 199  
Dueñas, Juan C. 166
- España, Sergio 56
- Fujita, Masahiro 181  
Furletti, Barbara 270
- García-Carmona, Rodrigo 166  
García-Domínguez, Antonio 136  
Giandomenico, Felicita Di 20  
Gottron, Thomas 217  
Grambow, Gregor 73
- Hermanns, Christian 91  
Hryniewicz, Olgierd 255  
Huynh, Nguyen T. 108
- Inverardi, Paola 20  
Ivanov, Ivan I. 3
- Jaakola, Marko 199
- Kolp, Manuel 121  
Kuchen, Herbert 91
- Le, Anh D. 108  
Le, Nhat-Van 108
- Marca, David A. 38  
Marcos-Bárcena, Mariano 136  
Matthes, Florian 237  
Medina-Bulo, Inmaculada 136  
Mohammadzadeh, Hadi 217
- Nakhaeizadeh, Gholamreza 217  
Navas, Álvaro 166  
Neubert, Christian 237  
Nguyen, Phung H. 108  
Nostro, Nicola 20
- Oberhauser, Roy 73  
Ovaska, Eila 199
- Pastor, Óscar 56  
Poelmans, Stephan 121  
Prasad, Mukul R. 181  
Purhonen, Anu 199
- Quan, Tho T. 108
- Rajan, Sreeranga P. 181  
Reichert, Manfred 73  
Ruiz, Marcela 56
- Schweiggert, Franz 217  
Spalazzese, Romina 20  
Spanò, Alvise 151  
Steinhoff, Alexander 237
- Tanida, Hideo 181  
Turini, Franco 270
- Wautelet, Yves 121