

Florian Daniel
Federico Michele Facca (Eds.)

LNCS 6385

Current Trends in Web Engineering

10th International Conference on Web Engineering
ICWE 2010 Workshops, Vienna, Austria, July 2010
Revised Selected Papers

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Florian Daniel
Federico Michele Facca (Eds.)

Current Trends in Web Engineering

10th International Conference on Web Engineering
ICWE 2010 Workshops, Vienna, Austria, July 2010
Revised Selected Papers

Volume Editors

Florian Daniel
University of Trento
Dipartimento di Ingegneria e Scienza dell'Informazione
Via Sommarive 14
38100 Povo, Trento, Italy
E-mail: daniel@disi.unitn.it

Federico Michele Facca
CREATE-NET, Via alla Cascata 56/D
38123 Povo, Trento, Italy
E-mail: federico.facca@create-net.org

Library of Congress Control Number: 2010938284

CR Subject Classification (1998): H.5, H.4, H.3, K.6, D.2, C.2, H.3.5, H.5.3

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

ISSN 0302-9743
ISBN-10 3-642-16984-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-16984-7 Springer Berlin Heidelberg New York

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.

springer.com

© Springer-Verlag Berlin Heidelberg 2010
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Foreword

Continuing its consolidated and prestigious tradition, the tenth edition of the International Conference on Web Engineering (ICWE 2010) complemented its research and industrial program with a selection of workshops extending the conference's program. The workshops offered Web engineering researchers and practitioners the opportunity for highly interactive sessions, which included in-depth, topical presentations and discussions of emerging research challenges and preliminary results. This volume collects the contributions of the hosted workshops and the co-located Doctoral Consortium.

In light of the need to offer an always novel and interesting program reflecting the emerging research of the Web engineering community, we made a huge effort to renovate the conference's workshop program, to enlarge the choice to authors and participants, and to attract high-quality contributions. The Web engineering community is a constantly growing group that, on the one hand, is working on a set of consolidated topics and, on the other hand, is constantly getting inspired by cutting edge technologies or societal trends. Thus, to enlarge the choice to both potential workshop organizers and authors, in designing this edition's workshop program we pursued two goals: First, we aimed at providing the typical audience of ICWE with a set of workshop topics that reflect the traditional interests of the community and that are as wide and comprehensive as possible. We achieved this goal thanks to well-established workshops covering topics such as model-driven development, quality and usability of Web applications, rich Internet applications (RIAs), and light-weight, component-based development of Web applications (e.g., mashups). Second, we wanted to attract new audiences to the conference, selecting also novel workshops that cover topics of an emerging or foundational nature that extend the focus of ICWE beyond its traditional topics. We reached this goal by betting on subjects such as the Semantic Web, semantic data management, the Internet of things, Web-enabled tourism, service-based compliance management, and e-crowdsourcing.

The result of this work is the selection of 9 workshops, out of 16 submitted workshop proposals, that were successfully held in Vienna on July 5 and 6, 2010:

- *MDWE 2010*: Sixth Model-Driven Web Engineering Workshop
- *QWE 2010*: First International Workshop on Quality in Web Engineering
- *SWIM 2010*: Second International Workshop on Semantic Web Information Management
- *SWEng 2010*: First International Workshop on Service Web Engineering
- *ESW 2010*: First Workshop on Engineering SOA and the Web
- *ComposableWeb 2010*: Second International Workshop on Lightweight Composition on the Web
- *EC 2010*: First International Workshop on Enterprise Crowdsourcing

- *TouchTheWeb 2010*: First International Workshop on Web-Enabled Objects
- *WEBTOUR 2010*: First International Workshop on Web Engineering and Tourism

The success of this year’s workshop program is reflected in the incredible number of submissions, the high level of participation, and the impact registered by the individual workshops. Together, all workshops attracted more than 100 workshop paper submissions, an outstanding result. Including the submissions to the Doctoral Consortium, the co-located events for this edition of ICWE jointly attracted almost as many submissions as the main conference. The high number of submissions, on the one hand, implied higher competition among the submissions and, on the other hand, assured the top quality of the selected papers.

Following the exceptionality of this year’s edition, we decided for the first time to establish a best workshop paper award, to be assigned to the best contribution out of all workshop paper submissions. The award was won by Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio for their paper entitled “*Semantic Wonder Cloud: Exploratory Search in DBpedia*,” which was presented at the SWIM 2010 workshop. You can find the paper in this volume.

As an acknowledgment of the quality of the workshop program, we are proud that we could reach an agreement with Springer for the publication of all accepted papers in Springer’s *Lecture Notes in Computer Science (LNCS)* series. We opted for post-workshop proceedings, a publication modality that allowed the authors – when preparing the final version of their papers for inclusion in the proceedings – to take into account the feedback they received during the workshops and to further improve the quality of their papers. We believe that this volume provides an interesting and up-to-date picture of what are the last trends and new ideas fermenting right now in the Web engineering community. Some of the papers included in this volume unveil unexpected, novel aspects and synergies that we think will be taken up in the future and may become mainstream research lines.

Without enthusiastic and committed authors this volume would not have been possible. Thus, our first thanks go to the researchers, practitioners, and PhD students who contributed to this volume with their work. We thank the organizers of the workshops and the Doctoral Consortium who reliably managed the organization of their events, the selection of the highest-quality papers, and the moderation of their events during the two workshop days. Finally, we would like to thank the General Chair of ICWE 2010, Gerti Kappel, and the Program Chairs, Boualem Benatallah, Fabio Casati, and Gustavo Rossi, for their constant support and trust in our work. We enjoyed organizing this edition of the workshop program, reading the articles, and assembling the post-workshop proceedings in conjunction with the workshop organizers. We hope you enjoy in the same way the reading of this volume.

Preface

The preface of this volume collects the prefaces of the post-workshop proceedings of the individual workshops. The actual workshop papers, grouped by event, can be found in the body of this volume.

Sixth International Workshop on Model-Driven Web Engineering (MDWE 2010)

Organizers: Gustavo Rossi, Nora Koch, Geert-Jan Houben, and Antonio Vallecillo

Web engineering is a specific domain in which model-driven engineering (MDE) can be successfully applied. Existing model-based Web engineering approaches already provide excellent methods and tools for the design and development of most kinds of Web applications. They address different concerns using separate models (navigation, presentation, workflows, etc.) and come with model compilers that produce most of the application's Web pages and logic based on these models. However, most of these Web engineering proposals do not fully exploit all the potential benefits of MDE, such as complete platform independence, meta-modeling, and model transformations.

In addition, there is an increasing trend towards the incorporation of emerging technologies like rich Internet applications, mashups, Semantic Web and (Semantic) Web rule languages, which aim at fostering application interoperability, especially within the scope of Web 2.0 and its related technologies and richer applications. These current trends involve new challenges both to the modeling and model-driven development domains. However, the effective integration of all these new techniques with the already existing model-based Web engineering approaches is still unresolved.

Accordingly, we invited original, high-quality submissions for the 2010 edition of the Model-Driven Web Engineering Workshop. In response to the call for papers, a total of eleven submissions were received. Each submitted paper was formally peer reviewed by at least three referees, and six papers were finally accepted for presentation at the Workshop and publication in the proceedings. This workshop builds on the success of the preceding 2005, 2006, 2007, 2008 and 2009 MDWE Workshops (held, respectively, in Sydney jointly with ICWE 2005, in Menlo Park jointly with ICWE 2006, and in Como jointly with ICWE 2007, in Toulouse jointly with MoDELS 2008 and in San Sebastián jointly with ICWE 2009). This year it was held in conjunction with the ICWE 2010 conference in Vienna, Austria.

The aim of this workshop was to provide a forum to discuss the state of the art in model-driven Web engineering (MDWE), where researchers and practitioners could meet to outline a roadmap that addresses the key challenges in this area. After each presentation, workshop participants and presenters had the opportunity to exchange some ideas. In the final session the lively discussion focused on the inclusion of new models in the model-driven process, such as models for quantitative analysis and models for understanding user behavior. Discussion was also centered on model evolution and the need of real examples for evaluating model-driven engineering approaches. Further information about the presented papers, together with the Call for Papers, and all the information relevant to the workshop, is available at the website of the event (see <http://mdwe2010.pst.ifi.lmu.de/>)

We would like to thank the ICWE 2010 organization for giving us the opportunity to organize this workshop, especially to the Workshops Chairs, Florian Daniel and Federico M. Facca, who were always very helpful and supportive. Many thanks to all those that submitted papers, and particularly to the contributing author and the presenters of the papers: Gerald Hübsch, Esteban Robles Luna, Alessio Gambi, Daniel Schwabe, Tobias George and Alessandro Bozzon. Our gratitude also goes to the reviewers and the members of the Program Committee for their timely and accurate reviews and for their help in choosing and improving the selected papers.

July 2010

Gustavo Rossi
Nora Koch
Geert-Jan Houben
Antonio Vallecillo

Program Committee

Luciano Baresi	Politecnico di Milano, Italy
Sara Comai	Politecnico di Milano, Italy
Olga De Troyer	Vrije Universiteit Brussel, Belgium
Damiano Distanto	Università Telematica TEL.M.A., Italy
María José Escalona	Universidad de Sevilla, Spain
Piero Fraternali	Politecnico di Milano, Italy
Howard Foster	Imperial College London, UK
Irene Garrigós	Universidad de Alicante, Spain
Athula Ginige	University of Western Sydney, Australia
Heinrich Hussmann	Universität München, Germany
Alexander Knapp	University of London, UK
Maristella Matera	Politecnico di Milano, Italy
Santiago Meliá	Universidad de Alicante, Spain
Pierre-Alain Muller	University of Haute-Alsace, France
Vicente Pelechano	Universidad Politécnica de Valencia, Spain

Alfonso Pierantonio	Università Degli Studi Dell'Aquila, Italy
Fernando Sánchez Figueroa	Universidad de La Rioja, Spain
Juan Manuel Vara	Universidad Rey Juan Carlos, Spain
Marco Winckler	University Paul Sabatier, France

First International Workshop on Quality in Web Engineering (QWE 2010)

Organizers: Silvia Abrahão, Cinzia Cappiello, Cristina Cachero, and Maristella Matera

The production of Web applications has been among the fastest growing segments of the software industry for several years. Web-based companies depend on customers using their sites, and most importantly, returning to their sites. This means that unlike many other applications, Web applications only succeed if they satisfy customers' needs. However, some recent studies suggest that more than 50% of the delivered Web applications are of poor quality.

To achieve quality for any class of Web products (e.g., a data-intensive application, a Web service, a community portal), the set of relevant quality attributes for Web artifacts must be clearly defined. Otherwise, quality assessment is left to the intuition or the responsibility of people who are in charge of the process. Quality models should be built to precisely identify the quality attributes and the relationships among them. Based on these quality models, adequate (i.e., quality aware) development and assessment techniques should then be applied during the whole application life cycle.

This need to reflect on and advance methods and techniques that help improve the quality of delivered Web applications led us to organize the first edition of the International Workshop on Quality in Web Engineering (QWE 2010) that was held in conjunction with the 10th International Conference on Web Engineering (ICWE), in Vienna, Austria, in July 2010. The event capitalized on our previous experience with the organization of three editions of the International Workshop of Web Usability and Accessibility (IWWUA), trying, however, to enlarge the initial usability and accessibility scope to the broader concept of quality. The main purpose was to discuss new, quality-oriented trends in Web application engineering, and to provide an international forum for information exchange on methodological, technical and theoretical aspects.

These proceedings collect the papers presented at the workshop. All the submitted papers were peer-reviewed by three independent reviewers. The acceptance rate of the workshop was 44%. The accepted papers (four regular papers and one short paper) discuss novel aspects of Web application quality, ranging from new quality models and quality-enhanced development processes to advanced quality factors, such as security in mobile environments, reliability of search engine results, and also adaptivity through recommendations.

We would like to thank all the authors for submitting their manuscripts to the workshop and contributing to the fine form that the interesting program

took. Also, we would like to thank the members of the Program Committee for their efforts in the reviewing process, and the ICWE organizers for their support and assistance in the production of these proceedings.

We are especially grateful to Shadi Abou-Zahra, Activity Lead at the WAI International Program Office (W3C, Wien, Austria), for his interesting keynote speech “Better Web: Accessible and Standards Compliant.” More details on the workshop are available at <http://gplsi.dlsi.ua.es/congresos/qwe10/>

July 2010

Silvia Abrahão
Cinzia Cappiello
Cristina Cachero
Maristella Matera

Program Committee

Shadi Abou-Zahra	World Wide Web Consortium (W3C)
Davide Bolchini	Indiana University, USA
Giorgio Brajnik	University of Udine, Italy
Ismael Caballero	University of Castilla-la-Mancha, Spain
Coral Calero	University of Castilla-la-Mancha, Spain
Tiziana Catarci	University of Rome, Italy
Sven Casteleyn	Vrije Universiteit Brussel, Belgium
Florian Daniel	Politecnico di Milano, Italy
Martin Gaedke	Chemnitz University of Technology, Germany
Bernd Heinrich	Innsbruck University School of Management, Austria
Geert-Jan Houben	Delft University of Technology, The Netherlands
Emilio Insfran	Universidad Politécnica de Valencia, Spain
Effie Lai-Chong Law	ETH Zürich, Switzerland
Maria Dolores Lozano	University of Castilla-la-Mancha, Spain
Vicente Luque Centeno	University Carlos III, Spain
Luis Olsina	Universidad Nacional de La Pampa, Argentina
Geert Poels	University of Ghent, Belgium
Simos Retalis	University of Piraeus, Greece
Gustavo Rossi	LIFIA, UNLP, Argentina
Carmen Santoro	ISTI-CNR, Italy
Monica Scannapieco	University of Rome, Italy
Wieland Schwinger	Johannes Kepler University, Austria
Marco Winckler	University Paul Sabatier, France

Second International Workshop on Semantic Web Information Management (SWIM 2010)

Organizers: Roberto De Virgilio, Fausto Giunchiglia, and Letizia Tanca

The ceaseless expansion of the World Wide Web is making it more and more complex for humans to efficiently find the needed information. The underlying idea of having a description of the data on the Web, organized in such a way as to be used by machines for automation, integration and reuse across various applications, has been exploited in several research fields.

As in the first edition, this International Workshop on “Semantic Web Information Management” (SWIM) aimed at reviewing the most recent data-centered solutions for the Semantic Web. In particular, its ambition was to present and analyze the techniques for semantic information management, by taking advantage of the synergisms between the logical basis of the Semantic Web and the logical foundations of conceptual modeling. Indeed, the leitmotif of this research is the proposal of models and methods conceived to represent and manage the so-called “semantic data,” that is, data appropriately structured to be easily machine-processable on the Web, according to semantic models (e.g. RDF, RDF(S), OWL). The long-standing experience of the information modeling community can provide a priceless contribution to the substantial problems arising in semantic data management.

The new research issues can be summarized by the following problems:

1. How can we efficiently and effectively store large amounts of semantic data?
2. How can we query semantic data and reason on them in a feasible way?
3. How can we exploit such semantic data in real world scenarios?

This workshop covers the emerging area of the Semantic Web, gathering researchers to debate, propose, and elaborate the foundations for a data-modeling approach to these problems, by presenting running research and projects on these topics.

June 2010

Roberto De Virgilio
Fausto Giunchiglia
Letizia Tanca

Program Committee

Nick Bassiliades	Aristotle University of Thessaloniki, Greece
Devis Bianchini	University of Brescia, Italy
Sourav S. Bhowmick	Nanyang Technological University, Singapore
Jorge Bernardino	Polytechnic of Coimbra, Portugal
Paolo Cappellari	Dublin City University, Ireland

Barbara Carminati	University of Insubria, Italy
Florian Daniel	University of Trento, Italy
Roberto De Virgilio	Università Roma Tre, Italy
Valeria De Antonellis	University of Brescia, Italy
Claudio de Souza Baptista	Universidade Federal de Campina Grande, Brazil
Pierluigi Del Nostro	Università Roma Tre, Italy
Tommaso Di Noia	Technical University of Bari, Italy
Eugenio Di Sciascio	Technical University of Bari, Italy
Ismael Navas Delgado	Universidad de Malaga, Spain
Ian Dickinson	HPL, UK
Alfio Ferrara	University of Milan, Italy
Flavius Frasinarc	Erasmus University Rotterdam, The Netherlands
Giorgio Gianforme	Università Roma Tre, Italy
Fausto Giunchiglia	Università di Trento, Italy
David M. Hansen	George Fox University, Oregon, USA
Stijn Heymans	Vienna University of Technology, Austria
Jan Hidders	Delft University of Technology, The Netherlands
Georg Lausen	University of Freiburg, Germany
Sebastian Link	Victoria University of Wellington, New Zeland
Brian Matthews	Rutherford Appleton Laboratory, UK
Amedeo Napoli	Lorraine Laboratory of IT Research and its Applications, France
Stefano Paolozzi	Università Roma Tre, Italy
Tore Risch	Uppsala University, Sweden
Simon Scerri	National University of Ireland, Ireland
Amandeep S. Sidhu	Murdoch University, Australia
Letizia Tanca	Polytechnic of Milan, Italy
A Min Tjoa	Vienna University of Technology, Austria
Riccardo Torlone	Università Roma Tre, Italy
Gottfried Vossen	University of Munster, Germany
Wolfram WöB	Johannes Kepler University Linz, Austria

First International Workshop on Service Web Engineering (SWEng 2010)

Organizers: Lyndon Nixon, John Domingue, and Barry Norton

The First Workshop on Service Web Engineering (SWEng) grew out of a thriving research vision of the service Web, driven by the continuing trend towards service-oriented architectures (SOA), cloud computing and Web 2.0 APIs, in which websites provide programmatic access to their content and functionality. We envision a future Web in which billions of services will be seamlessly found, composed and executed just as how in today's Web we can find and browse

billions of documents, enabled by the use of semantic technologies for describing richly the functional and non-functional characteristics of services and enabling systems to automate the requisite mediation and agreement between services.

July 2010

Lyndon Nixon
John Domingue
Barry Norton

Program Committee

Luciano Baresi	Politecnico di Milano, Italy
Salima Benbernou	Université Paris Descartes, France
Zeta Dooly	Telecommunications Software and Services Group (TSSG), Ireland
Kevin Doolin	Telecommunications Software and Services Group (TSSG), Ireland
Schahram Dustdar	Technical University of Vienna, Austria
Anastasios Gavras	Eurescom, Germany
Stamatis Karnouskos	SAP Research, Germany
Irene López de Vallejo	Tekniker, Spain
Carlos Pedrinaci	The Open University, UK
Klaus Satzke	Alcatel-Lucent, Germany
Pedro Soria-Rodriguez	Atos Research & Innovation, Spain
Ioan Toma	STI Innsbruck/University of Innsbruck, Austria
Tomas Vitvar	University of Innsbruck, Austria
Dominic Zyskowski	Poznan University of Economics, Poland

First International Workshop on Engineering SOA and the Web (ESW 2010)

Organizers: Uwe Zdun, Schahram Dustdar, and Bruno Crispo

Service-oriented architectures (SOA) are nowadays used as the backbone of many Web applications. The First Workshop on Engineering SOA and the Web (ESW 2010) focused on the overlap of engineering service-oriented systems and Web applications into a coherent system. The main goals of the workshop were to bring together experts, both from industry and academia, who work on the interdependent context of SOA and Web applications and use engineering methods to make that link.

The workshop focus was on high-quality papers in the topic areas: Monitoring of SOAs via the Web, Management of SOAs via the Web, Governance of SOAs via the Web, Integration of SOAs and Web Applications, Web Dashboards for SOAs, Monitoring and Management for Compliance of SOA, Metrics for SOA Management, Security Management in SOA, and Engineering Methods

and Approaches for SOAs and Web Applications. The following five papers were accepted for presentation at the workshop, and can also be found in the proceedings:

- Carlos Rodríguez, Patrícia Silveira, Florian Daniel and Fabio Casati. “Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction”
- Maciej Gawinecki, Giacomo Cabri, Marcin Paprzycki and Maria Ganzha. “Trade-Off Between Complexity of Structured Tagging and Effectiveness of Web Service Retrieval”
- Soumaya Marzouk, Afef Jmal Maâlej and Mohamed Jmaiel. “Aspect Oriented Checkpointing Approach of Composed Web Services”
- Afef Mdhaffar, Soumaya Marzouk, Riadh Ben Halima and Mohamed Jmaiel. “A Runtime Performance Analysis for Web Service-Based Applications”
- David Schumm, Oktay Turetken, Natallia Kokash, Amal Elgammal, Frank Leymann and Willem-Jan van den Heuvel. “Business Process Compliance Through Reusable Units of Compliant Processes”

These papers went through a rigorous review process, with three or four reviews for each of the papers. We would like to thank the members of the Program Committee for delivering in time their reviews and the ensuing discussion that led to the final selection of the papers.

July 2010

Uwe Zdun
Schahram Dustdar
Bruno Crispo

Program Committee

Farhad Arbab	Stichting Centrum voor Wiskunde en Informatica, The Netherlands
Aliaksandr Birukou	University of Trento, Italy
Vincenzo D’Andrea	University of Trento, Italy
Mohand-Said Hacid	Université Claude Bernard Lyon 1, France
Frank Leymann	Universität Stuttgart, Germany
Willem-Jan van den Heuvel	Stichting Katholieke Universiteit Brabant, The Netherlands
Huy Tran	Technische Universität Wien, Austria
Mark Strembeck	Vienna University of Economics, Austria
Ralph Mietzner	Universität Stuttgart, Germany
Marco Aiello	Rijksuniversiteit Groningen, The Netherlands
Pietro Mazzoleni	IBM, USA
Guenter Karioth	IBM, Switzerland
Philip Robinson	SAP

Second International Workshop on Lightweight Composition on the Web (ComposableWeb 2010)

Organizers: Florian Daniel, Sven Casteleyn, and Geert-Jan Houben

After its first edition in conjunction with last year's ICWE in San Sebastian, Spain, this year's edition of ComposableWeb also took place in conjunction with ICWE, this time in Vienna, Austria. The workshop focused on research, practical experiences, and novel ideas in the context of component-based development of Web applications, lightweight composition on the Web, Web 2.0, and mashups. The goal of the workshop was to provide a discussion forum bringing together researchers and practitioners working in these areas, in order to jointly advance current state-of-the-art solutions. The topics of the workshop typically attract enthusiastic people that like to play with novel technologies and that try to make application development accessible also to less skilled developers or—as envisioned by many—even to end-users.

The discussion at the end of the first edition of the workshop manifested the desire of the participants to have more discussions and hands-on-experience reporting during the workshop, and to give more space to the practical issues of the presented works. We took these hints seriously into consideration in the organization of this year's edition of the workshop and accompanied the traditional workshop paper presentations with (1) a keynote talk by Boualem Benatallah that reviewed the state of the art in service composition and reuse and looked at end-user programming with a critical eye, providing lots of insights and experience, and (2) a demo session dedicated to showcase running (or not) prototypes in a very informal, stand-up fashion, also involving workshop participants who did not have a formal paper to present during the workshop but that nevertheless were willing to contribute.

Regarding the scientific program, after a rigorous reviewing process 6 papers, out of 14 submissions, were accepted for presentation. This year the acceptance rate was lower than last year. This was related to the fact that slots in the workshops agenda had to be allotted for the keynote and the demo sessions and that publishing the proceedings of the workshop with Springer required us to keep the acceptance rate at a competitive level.

It turned out that the revised format of the workshop and the strong competition both contributed to a successful event with an average of 30-40 attendees throughout the whole day—a result we are particularly proud of (especially given this year's strong competition also among the different workshops hosted by ICWE), which also convinced us to re-propose the workshop in a similar format next year.

We would like to thank all the authors who contributed to the workshop with their papers and presentations, Boualem for his stimulating keynote, and the audience for actively participating in the discussions. We also would like to thank the following people who contributed to the demo session by providing details about their work and prototypes: Junxia Guo (partial information extraction tool), Tomoya Noro (mobile application demo), Cesare Pautasso (RESTful

mashups with JOpera), In-Young Ko (GeoWorlds Information Manager), Oscar Diaz (the crowd as co-author of application functionality), and Tobias Nestler (ServFace Builder for visual service composition). Finally, we would like to thank the ICWE Organizers and Workshop Chairs for hosting the workshop and providing a nice, relaxed, yet constructive environment.

July 2010

Florian Daniel
Sven Casteleyn
Geert-Jan Houben

Steering Committee

Sven Casteleyn	Vrije Universiteit Brussel, Belgium
Florian Daniel	University of Trento, Italy
Maristella Matera	Politecnico di Milano, Italy
Geert-Jan Houben	TU Delft, The Netherlands
Olga De Troyer	Vrije Universiteit Brussel, Belgium

Program Committee

Sören Auer	University of Leipzig, Germany
Boualem Benatallah	University of New South Wales, Australia
Fabio Casati	University of Trento, Italy
Francisco Curbera	IBM Research, USA
Peter Dolog	Aalborg University, Denmark
Schahram Dustdar	Technical University of Vienna, Austria
Tom Heath	Talis Information Ltd, UK
John Musser	ProgrammableWeb.com, USA
Cesare Pautasso	University of Lugano, Switzerland
Florian Rosenberg	CSIRO ICT Centre, Australia
Gustavo Rossi	Universidad Nacional de La Plata, Argentina
Michael Weiss	Carleton University, Canada

First International Workshop on Enterprise Crowdsourcing (EC 2010)

Organizers: Claudio Bartolini and Maja Vuković

Web 2.0 technologies have enabled harnessing large crowds of users for mass data collection and problem solving activities. Over the past few years, crowdsourcing has been employed in a range of domains, beyond pharmaceutical research and software development. There are two types of approaches that have been proposed in order to catalyze the involvement of the crowd. The first approach relies on motivating humans to share information and by that either gain creditability or obtain the equivalent information. The other approach provides explicit and tangible incentives to people for their work (e.g., monetary prizes).

With the increasing interest in harnessing large crowds as a scalable workforce online, enterprises require an understanding of the underlying business model and how transition to a crowdsourcing model would affect their existing processes. Twenty participants from research and industry took part in the workshop, which featured five position papers and two invited speakers. These proceedings are the output of the workshop. The contributions to the workshop can be broadly categorized into three areas: (1) quality assurance in crowdsourcing, (2) applications of enterprise crowdsourcing and (3) challenges in engaging enterprise crowds across geographies.

The workshop opened with the keynote from Schahram Dustdar, providing an overview of key research elements in human-based services. Karnin et al. presented an application of crowdsourcing to the processing of scanned documents. La Vecchia et al. described how to transform canonical business processes to crowdsourced business processes, while retaining the same level of quality and control of traditional outsourcing approaches with a conventional workforce. Maja Vukovic presented a novel application of crowdsourcing in IT asset management and discussed lessons learned from enterprise deployment.

Kern, et al. proposed a novel, matrix-based model for classifying quality assurance in enterprise crowdsourcing, and identifying the corresponding mechanism for engaging the crowd for data assurance. Arellano et al., applied the Metropolis Model for crowdsourced website development to promote script-based crowdsourcing. Oliviera et al. provided early insights into challenges of engaging subject matter experts in an open innovation processes, and identified the difference between North American and European contexts.

Many commonalities were identified across the presented works and discussions, namely: the importance of data quality, the integration of crowdsourcing with the business process, and a better understanding of how to engage and sustain the crowd of high contributors.

July 2010

Claudio Bartolini
Maja Vuković

Publicity Chair

Hamid Motahari HP Labs, USA

Program Committee

Daren Brabham	University of Utah, USA
Fabio Casati	University of Trento, Italy
Riley Crane	MIT, USA
Schahram Dustdar	Technical University of Vienna, Austria
Ohad Greenshpan	IBM Research, Israel
Vassilis Kostakos	University of Madeira, Portugal
Osamuyimen Stewart	IBM Research, USA

First International Workshop on Web-Enabled Objects (TouchTheWeb 2010)

Organizers: Fernando Lyardet and Vicente Pelechano

The first edition of the TouchTheWeb workshop took place at the Technical University of Vienna premises. The papers presented in these proceedings have been selected in a peer-review process, where each of the papers was reviewed by at least two members of the Program Committee. Thematically, the workshop was organized into three sessions: Linking the Web of Things, Emerging Applications & Interaction Paradigms, and Research Challenges for the Web of Things.

The first session discussed core technologies and approaches for linking, combining and exchanging information between Web-enabled things. The first paper “Connecting Smart Things Through Web Services Orchestrations” presents an approach to apply Web services concepts and standards such as WSDL and WS-BPEL to the physical world. This follows two clear trends in the community: on the one hand, REST-based devices that require minimal computing power to serve requests, and on the other hand, the growing capabilities of embedded hardware that enables such devices to be treated as full-fledged Web services. While the latter approach tends to abstract how the communication takes place from the nature of the participating entity (small devices), REST-based devices allow for a broader number of things to be inter-connected. The second paper “Mashing Up Your Web-Enabled Home” makes the case for enabling users to create their mashups of REST-based entities, and the value of sharing those mashups with the community. The first session was completed with the paper “Triple Space-Based Semantic Distributed Middleware for Internet of Things” that proposed a tuple-based scheme for sharing information among Web-Enabled things.

The second session focused on putting the Internet of Things to work with novel interaction mechanisms for contextual assistance to users performing different tasks. The paper “Touch-Based Services’ Catalogs for AAL” presents an infrastructure based on the concept of a catalog for providing a flexible linkage between physical elements and digital services. The second paper of this session, “Designing Context-aware Interactions for Task-based Applications,” introduces design principles based on the concept of simplicity for supporting workflows in an environment full of digitally augmented things.

Finally, the last session was devoted to summarizing several issues that appeared as common concerns throughout the different sessions: the need to find innovative applications and new scenarios to better understand how the physical Web can become a transforming tool. Another challenge detected was to find the right granularity in the relationship between physical elements and their digital counterparts beyond a one-to-one relationship. For example, the possibility of decoupling the service descriptions in different perspectives according to the shared properties of physical elements (e.g., devices that can be turned on and off can provide this interface in addition to other interfaces of their specific services was explored).

Other, more technical issues discussed were the role of space and spatial relationships for building meshes of things, and mechanisms for discovering things (e.g., distinguishing ordinary physical elements from those digitally augmented) and their services (e.g., finding new services that dynamically appear and disappear).

July 2010

Vicente Pelechano
Fernando Lyardet
Pau Giner

Program Committee

Erwin Aitenbichler	Technische Universität Darmstadt, Germany
Fabiano Dalpiaz	University of Trento, Italy
Martin Gaedke	Technische Universität Chemnitz, Germany
Kris Luyten	Hasselt University, Belgium
Tomás Sánchez López	Cambridge University, UK
Diego López de Ipiña	Universidad de Deusto, Spain
Kristof Van Laerhoven	Darmstadt University of Technology, Germany
Gustavo Rossi	Universidad Nacional de La Plata, Argentina
Andrew Tokmakoff	Phillips Research, The Netherlands
Iñaki Vázquez	Universidad de Deusto, Spain
Reiner Wichert	Fraunhofer IGD Darmstadt, Germany
Andreas Zinnen	SAP Research CEC Darmstadt, Germany

First International Workshop on Web Engineering and Tourism (WEBTOUR 2010)

Organizers: Hannes Werthner, Birgit Pröll, Arno Scharl, and Christoph Grün

The Internet has become a cornerstone of the tourism and travel industry and created an online travel market that helps tourists to search for information and book their trip online. The huge number of websites that offer travel-related information, however, might lead to daunting information overload. In order to support tourists in decision-making, new methods and technologies (e.g., Semantic Web, recommender systems, context-sensitive approaches, or innovative user interfaces) are required to deliver highly targeted services to tourists. Also the supplier side, including destination management and travel organizations, has to keep up with the rapid developments in ICT. For this purpose, they have to use innovative Web mining methods to analyze the market, inter-organizational Web applications/services to exchange information and novel forms of social Web and rich Internet applications to enforce the contact with their customers.

The crucial prerequisite for all these developments is proper Web engineering to allow for systematic development and the maintenance of next-generation tourism applications for the Web.

The purpose of the International Workshop on Web Engineering and Tourism (WEBTOUR) was to bring together researchers from diverse communities who are interested in discussing ideas and ongoing work related to the field of Web engineering and e-tourism.

WEBTOUR was organized in conjunction with the 10th International Conference on Web Engineering (ICWE) from July 5–9, 2010 in Vienna, Austria. In total, five papers were accepted for presentation. They focused on algorithms for supporting tourists in trip planning, Web-based tourist guides that integrate information from public transportation systems as well as frameworks that facilitate the evaluation of tourism destination websites with respect to their communication efficacy. Apart from these papers, the workshop program included an invited talk by Dieter Merkl, Professor at the Institute of Software Technology at the Vienna University of Technology, about cultural awareness in collaborative virtual environments.

Finally, the WEBTOUR organizers would like to thank the Program Committee members for their valuable comments on the submissions, the authors for inspiring papers, the audience for the interest in this workshop, and the Workshop Chairs as well as the ICWE Organizers for hosting this workshop.

July 2010

Hannes Werthner
 Birgit Pröll
 Arno Scharl
 Christoph Grün

Program Committee

Matthias Baldauf	Forschungszentrum Telekommunikation Wien (FTW), Austria
Paul O'Brien	Griffith University, Australia
Astrid Dickinger	MODUL University Vienna, Austria
Alexander Felfernig	Graz University of Technology, Austria
Franca Garzotto	Politecnico di Milano, Italy
Ulrike Gretzel	Texas A&M University, USA
Martin Hitz	University of Klagenfurt, Austria
Wolfram Höpken	University of Applied Sciences Ravensburg-Weingarten, Germany
Jens Krösche	Hagenberg University of Applied Science, Austria
Katina Michael	University of Wollongong, Australia
Francesco Ricci	Free University of Bozen-Bolzano, Italy
Wieland Schwinger	Johannes Kepler University of Linz, Austria
Steffen Staab	University of Koblenz-Landau, Germany
Nathalie Steinmetz	seekda, Austria
Markus Zanker	University of Klagenfurt, Austria

ICWE 2010 Doctoral Consortium

Organizers: Cesare Pautasso, and Takehiro Tokuda

The ICWE 2010 Doctoral Consortium held at the Vienna University of Technology was a unique opportunity for all participating PhD students to present their work and share their ideas in an international setting.

The aim of the ICWE 2010 Doctoral Consortium was to provide PhD students with an encouraging atmosphere to present their research, receive useful feedback from senior researchers, and exchange ideas/experiences with other PhD students in the area of Web engineering.

The 2010 edition of the ICWE Doctoral Consortium included 10 position papers by PhD students from all over the world selected out of 14 submissions. Each student gave a ten-minute presentation and received valuable comments both from their peers and from senior researchers. Through the consortium and by attending the entire ICWE 2010 conference the PhD students came to know each other very well and gained a better understanding of the multifaceted ongoing research on Web engineering.

We would like to thank the members of the DC Program Committee for their valuable feedback to the PhD students as well as the ICWE 2010 organizers for their excellent support in running the doctoral consortium. Also, we are grateful to Erik Wilde, Daniel Schwabe, and Martin Gaedke for actively participating during the consortium and for their contribution to the closing panel discussion.

July 2010

Cesare Pautasso
Takehiro Tokuda

Program Committee

Sven Casteleyn	Vrije Universiteit Brussels, Belgium
Oscar Diaz	University of the Basque Country, Spain
Peter Dolog	Aalborg University, Denmark
Athula Ginige	University of Western Sydney, Australia
Nora Koch	LMU Munich, Germany
Maristella Matera	Politecnico di Milano, Italy
Werner Retschitzegger	Johannes Kepler University Linz, Austria
Erik Wilde	University of California, Berkeley, USA

Table of Contents

Sixth Model-Driven Web Engineering Workshop (MDWE)

Rapid Development of Composite Applications Using Annotated Web Services	1
<i>Lars Dannecker, Marius Feldmann, Tobias Nestler, Gerald Hübsch, Uwe Jugel, and Klemens Muthmann</i>	
From Mockups to User Interface Models: An Extensible Model Driven Approach	13
<i>José Matías Rivero, Gustavo Rossi, Julián Grigera, Juan Burella, Esteban Robles Luna, and Silvia Gordillo</i>	
Model-Driven Web Engineering Performance Prediction with Layered Queue Networks	25
<i>Alessio Gambi, Giovanni Toffetti, and Sara Comai</i>	
Models and Meta Models for Transactions in Web Applications	37
<i>Mark Douglas Jacyntho and Daniel Schwabe</i>	
Using Actions Charts for Reactive Web Application Modeling	49
<i>Nina Geiger, Tobias George, Marcel Hahn, Ruben Jubeh, and Albert Zündorf</i>	
Modeling Search Computing Applications	61
<i>Alessandro Bozzon, Marco Brambilla, Alessandro Campi, Stefano Ceri, Francesco Corcoglioniti, Piero Fraternali, and Salvatore Vadacca</i>	

First International Workshop on Quality in Web Engineering (QWE)

Developing Security Assessment Models in Web ² Mobile Environments	73
<i>Bong Gyou Lee, Hyunsik Seo, Giseob Byun, Keon Chul Park, Soo Kyung Park, and Taishiya Kim</i>	
Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications	85
<i>Daniel Mican and Nicolae Tomai</i>	
Quality in Use Model for Web Portals (QiUWeP)	91
<i>Mayte Herrera, M^a Ángeles Moraga, Ismael Caballero, and Coral Calero</i>	

Towards Support Processes for Web Projects	102
<i>Pablo Becker and Luis Olsina</i>	
Reliability Verification of Search Engines' Hit Counts: How to Select a Reliable Hit Count for a Query	114
<i>Takuya Funahashi and Hayato Yamana</i>	
Second International Workshop on Semantic Web Information Management (SWIM)	
Selecting Materialized Views for RDF Data	126
<i>Roger Castillo and Ulf Leser</i>	
Semantic Wonder Cloud: Exploratory Search in DBpedia	138
<i>Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio</i>	
Managing Adaptivity in Web Collaborative Processes Using Policies and User Profiles	150
<i>Juri Luca De Coi, Marco Fisichella, and Maristella Matera</i>	
Transformation of the Common Information Model to OWL	163
<i>Andreas Textor, Jeanne Stynes, and Reinhold Kroegeer</i>	
Improving Web Search Results for Homonyms by Suggesting Completions from an Ontology	175
<i>Tian Tian, James Geller, and Soon Ae Chun</i>	
How to Modify on the Semantic Web? A Web Application Architecture for Algebraic Graph Transformations on RDF	187
<i>Benjamin Braatz and Christoph Brandt</i>	
T ² .O.M. T.O.M.: Techniques and Technologies for an Ontology-Based Mobility Tool with Open Maps	199
<i>Michele Ruta, Floriano Scioscia, Saverio Ieva, and Eugenio Di Sciascio</i>	
An Approach to Semantic Information Retrieval Based on Natural Language Query Understanding	211
<i>Beniamino Di Martino</i>	
Slicing Linked Data by Extracting Significant, Self-describing Subsets: The DBpedia Case	223
<i>Michele Minno, Davide Palmisano, and Michele Mostarda</i>	
Automatically Identifying Bounds on Semantic Annotations for Bioinformatics Web Service Input Parameters	232
<i>Ravinder Singh, Sean Bechhofer, Khalid Belhajjame, and Suzanne M. Embury</i>	

First International Workshop on Service Web Engineering (SWEng)

REST Inspired Code Partitioning with a JavaScript Middleware	244
<i>Janne Kuuskeri and Tommi Mikkonen</i>	
The SOA Paradigm and e-Service Architecture Reconsidered from the e-Business Perspective	256
<i>Stanisław Ambroszkiewicz, Waldemar Bartyna, Marek Faderewski, Dariusz Mikulowski, Marek Pilski, Marcin Stepniak, and Grzegorz Terlikowski</i>	
Semantic Annotation of RESTful Services Using External Resources	266
<i>Victor Saquicela, Luis.M. Vilches-Blázquez, and Óscar Corcho</i>	

First Workshop on Engineering SOA and the Web (ESW)

Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction	277
<i>Carlos Rodríguez, Patrícia Silveira, Florian Daniel, and Fabio Casati</i>	
Trade-off between Complexity of Structured Tagging and Effectiveness of Web Service Retrieval	289
<i>Maciej Gawinecki, Giacomo Cabri, Maria Ganzha, and Marcin Paprzycki</i>	
Aspect-Oriented Checkpointing Approach of Composed Web Services	301
<i>Soumaya Marzouk, Afef Jmal Maâlej, and Mohamed Jmaiel</i>	
A Runtime Performance Analysis for Web Service-Based Applications	313
<i>Afef Mdhaffar, Soumaya Marzouk, Riadh Ben Halima, and Mohamed Jmaiel</i>	
Business Process Compliance through Reusable Units of Compliant Processes	325
<i>David Schumm, Oktay Turetken, Natallia Kokash, Amal Elgammal, Frank Leymann, and Willem-Jan van den Heuvel</i>	

Second International Workshop on Lightweight Composition on the Web (ComposableWeb)

An Approach to Enable Replacement of SOAP Services and REST Services in Lightweight Processes	338
<i>Teodoro De Giorgio, Gianluca Ripa, and Maurilio Zuccalà</i>	

Context, Quality and Relevance: Dependencies and Impacts on RESTful Web Services Design	347
<i>Hong-Linh Truong, Schahram Dustdar, Andrea Maurino, and Marco Comerio</i>	
Quality-Based Recommendations for Mashup Composition	360
<i>Matteo Picozzi, Marta Rodolfi, Cinzia Cappiello, and Maristella Matera</i>	
Partial Information Extraction Approach to Lightweight Integration on the Web	372
<i>Junxia Guo, Prach Chaisatien, Hao Han, Tomoya Noro, and Takehiro Tokuda</i>	
Domain-Specific Mashups: From All to All You Need	384
<i>Stefano Soi and Marcos Baez</i>	
Conceptual and Usability Issues in the Composable Web of Software Services	396
<i>Abdallah Namoun, Tobias Nestler, and Antonella De Angeli</i>	

First International Workshop on Enterprise Crowdsourcing (EC)

Crowdsourcing in the Document Processing Practice (A Short Practitioner/Visionary Paper)	408
<i>Ehud D. Karnin, Eugene Walach, and Tal Drory</i>	
Definition of a Crowdsourcing Innovation Service for the European SMEs	412
<i>Fábio Oliveira, Isabel Ramos, and Leonel Santos</i>	
Script Programmers as Value Co-creators	417
<i>Cristóbal Arellano, Oscar Díaz, and Jon Iturrioz</i>	
Quality Assurance for Human-Based Electronic Services: A Decision Matrix for Choosing the Right Approach	421
<i>Robert Kern, Hans Thies, Cordula Bauer, and Gerhard Satzger</i>	
Collaborative Workforce, Business Process Crowdsourcing as an Alternative of BPO	425
<i>Gioacchino La Vecchia and Antonio Cisternino</i>	

First International Workshop on Web-Enabled Objects (TouchTheWeb)

Connecting Smart Things through Web Services Orchestrations	431
<i>Antonio Pintus, Davide Carboni, Andrea Piras, and Alessandro Giordano</i>	

Mashing up Your Web-Enabled Home	442
<i>Dominique Guinard</i>	
A Triple Space-Based Semantic Distributed Middleware for Internet of Things	447
<i>Aitor Gómez-Goiri and Diego López-de-Ipiña</i>	
Touch-Based Services' Catalogs for AAL	459
<i>Jose Bravo, Ramon Hervás, and Jesus Fontecha</i>	
Designing Context-Aware Interactions for Task-Based Applications	463
<i>Pablo Muñoz, Pau Giner, and Miriam Gil</i>	

First International Workshop on Web Engineering and Tourism (WEBTOUR)

Tourist Trip Planning Functionalities: State-of-the-Art and Future	474
<i>Wouter Souffriau and Pieter Vansteenwegen</i>	
Personalized Tourist Route Generation	486
<i>Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau</i>	
Automated Generation of Itineraries in Recommender Systems for Tourism	498
<i>Pierpaolo Di Bitonto, Francesco Di Tria, Maria Laterza, Teresa Roselli, Veronica Rossano, and Filippo Tangorra</i>	
A Method for Assessing Website Communicative Efficacy Using a Semantic Annotation Tool	509
<i>Nadzeya Kiyavitskaya, Nicola Zeni, Cristina Coulleri, and Luisa Mich</i>	
A Process Framework for Semantics-Aware Tourism Information Systems	521
<i>Olawande J. Daramola</i>	

ICWE 2010 Doctoral Consortium

Use of Hypermedia Tools for End-User Development	533
<i>Sebastian S. Ortiz-Chamorro, Gustavo Rossi, and Daniel Schwabe</i>	
A Document-Centric Approach to Open Collaboration Processes	538
<i>Nelly Schuster, Christian Zirpins, and Stefan Tai</i>	
Description-Based Mashup of Web Applications	545
<i>Junxia Guo and Takehiro Tokuda</i>	

iSemServ: Towards the Engineering of Intelligent Semantic-Based Services	550
<i>Jabu Mtsweni, Elmarie Biermann, and Laurette Pretorius</i>	
Sustaining High-Availability and Quality of Web Services	560
<i>Erbin Lim and Philippe Thiran</i>	
Client-Side Adaptation: An Approach Based in Reutilization Using Transversal Models	566
<i>Sergio Firmenich, Silvia Gordillo, Gustavo Rossi, and Marco Winckler</i>	
QuEF (Quality Evaluation Framework) for Model-Driven Web Methodologies	571
<i>F.J. Domínguez-Mayo, M.J. Escalona, and M. Mejías</i>	
Consistent Cache Maintenance for Database Driven Websites	576
<i>Paweł Leszczyński and Krzysztof Stencel</i>	
Improvements of Webometrics by Using Sentiment Analysis for Better Accessibility of the Web	581
<i>Radek Malinský and Ivan Jelínek</i>	
Social Interaction with Cultural Heritage on the Web	587
<i>Max Arends and Doron Goldfarb</i>	
Author Index	593

Rapid Development of Composite Applications Using Annotated Web Services

Lars Dannecker¹, Marius Feldmann², Tobias Nestler¹,
Gerald Hübsch², Uwe Jugel¹, and Klemens Muthmann²

¹ SAP Research Center Dresden

Chemnitz Str. 48, 01187 Dresden, Germany

{lars.dannecker,tobias.nestler,uwe.jugel}@sap.com

² Technische Universität Dresden, Department of Computer Science,

Institute for Systems Architecture, Computer Networks Group

{marius.feldmann,gerald.huebsch,klemens.muthmann}@tu-dresden.de

Abstract. Developing service-based interactive applications is time consuming and nontrivial. Annotating web services with additional information about the user interface and behavior of the service promises to ease and accelerate the development process. In this paper we present a model-driven development approach that utilizes the advantages of annotated web services during the service composition. It enables a rapid development of simple service-based applications in a graphical manner. Besides the description of the process and the models, our current service composition tool implementation, the "ServFace Builder" is introduced. The ServFace Builder infers graphical UIs from functional interface descriptions such as WSDL files which then can be combined in a presentation-oriented way to a service-based interactive application.

Keywords: Service Composition, Model Creation, Service Frontends.

1 Introduction

Nowadays web services are an important means for developing distributed applications. However, developing user interfaces for web services is still carried out manually by software developers for every new service. This process can be expensive and error prone. UI elements have to be created, parameters have to be bound to input and output fields, code for invoking web service operations has to be written for one or several of the application's pages, and also many other UI events, such as list selection or scrolling, have to be handled. Currently, many of these steps cannot be performed automatically due to missing tool support.

This paper describes the internals of the ServFace Builder - a Rich Internet Application (RIA) development tool that integrates a model transformation from an annotated service description to an executable web application. This tool is capable to automate most steps in the development process of service-based interactive applications. In this paper, we show how the ServFace Builder reduces the effort to execute reoccurring development steps and how the targeted user group of domain experts and other non-programmers is supported by the tool.

The main concept of the ServFace Builder is the inference of user interfaces from functional service interface description files such as WSDL documents. However, the information within service descriptions do not contain sufficient information for generating useable UIs. A simple example for missing UI information is a human-readable, localized label of an input field.

To represent such information, the ServFace project applies the concept of service annotations. By using a specific authoring tool, ServFace Annotations can be attached to any node (e.g. an operation or a parameter) in a functional interface description to provide explicit information for UI integration. The annotations themselves are sufficient to generate usable UI parts of an application, but to integrate the parts into a bigger context an application meta-model for storing the overall application structure, additional logic for the UI and relations between different pages are necessary. Therefore we have developed the Composite Application Model (CAM). The ServFace Builder allows to transform annotated service descriptions via a model-to-model transformation to a part of a CAM instance. The CAM is the storage format and the input model for model-to-code transformations.

In this paper, we discuss an approach for presentation-oriented design of rich internet applications. In Section 2, we explain the benefit of annotated service descriptions and in Section 3, we describe a generic meta-model for describing interactive service-based applications. The ServFace Builder uses those ideas to allow end-user friendly web application development. The tool's annotation-driven, automatic UI generation creates a visual representation for service operations, which can then be combined in a presentation-oriented way (see Section 4). Additionally, the ServFace Builder supports development for several target platforms by using our generic meta-model and a sophisticated model-to-code transformation, described in Section 5. We finally evaluate all concepts and the tool in Section 6, using a university portal web application as a complex scenario.

2 Service Annotations

Our service composition approach is based on services that are described via the Web Service Description Language (WSDL) and enriched with UI-related *service annotations*. These annotations are reusable information fragments, which are typically not available for an application developer or service composer. They provide extensive additional information covering aspects of the visual appearance of a service (e.g., add a *label* or *group* parameters, *enumerations* for predefined values) the behavior of UI elements (e.g., client-side *validation* rules, *suggestion* of input values) and relations between services (e.g., *semantic data type relation*). The annotations are created by the service developer to provide additional information about the web service. Thus, they can be seen as a kind of knowledge transfer from the service developer to the service composer to facilitate the understanding, to enhance the automated UI generation and to simplify the composition of web services. Once defined the additional information provided by the annotations are available for all application developers that intent to use the annotated service.

3 Meta-model Overview

For representing pieces of information, required to build interactive service-based applications, the following three domain-specific meta-models have been developed. An overview of these models is given in the next paragraphs.

Annotation Model. The Service Annotations are stored in an annotation model based on a formally defined meta-model that references the elements within the WSDL. An in-depth discussion of the annotation model can be found in [1].

Service Model. To infer a UI for a service operation as a basis for the presentation-oriented authoring process (compare sec. 4), a tool internal representation of an annotated web service is necessary. For this purpose a specific meta-model is used, which is instantiated by a WSDL parser. Besides the information contained within the functional interface description, the model instances hold references to the service annotations represented as instances of the ServFace Annotation Model. The service model and its structure is described in detail in [2].

Composite Application Model. The service model is an integral part of a domain-specific meta-model named *Composite Application Model* (CAM). It describes the integrated services, their connections in form of data flows as well as the navigation flow of the entire application. The CAM is used as the internal model of the ServFace Builder and furthermore serves as the input for the generation of executable applications. The meta-model is not bound to a specific platform and thus can be reused for various target platforms. Concerning the CAM, a service-based interactive application is formed by a set of interconnected pages. Each page is a container for service front-ends as described in section 4.2. The CAM instance is continuously synchronized with authoring actions performed by the user (e.g., add page, integrate front-end, define data flow). The CAM is described in detail in [2].

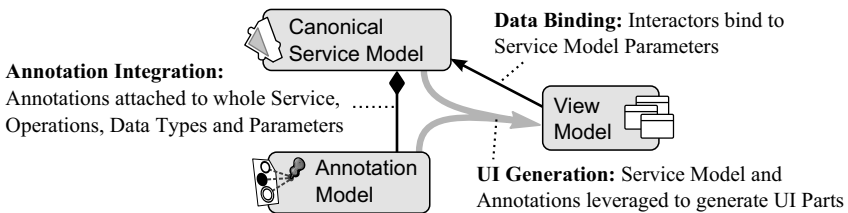


Fig. 1. Relationships between the models

4 ServFace Builder

The ServFace Builder (Fig. 2) is a web-based authoring tool developed in the course of the EU-funded project ServFace[1]. The graphical design-time tool utilizes

¹ <http://www.servface.eu>

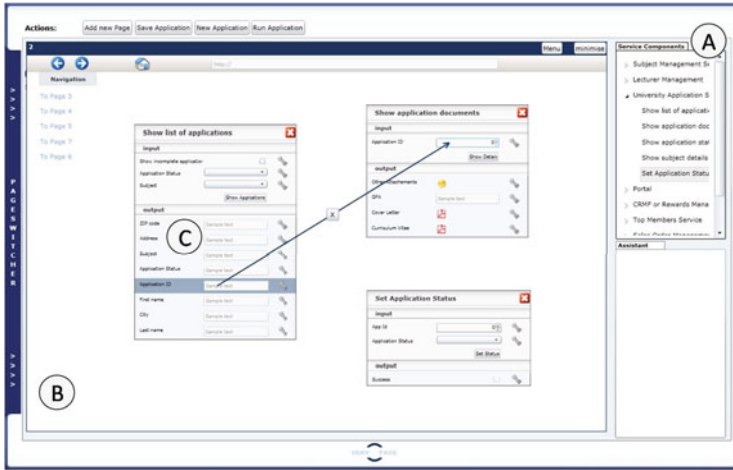


Fig. 2. The ServFace Builder

the described meta-models in order to realize a model-driven development of multi-page interactive applications for various target platforms and devices.

The tool applies the approach of service composition at the presentation layer [3] in which applications are built by composing web services based on their frontends, rather than application logic or data. It builds upon the concept of UI integration introduced by [4]. Our UI-centric composition approach aims to support non-programmers that are familiar with the semantics of their specific application domain. They understand the meaning of the provided service functionality without having a deep understanding of technical concepts like web services or service composition.

The goal of the graphical composition concept is to hide the complexity of the actual programming task by representing a service entirely by its corresponding UI and let the user only work with these visual representations in order to model all relevant aspects of the application development process. Thus, our approach fully abstracts from technical details in a presentation-oriented way. This enables people without substantial technical background to use our tool and create interactive applications based on web services.

The ServFace Builder consists of several components. The Service Browser (fig 2.A) shows all services available for the application creation process. The user creates the application on the Composition Area (fig 2.B) which represents a page of the final application. Each page contains the UIs of the added service operations (fig 2.C).

4.1 Composition Process

To create a multi-page interactive service-application with the ServFace Builder the user has to perform the following steps:

Select target platform: As the initial step, the user has to select the target platform from a set of predefined platform templates (compare sec. 4.4).

Integrate service operation: After the platform selection, a blank initial page is created and the actual authoring process begins. The user can integrate service operations from the Service Browser by dragging them to the Composition Area. The tool automatically creates the corresponding service frontend.

Define data flow: In order to define a data flow between two service frontends, the user has to connect UI elements of both frontends to indicate the relationship.

Define navigation flow: Frontends can be placed on one page or distributed over several pages to create a multi-page application. Pages can be connected in order to create a navigation flow.

Deploy application: After finishing the design process, the modeled application can be deployed according to the selected platform.

The following sections describe the particularities of the ServFace Builder in detail.

4.2 Automated UI Generation

During the design process each service operation is visualized by a generated UI that is called *service frontend*. The frontends consist of a nested container structure, which includes UI-elements like text fields or combo boxes that are bound to the corresponding operation parameters. Frontends are automatically generated by the *Visualization Engine* of the ServFace Builder when the user adds an operation to the application. The basis of the generation process is the associated service model that is retrieved from a remote repository containing all available annotated services. The engine parses the service model and analyses the elements of a particular service operation. For each service element that needs to be displayed in the service frontend, a proper UI element is inferred based on the class of the parameter (e.g. input or output), the basic data type, the data type configuration (e.g. enhancements, restrictions) and the cardinality. Complex data types are displayed as lists or tables in most cases. However, for complex types with a deeply nested or recursive structure the ServFace Builder provides a special widget. To enhance the visual appearance and the functionality of the service frontends, the ServFace Builder utilizes the information of the ServFace service annotations. Furthermore, we consolidated several HCI guidelines e.g. provided by companies such as Apple [5] or usability experts like [6] to form a set of UI design recommendations. They are used within the frontend generation to ensure a usable UI. Whenever the Visualization Engine finds annotations attached to an element of the service model, it uses an effect description for each annotation. This effect description specifies the effects the particular annotation has on the representation of a service operation. The effects are then incorporated into the UI element generation. During the generation process the current CAM instance is updated accordingly. Afterwards, the frontend is displayed to the user. The generation process that is depicted in figure 3 runs completely automatic in

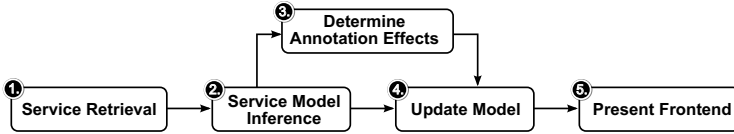


Fig. 3. The frontend generation process of the ServFace Builder

the background and without the necessity of further interaction, as the user just needs to add the service operation to the service composition. This simplifies and accelerates the design process of the application, since there is no need for a manual UI creation.

4.3 Presentation-Oriented Modification and Composition

After the frontend generation process has been realized, the user can modify several aspects of the frontends such as renaming or hiding elements. In addition, the frontends can be composed with other web service operations in a graphical manner to create the interactive application. This means that the user directly interacts with the input and output elements of the frontends to directly connect several service operations. A connection is created by a sequence of two clicks that identify the target UI element that is to be filled with data and the source UI element from which the data is to be taken. The data flows are visualized in form of arrows between the connected UI elements. The direct use of the UI elements benefits from the fact that form-based applications are well-known and most users are familiar with them. Furthermore, the user can create several pages to place the frontends on. All created pages are shown in a graphically structured overview and can be connected to model page transitions. The definition of the navigation flow is done in a graphical way as well. The user has to connect two pages in order to create a page transition. These presentation-oriented concepts distinguish our approach from other composition editors like Yahoo! Pipes² or the IBM Mashup Center³ that often use abstract concepts like ports and wiring as a depiction for a connection between service operations. In addition they focus on the processing of data, rather than the creation of service-based interactive applications.

The ServFace Builder interprets the changes done by the user automatically in the background and adjusts the affected parts of the CAM accordingly. Thus, the user creates an application in WYSIWYG (What you see is what you get) style without a need to write any code or manually change the CAM. This lowers the barrier to use the editor, even for inexperienced users.

4.4 Support for Different Target Platforms

Our approach allows users to create applications for different platforms. The ServFace Builder supports this by offering a selection of several target platform

² <http://pipes.yahoo.com>

³ <http://www.open-mashups.org/>

templates that adjust the CAM instance and the visualization of the UI to reflect the visual appearance of the application on the specific platform. This gives the user the possibility to directly adjust applications to the specifics of the target platform, e.g. limited space due to a smaller screen size. The templates conform to a pre-defined XML schema. This ensures the possibility to easily create new templates and therefore enhance the platform support of the ServFace Builder. An alternative to this approach would be to create an application in an abstract way, not knowing how it will look like on the target platform. Such applications can be transformed into executable files for any of the supported platforms. Because the look and feel of an application strongly differs for different platforms, the creation of an abstract application can lead to unexpected or even unusable results. In addition we adopt the WYSIWIG principal which makes a platform specific representation a requirement.

While the CAM as a meta-model is platform independent, its instances contain specific aspects and configurations of the chosen platform like concrete UI elements. This is necessary to ensure a proper model-to-code transformation. It is planned to enhance the ServFace Builder in the future to allow a switch between several platform templates at design time which results in the creation of multiple CAM instances and applications at the same time.

5 Transforming the Application Model into an Executable Application

The final results of the ServFace Builder are deployable service based interactive application packages, each possibly for several different platforms. In our research we concentrated on two kinds: Packages for the mobil platform in the form of Google Android and packages for the Web case using Java Enterprise Edition. The following paragraphs describe at first the general steps to create a running application from a CAM instance and then delve into some details of the Android platform.

5.1 Model-To-Code Transformation

Two contrary approaches are possible to create a running application from a CAM instance. It is possible to either have a platform specific interpreter that reads the CAM instance during runtime or to generate a complete application with a model-to-code transformation. Because a generated application is faster than an interpreted one, the ServFace Builder uses the generation approach.

The model-to-code transformation as shown in figure 4 takes instances of the three metamodels - CAM, Service Model and the Annotation Model - as input and transforms them to application packages. Two important findings were made during the implementation of the model-to-code transformation. The first was that there is very much static code, which stays the same for every generated application. Generating this over and over again produces much overhead during the generation phase. The static code results from the static structure of the three

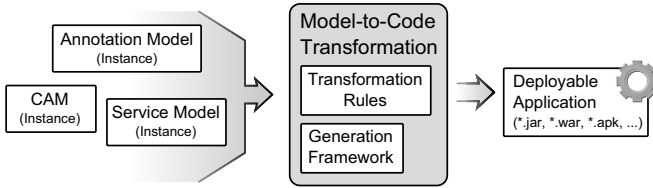


Fig. 4. Model-to-code transformation process

metamodels and thus represents their structure inside the code. To accelerate code generation and to simplify the generation rules, each metamodel was thus implemented as a static framework, which only needs to be instantiated upon generation time. The second finding is that platform specific information needs to be added at generation time as well. So the code generation rules and the implementation of the metamodel framework is platform dependent.

5.2 Android Platform

This section presents some issues with the runtime generation for the android platform as one example of supported platforms. Android applications are structured by `Activity` objects, which are screens a user can interact with. The ServFace Android generation framework and thus all generated applications are structured around a central dispatcher `Activity`. This `Activity` controls the control and dataflow by calling the correct `Activity` objects at the right time. Upon completion of its tasks every `Activity` returns control flow to the dispatcher, which then calls the next one. CAM Pages, since they can contain an arbitrary number of interactors, are split into several Android activities. These activities represent the page content tree from the CAM and the user can navigate along its edges to fill out input fields, invoke services and view output interactors. Annotations are resolved during the transformation.

The most complex part is the mapping of datatypes to interactors. Since webservices may accept and produce arbitrary values, the framework reads and displays these values on a best effort basis. This means for example if a list widget is selected to display a return value, the framework tries to convert the value to a list. If this is not successful it falls back to just displaying a string representation of the return value as the first entry of the list. The framework contains conversion rules for the most common object types for each widget and fallback rules for generic types (like `Object`).

6 An Example Scenario

The following sample scenario shows how the ServFace Builder can be integrated into web applications that are built upon annotated services. Furthermore, it demonstrates the potential and practical applicability of the concepts behind annotated services.

Our scenario is built around a web-based student portal through which a university gives students the possibility to organize their studies. Enrolled students receive a log-in for the portal by which they can manage their course and lecture subscriptions and view their timetable online (fig. 5a). The functionality of the portal is completely implemented by operations of a single Portal Web Service. As the number of students owning smartphones rapidly increases, the university decides to make subsets of the portal functionality available for these mobile platforms. Instead of implementing a mobile version at high cost that may not meet the requirements of all students, the university annotates the Portal Web Service and adds a link to the ServFace Builder through a logo on the portal's main page (lower left corner in fig. 5a), allowing students to build their own mobile applications based on the functionality of the Portal Web Service.

In our case, a student of psychology wants mobile access to her time table and details, e.g. to find out the room number where the next lecture is taking place, of the courses she is enrolled in. In the portal, she clicks on the logo and the ServFace builder opens (fig. 5b). She selects Google Android as the target platform for the new application. The ServFace builder shows the list of operations offered by the Portal Web Service. The student scans the list and identifies two operations: getTimetable, which returns the student timetable, including the id

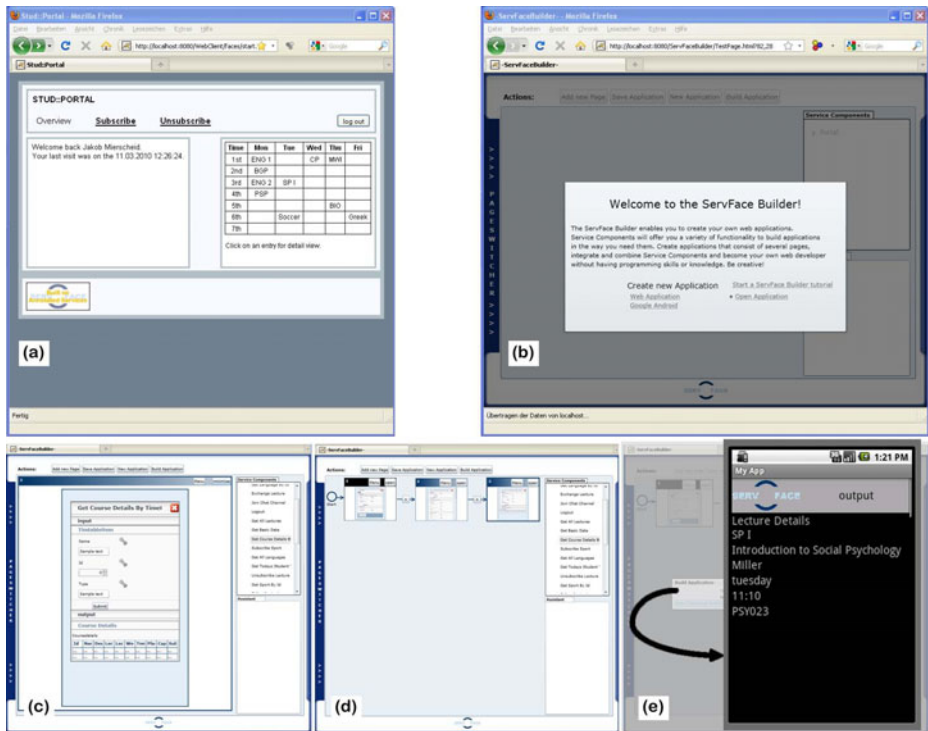


Fig. 5. University Portal Scenario

of each lecture given the student id, and `getDetails`, which returns lecture details given the lecture id, as the ones that meet her requirements. She now visually composes these operations using the `ServiceBuilder` by first dragging each operation on a separate page. Based on the annotations, a suitable UI is generated for each operation (fig. 5k). Also, the student is informed that she needs to add the login operation before she can use the others, creating a third page. In a next step, she defines the page flow (fig. 5l) and data flows based on input and output fields of the generated UIs. She links the input field for the `getDetails` operation to the output field of `getTimetable` that contains the course id. Another data flow is defined to fill the input field of the `getTimetable` operation that is already available from the login operation. She now clicks the 'Build Application' button of the `ServiceBuilder` which sends the `CAM` instance to the model-to-code transformation running on a server. The generation result of the transformation is a deployable Google Android application that integrates the Portal Webservice. Finally, a download link to the deployable Google Android application is presented (fig. 5e). After downloading and installing the application, the student can use a part of the portal functionality on her smartphone.

7 Related Work

Service Annotations. Our concept of service annotations tries to leverage existing work as much as possible and is partially based on former approaches, i.e., the GUIDD annotations [7] used by Dynvoker [8] and by the WSGUI project [9]. As a runtime engine applying the concept of annotated services for ad-hoc service usage, Dynvoker uses the GUIDD annotations to generate user interfaces for web services at runtime. Although these existing approaches already cover the generation of user interfaces for single web services, no solution is available for service composition and the usage of annotations for an application's design time.

Mashups. The so called Web 2.0 offers users the capability to build sophisticated web applications like mashups by combining distributed resources and contents into new *composite* web applications. Graphical *mashup platforms* (e.g. Open Mashup [4] and Yahoo Pipes [5]) aiming for a (partially) assisted web development for less skilled programmers or even non-programmers. However, mashups mainly focus on the data aggregation and still lack of concepts to create composite service-based applications by end-users ([10], [11]).

UI Integration. Research projects like CRUISe [12] or mashArt [11] build upon the fundamentals of UI integration and follow a comparable philosophy centered around the idea of event-based UI components. The three main differentiations of our work to both projects can be seen in the clear restriction to the presentation layer (1), the direct interaction with UI elements during the design time in

⁴ <http://www.open-mashups.org/>

⁵ <http://pipes.yahoo.com>

order to define all composition tasks on the UI level (2) and the focus on non-programmers and skilled web users as our main target group (3).

MDD of service-based applications. There already exist model-driven development tools that can be used for the creation of service-based applications such as UsiXML [13], EMODE [14] and Teresa/Maria [15]. These approaches apply a model transformation chain starting with abstract models and refining them to instances of concrete models, which are then transformed to executables or interpreted at run time. In contrast our approach of using annotations presents a loose coupling of model-driven development techniques and service annotations instead of a strong integration of abstract UI-models. In addition, our models are designed to be generic and reusable for applications on different platforms, which distinguishes our approach from solutions such as AndroMate [16] (Android only) and WebRatio [17] (Web Applications only).

8 Conclusion and Future Work

The paper presents a model-driven development approach for service-based interactive applications. Part of this approach is the ServFace Builder, a tool for the presentation-oriented composition of web services. The ServFace Builder automatically infers UIs for service operations. The user can combine these UIs via navigation and data flows in a visual manner to create a service-based application which is stored as an instance of a generic application meta-model called CAM. This instance serves as input for model-to-code transformations targeting different platforms. In addition to the already conducted end user studies (e.g. compare [18]), future work will concentrate, at first, on additional evaluations of the ServFace Builder's usability on a large group of real end users, and in different scenarios. Furthermore, it will be analysed how generated service frontends can be embedded into the context of already existing applications.

Acknowledgment

This work is supported by the EU Research Project (FP7) ServFace.

References

1. Janeiro, J., Preussner, A., Springer, T., Schill, A., Wauer, M.: Improving the Development of Service Based Applications Through Service Annotations. In: Proceedings of the WWW/Internet Conference (2009)
2. Feldmann, M., Nestler, T., Jugel, U., Muthmann, K., Hübsch, G., Schill, A.: Overview of an End User enabled Model-driven Development Approach for Interactive Applications based on Annotated Services. In: Proceedings of the 4th Workshop on Emerging Web Services Technology. ACM, New York (2009)
3. Nestler, T., Feldmann, M., Preussner, A., Schill, A.: Service Composition at the Presentation Layer Using Web Service Annotations. In: Proceedings of the Composable Web 2009 Workshop, ICWE (2009)

4. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing* 11(3), 59–66 (2007)
5. Apple Developer Connection: Apple Human Interface Guidelines. Technical report, Apple Inc. (2009)
6. Krug, S.: Don't make me think! A Common Sense Approach to Web Usability. New Riders (2006)
7. Kassoff, M., Spillner, J.: GUI Deployment Descriptor (GUIDD) Standard Specification and Documentation (2006), <http://inf.josefspillner.de/webservices/guidd.html>
8. Spillner, J., Feldmann, M., Braun, I., Springer, T., Schill, A.: Ad-hoc Usage of Web Services with Dynvoker. In: Mähönen, P., Pohl, K., Priol, T. (eds.) *ServiceWave 2008*. LNCS, vol. 5377, pp. 208–219. Springer, Heidelberg (2008)
9. Kassoff, M., Kato, D., Mohsin, W.: Creating GUIs for Web Services. *IEEE Internet Computing* 7(5), 66–73 (2003)
10. Ro, A., Xia, L.S.Y., Paik, H.Y., Chon, C.H.: Bill Organiser Portal: A Case Study on End-User Composition. In: Hartmann, S., Zhou, X., Kirchberg, M. (eds.) *WISE 2008*. LNCS, vol. 5176, pp. 152–161. Springer, Heidelberg (2008)
11. Daniel, F., Casati, F., Benatallah, B., Shan, M.C.: Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In: *Proceedings of the 28th International Conference on Conceptual Modeling*, pp. 428–443 (2009)
12. Pietschmann, S., Voigt, M., Rümpel, A., Meißner, K.: CRUISe: Composition of Rich User Interface Services. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) *ICWE 2009*. LNCS, vol. 5648, pp. 473–476. Springer, Heidelberg (2009)
13. Vanderdonckt, J.: Model-driven engineering of user-interfaces: Promisses, successes, failures, challenges. In: *Proceedings of ROCHI 2008* (2008)
14. Behring, A., Heinrich, M., Winkler, M., Dargie, W.: EMODE Model-Driven Development of Multimodal, Context Sensitive Applications. *i-com* 6(3) (2007)
15. Paterno, F., Santoro, C., Spano, L.: Designing Usable Applications Based on Web Services. In: *Proceedings of the Int. Workshop on Interplay between Usability Evolution and Software Development* (2008)
16. Ebben, P., Heerink, L., Reitsma, J., Steen, M.: Andromate project (2008), <http://www.lab.telin.nl/~msteen/andromate/>
17. Brambilla, M., Comai, S., Fraternali, P., Matera, M.: Designing Web Applications with WebML and WebRatio, *Web Engineering: Modelling and Implementing Web Applications*. Technical report. Springer (2007)
18. Namoune, A., Nestler, T., Angeli, A.D.: End User Development of Service-based Applications. In: *Proceedings of the Second Workshop on HCI and Services*, at HCI (2009)

From Mockups to User Interface Models: An Extensible Model Driven Approach

José Matías Rivero^{1,2}, Gustavo Rossi^{1,2}, Julián Grigera¹, Juan Burella^{2,3},
Esteban Robles Luna^{1,2}, and Silvia Gordillo¹

¹ LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{mrivero, gustavo, julian.grigera, esteban.robles,
gordillo}@lifia.info.unlp.edu.ar

² Also at Conicet

³ Departamento de Computación, Universidad de Buenos Aires
jburella@dc.uba.ar

Abstract. Sketching web applications with mockup tools is a common practice that improves the process of elicitation and validation of requirements in web applications. However, mockups are used as a “quick and dirty” way of gathering requirements, thus discarded before development. As a consequence, concepts captured in them are usually lost in the manual transformation between mockups and the final user interface. In this paper we present a model-driven approach that overcomes this problem by importing mockups and then transforming them into a technology-dependent model. Development then begins from the imported version of the mockups.

Keywords: Mockups, User-Interface, MDWE, TDD, MDD.

1 Introduction

Agile methods are appealing for Web applications because they help to provide quick feedback to customers, following short development iterations and involving them in the development endeavor. Being requirement elicitation and managing an important aspect of Web application development, most mature model driven web engineering approaches (MDWE) use informal textual descriptions (e.g. Use case or User story) to capture them. However, some requirements are not clearly understood and depend on the interpretation of the observer. Particularly, requirements related to interface and interaction issues that are paramount in Web applications are usually remain unchecked until the application has been partially developed.

Mockups have become a very popular artifact to capture requirements in agile methods. A mockup is a sketch of a possible user interface (UI) of the application that helps to agree on broad aspects of the UI and can be easily created by any stakeholder. In the last years, its use has been quickly expanded, generating a myriad of tools such as Axure [1], Pencil [2] or Balsamiq [3] that help to create and administrate mockups. Plain HTML can be also used to provide more detailed and realistic UIs. However, most development approaches use them informally and consequently as a

“quick and dirty” way of gathering requirements, not providing ways to reuse them in the development process.

In this paper we present a model-driven approach for reusing mockups, engaging them as part of either a MDWE process or even in a more handcrafted coding based approach. We first show how mockups can be created by using any of the aforementioned tools and then imported as instances of a metamodel. Then, we show how to transform these instances to produce either models or technology dependent code that can be used during the development process. Also, as part of our technology dependent transformations, we show how UI structure is separated from UI behavior to allow seamless evolution along the development cycles. In summary, the contributions of this paper are the following:

- We present a metamodel that abstracts a common set of mockups models and show how we can import mockup model instances.
- We show how instances of the mockup metamodel can be translated to different technology dependent concrete interface models.
- We show that the approach is easily extensible to other mockup tools and UI technologies.
- Finally, we show how we handle evolution in our approach.

The rest of the paper is structured as follows: in Sect. 2 we present some related work. In Sect. 3 we present our approach in detail. In Sect. 4, we show how we handle evolution and in Sect. 5 we show some implementation details. In Sect. 6 we show how we can easily extend the approach to use different mockup tools and new technology dependent transformations. Finally in Sect. 7 we conclude and present some future work we are pursuing.

2 Related Work

Several successful experiences with UI mockups in the context of agile development processes in industry have been reported in the literature. In [4, 5], the use of UI prototypes in conjunction with User stories in real projects was considered a key factor in the entire development process, and facilitated interaction with stakeholders and between different work teams with distinct roles. A similar approach is described in [6], where user interface mockups resulted useful for interaction between analysts, developers and customers in companion with summarized User stories.

At the same time many model-driven approaches to user interface specification have been proposed; in most of them UI definition starts by specifying domain objects or concepts like tasks or classes [7], thus neglecting the early capture of aspects like presentation and look and feel that are essential for customers and final users. In [8], a MDA-compliant model-driven UI environment has been defined in order to cope with the complexities of defining modern user interfaces. The approach presented in the paper consists in a framework allowing *multi-context* UI definition starting from a model belonging to one of four levels of abstraction, from task and concepts to final interface specification in a concrete technology. Once defined the model in a concrete level of abstraction, the framework facilitates transformations to any other level of abstraction, including the less abstract of all: the *Final UI (FUI)* implementation.

Also, transformations between models at different *contexts of use* like, for example, desktop or Pocket PCs, are provided. In the context of the paper, the approach presented in our work depicts a *transformation path* from mockups to an abstract mockup model similar to a *Concrete UI (CUI)*, and then *reification* (a decreasing abstraction transformation) to a *Final UI (FUI)* for supported technologies.

We have tested and analyzed several mockup UI tools such as Axure, Pencil, GUI Design Studio [9] and Balsamiq to define our UI mockup metamodel concepts. Despite some of these tools provide interaction or behavior specification, the metamodel proposed in this paper only tackles structural UI aspects, in order to support the vast majority of mockup tools.

3 Out Approach in a Nutshell

In order to introduce the approach, we will show how we use it in the context of a WebTDD [10] process, though it can be also used with other development styles such as RUP based processes [11] or even Extreme Programming [12]. Nevertheless, we are pursuing better automation and assistance for development tasks, with some additions and enhancements further commented in Sect. 7.

In WebTDD, requirements are captured using mockups and WebSpec [13]. The approach presented in this paper crosscuts two important steps of the WebTDD process: Requirements gathering and Implementation (UI derivation) (Fig. 1).

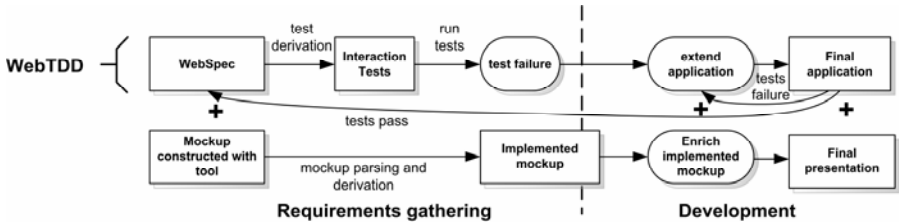


Fig. 1. Our approach's process

During the requirements gathering steps, the approach is used to import mockups and derive them to a concrete technology so that customers can check on a preliminary version of the application's interface. During the development phase, the UI of each requirement can be implemented directly from the "derived" version of the mockups, bridging the gap between the original mockups and the final presentation.

The approach proposed in this paper (detailed in Fig. 2) consists in using a metamodel that helps to abstract mockups in a tool-independent way. A collection of mockup parsers (1) are provided for each tool, allowing a concrete mockup conversion from each one into our metamodel. When the mockup parsing ends, a post-processing is performed (2) to rearrange the parsed UI controls, and the final abstract mockup model is then obtained. Once the whole importing process finishes and the complete abstract mockup model is instantiated (3), it can be used to derive stub UI classes/models implemented with a concrete technology (4) - similar to what mockup tools like Axure provide. For each technology of interest, a code generator can be

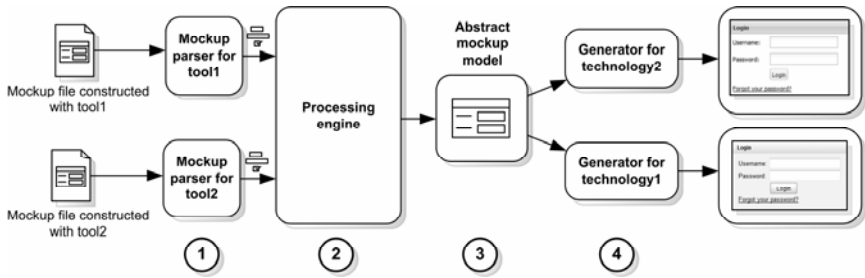


Fig. 2. Our approach in action

constructed to transform mockup models to concrete UIs able to run in a browser. This code generator will choose the most appropriate widgets in the particular web technology to implement the respective UI components present in the abstract mockup, deriving the code that constructs and configures each one.

3.1 Importing Mockups

The mockup importing process starts with individual UI component detection and parsing from a mockup file created with a specific tool (Fig. 2, step 1). Since most of mockup tools focus in user interface sketching, unlike common UI frameworks or technologies they don't provide ways of defining natural UI control composition. Nevertheless, our metamodel contemplates this kind of composition in order to derive complete UI specifications for concrete technologies. Thus, a mockup parser for a concrete tool must scan a mockup file composed with the tool and return a collection of metamodel controls representations grouped in "clusters". Each "cluster" represents a set of components in a unique graphic space (for example, a page, a window or another UI control grouping concept that a mockup tool could define).

Once the UI control groups are obtained, a post-processing over each particular group is performed. The first processing task performs hierarchies detection: if a UI control is graphically inside another and the first one is a composite control, the second one is added as a child of the first. Finally, the controls are grouped in *Pages*.

At this phase, a UI model is composed of a set of *pages* containing a hierarchy-arranged collection of controls each one. Because of the myriad of different Web UI technologies implementations, an absolute positioning scheme is not sufficient to model a user interface in a platform-independent way. To avoid this problem, the components are arranged into a specific platform-independent layout used later to derive positioning information in the code generation phase through application of layout inference algorithms.

After this last post-processing task, the complete UI specification model with a platform-independent layout configuration is obtained. This abstract and formal UI description can be used later for UI platform-specific specifications derivation.

3.2 Automatic UI Model Derivation

Models obtained in the parsing phase can be derived to different concrete implementations using the previously mentioned code generators. The derivation process consists in

iterating through each model element in a Visitor-like process [14] and generating an intermediate representation of the UI. This representation is used to derive the final UI code that will become part of the web application from the original abstract mockup model.

The derivation also considers the generation of code that assigns a unique identifier to every UI component being constructed in the concrete technology. This feature is fundamental in order to reference user interface controls from manually written code that implements the UI behavior or enhance the derived components features. Finally, code generators derive the initial structure of code files that can be used to enrich generated UI components or add behavior. The separation of derived source code that captures the representation of the mockup in the concrete technology and the manually written source code is paramount to isolate UI interaction or detailed specification from automatically generated user interface code and structure.

4 Evolution

We introduced our approach in the context of agile methodologies, because mockups are commonly used in those as a requirements gathering tool. Since agile processes are characterized as iterative, with short development cycles and continuous interaction with final users and stakeholders, changes in the UI specification are common, resulting in a great number of user interface iterations through the complete development process. Thus, the evolution of the UI is an important concern to be considered in our approach. An extremely simple example of this problem can be observed in Fig. 3. The Balsamiq mockup shows a first version of a login screen that has a simple change to fulfill a new stakeholder requirement.

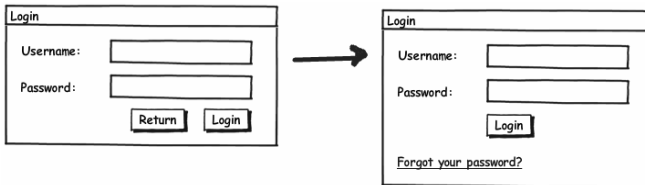


Fig. 3. A UI mockup change example

The possibility of enriching the generated mockup implementations without modifying the generated source code, allows a clear separation between UI presentation and behavioral concerns. Consequently, if the structural UI definition doesn't change, modifications to be made in behavioral aspects as a result of the application's natural evolution can be done isolated, therefore simplifying evolution.

Structural changes in UI mockups present a more complex problem, because modifications in the user interface structure could cause invalid or obsolete references to components that have changed or have been deleted in a new iteration. In the presented example, a change in the UI is proposed where a button is deleted and a link is added. This modification implies that the behavior code referencing the deleted "Return" button is no longer valid in the new version of the UI and consequently should

be removed or changed. Additionally, since all the behavior code is maintained, actions associated to the new added link should be coded manually.

In order to solve these problems, an indirect way of referencing UI components during the implementation is proposed. The solution we propose consists in the construction of an identifier translation function to associate natural name identifiers for UI components to native ids assigned by the code generators; we call this function a *reference translator*. This approach has two advantages. First, the implementation code references UI controls through natural identifiers as in pure manually coded Web applications. Besides, if in some future iteration an existing component identifier assigned by the code generator changes, a simple tuning in the *reference translator* is sufficient to solve the reference problem. Changing a mockup source file could imply one of three possible actions from an individual UI component point of view: (1) a new UI control is created, (2) an existing UI control is preserved and possibly modified or (3) an existing UI control is deleted in the new iteration. The first one presents no complexity since it doesn't invalidate any behavior implemented in code. The second could entail a reference problem if the automatically generated UI component identifier changes from the previous iteration. In this case, a correction to the *reference translator* solves the problem. Finally, the last case implies behavioral changes and, therefore, the manually written source code should be changed to reflect the interaction semantics of the new UI.

Because of the non-predictable nature of UI control id generation, the best solution is to redefine the *reference translator* after each iteration. Fig. 4 shows how a UI change like the one expressed in the example of Fig. 3 can be handled with preservation of manually added behavioral code using the *reference translator* approach.

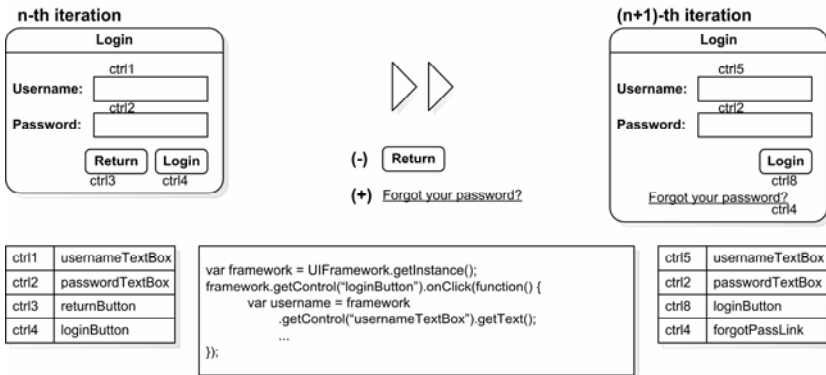


Fig. 4. Code preservation through mockup UI evolution using a *reference translator*

In the context of the MDD architecture vocabulary [15], the commented *reference translator* could be considered as part of a *domain framework* for a concrete implementation platform. All access to the UI structure should be done using the provided framework in order to write portable code between iterations.

So far, the presented approach solves the reference consistency problems between automatically generated UI and manually coded behavior when new UI components

are added or existing ones are modified between iterations. However, the control deletion is still a problem, since some of the controls referenced in the code written by hand could no longer exist in the new UI version. Due to the potentially complex logic implemented in behavior code, it is very difficult to apply an automatic and safe refactoring that removes all the operations implied by a deleted control.

Nevertheless, the problem can be solved in a test-driven way. When the first UI iteration is generated, a test is written to check that the structural requirements are satisfied. When a mockup is changed and a new UI version is obtained, the user interface structure test is run after the respective changes were done in the *reference translator*. If the test fails, the handwritten code should be changed to fit the new UI structure. On the other hand, if the test passes, it shows that the current handcrafted code will work with the new user interface structure, but its semantic could be obsolete or invalid. In both cases, the test should be updated in order to assert the new UI structure features. An outline of the UI iteration life cycle can be observed in Fig. 5.

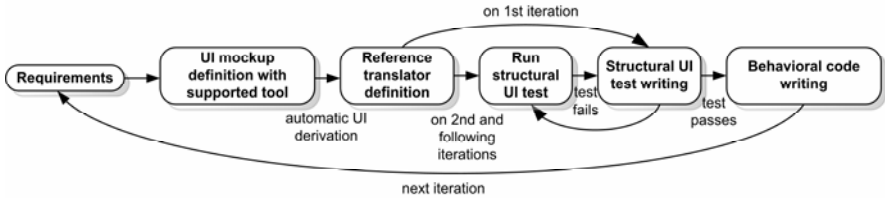


Fig. 5. UI life cycle

In conclusion, in handcrafted code approaches, on the first UI iteration, three artifacts are derived from abstract mockups specified with our metamodel: the user interface structure, a first version of the *reference translator* and an initial code behavior structure. To obtain the first functional UI version, the initial behavior must be written after assigning natural ids to those automatically generated in the *reference translator*. Also, user interface structural test writing in this phase is encouraged in order to reflect the initial UI structure requirements. After each iteration, the UI structure and *reference translator* are regenerated, but the behavior code and structural test are maintained and should be updated accordingly.

Assuming that the mockup tool assigns and maintains a unique id for each widget in the mockup (as Balsamiq does), it can be stored in metamodel UI components (as is shown in Sect. 5). Once stored, this id can be used to trace changes made between mockup iterations through metamodel instances comparison, thus facilitating or even avoiding some of the manual steps commented later.

5 Implementation

In Fig. 6, the structure of the proposed metamodel derived by finding and abstracting common features between the set of analyzed tools is denoted. An abstract mockup model has a root object which is an instance of the `MockupModel` class. A `MockupModel` is composed by one or more *pages* (`Page` instances), which in turn

contains a collection of `UIControls`, each one representing a UI mockup component. An UI control has position and size information and can belong to one of two classes: `CompositeControl` and `SimpleControl`. The first ones work as a container of another `UIControls`, while the second ones are atomic UI components like buttons, links and textboxes. The set of UI controls types included in the meta-model are those present in all the observed mockup tools.

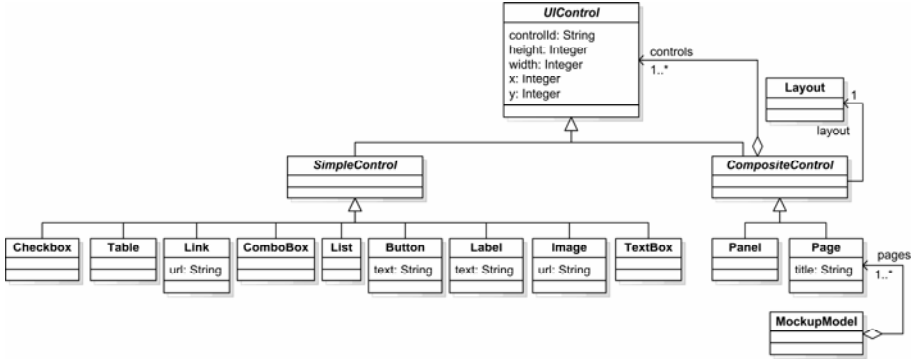


Fig. 6. The core mockup metamodel

Additionally, the metamodel considers different layouts that can be associated to any composite control (including Pages) to arrange its inner UI controls. As shown in the figure, the layout is a separated concern and does not mix with the UI definition. We modeled three different kinds of layout, inspired in some `LayoutManager`s present in Swing [16]: `FlowLayout`, `BoxLayout` and `GridBagLayout`. A `FlowLayout` does not sort the UI controls in any particular way, but simply puts one after another in an arbitrary manner. A `BoxLayout` aligns UI controls in a vertical or horizontal sequence. Finally, a `GridBagLayout` arranges components in an HTML table-like way: each UI control is placed in a particular row and column of a grid, and is extended for a concrete numbers of columns and rows right and down respectively. As said before, this layout was the one chosen to be used as default because of its richness and flexibility, and an iterative algorithm is applied over the parsed widgets in order to configure it. This algorithm starts with a `GridBagLayout` with 1 row and 1 column, and tries to put the components in a particular cell depending on its relative position in its parent widget. If more than one component is assigned to a cell, a row or a column is added and the process starts again. To choose between adding a new row or a new column, the algorithm detects in which direction the colliding components are closer, and chooses the option that promotes better widget isolation. A graphically representation of the process execution can be appreciated in Fig. 7. Similar algorithms can be applied to infer another layouts like, for example, `BoxLayout`s.

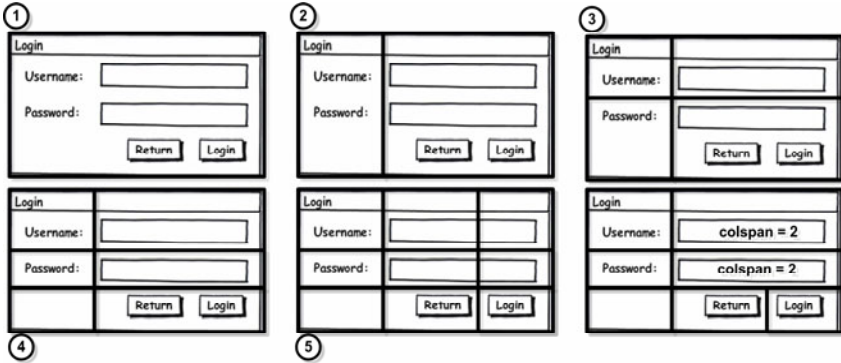


Fig. 7. GridBagLayout inference algorithm execution

We claim that the proposed metamodel is sufficient to specify the structural composition of a large set of common UIs. Likewise, with the addition of layout-oriented composite widgets supported by existing mockup tools like tab or accordion panels, an even wider set of UI could be expressed. To preserve generality, the metamodel limits the existing UI widgets to those which are commonly present in most popular mockup tools and avoids the specification of advanced aspects like behavior or interaction. Since only static aspects of an UI are considered, it is important to allow the derived models to be extended in several ways. As has been said, the generated implementation assigns an id to every derived UI control that can be used to obtain a reference to the component in runtime and manipulate it, for example, attaching event listeners to add interaction to the mockup. If some features of one or more user interface components in a concrete technology only can be specified at construction time, the *domain framework* provided allows defining them manually in a separated source code file, avoiding changes in automatically generated construction code.

As a proof of concept, we successfully developed mockup model parsers for the mockup tools Pencil, GUI Design Studio and Balsamiq, and constructed code generators for YUI [17] and Ext JS [18] Web technologies. In the context of MDWE approaches, we are implementing transformations to convert instances of our metamodel to UsiXML Concrete UI (CUI) models [8] and WebRatio templates [19]. With this addition, the model-driven approach presented in this paper can be linked to other MDWE and model-driven UI methodologies and thus be combined with them.

6 Framework Extensibility

In the following subsections we show how we can extend our approach with a new mockup tool and a new derivation to a technology dependent framework. Once defined a mockup parser for a new mockup tool or a code generator for a new concrete technology, it can be easily plugged in our framework.

6.1 Adding New Mockup Tools

Adding support for a new tool in the approach consists in implementing an interface that receives a mockup source (e.g. a file) and returns a collection of controls separated in independent graphically spaces or “clusters”. All the subsequent post-processing is performed with existing software components, thus allowing the reuse of all the mentioned tasks for any new parser that could be defined in the future.

In Fig. 8.a we show a class diagram of a mockup translator structure. A `MockupTranslator` is the class responsible of taking a concrete mockup source (the generic type `TSource`, usually a `File`) and translating it into an instance of our metamodel (`MockupMetamodel`). An instance of a `MockupTranslator` is configured with a concrete `ControlParser`, which implements the UI control parsing and grouping for a concrete mockup tool. Additionally, an instance of `ControlParser` is configured with a concrete `MockupMetamodelFactory`, which provides methods for constructing metamodel elements for a particular representation of our metamodel (e.g., a memory-stored one).

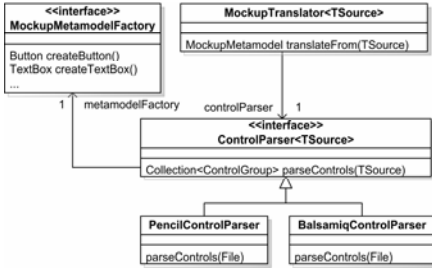


Fig. 8.a. Mockup translator structure

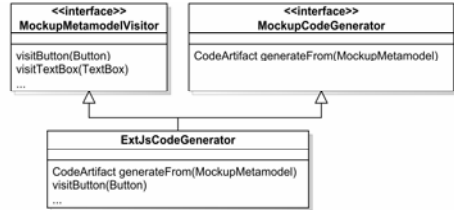


Fig. 8.b. Code generator structure

The fact that the architecture expects from the mockup parsers only individual control parsing and spatial grouping implies a few assumptions in the features of the source mockup format and, consequently, in the mockup tool used to construct it. Hence, a mockup tool capable of drawing variable-size UI components and supporting the set of user interface elements included in the metamodel is sufficient to compose a complete mockup translatable to our metamodel.

6.2 Adding New Technology Derivations

Extending the derivation scheme to add support for a different Web technology requires the implementation of an interface that receives an instance of our metamodel and derives a collection of specifications which will be translated later by our framework into specific implementation artifacts as mentioned before. The framework implements an extensible code generation library that resolves some usual code generation concerns like indentation and files creation and also provides an OO way of defining and composing generated code artifacts. The idea behind the construction and use of that library was having a non-restrictive and pure object-oriented approach to define and compose textual representations generation.

The structure of a common code generator using our framework can be observed in Fig. 8.b. Any code generator defined must implement the `MockupCodeGenerator` interface in order to be plugged into our framework. The `CodeArtifact` class is defined by the code generation library and generalizes and abstracts some aspects relative to the generation of textual code artifacts. Additionally, the interface `MockupMetamodelVisitor` defines methods that must be implemented in order to apply the Visitor design pattern for code generation, and thus its implementation is encouraged. Adding support for MDWE and model-driven UI approaches implies the definition of code generators for deriving textual model representations as those generated by MDD tools to serialize models into files. The derived model could be later imported in the respective model-driven tool and thus can be used in the context of its associated MDD methodology.

7 Concluding Remarks and Further Work

In this paper we outlined a model-driven approach to represent UI mockups allowing to import them from existing sketching tools and generating code for different modern Web technologies. The implemented software allows translation from mockups constructed with any of the mockup tools supported to any Web technologies, using our mockup abstract model representation as a “pivot”. The architecture of the framework constructed implies a small amount of work to add support for new mockup tools or Web technologies. Additionally, we show how the derived code for Web technologies can be enhanced to obtain final implementations and handle application evolution, allowing mockups being a reusable software specification artifact. Finally, the automatic model-to-code translation promotes a uniform UI design style, avoids common manual coding errors [20] in UI implementation and problems related to presentation discrepancies between browsers in web applications.

The construction of a tool that facilitates the introduction of refinements to the parsed UI models (e.g. widget composition) and also captures changes made in metamodel instances in order to apply refactorings and changes necessary to reflect them in the underlying implementation represents a potential future work. More detailed UI specifications surely should be needed in real world cases as can be seen, for example, observing modern interaction patterns and UI widgets in Web applications [20]. Adding support for more mockup tools, Web technologies, MDWE and model-driven UI methodologies represents a fruitful field for future work.

Finally, linking our UI metamodel with others metamodels oriented to domain, behavior, data transformation and process workflow definition could enhance model instances semantics, resulting in more code artifacts able to be automatically generated. However, while these semantic increments could provide a more MDA-based process, how to link the derived code with code-based agile methodologies remains being a challenge and represents some important future work we are pursuing.

References

1. Axure, <http://www.axure.com>
2. Pencil, <http://www.evolus.vn/pencil>
3. Balsamiq, <http://www.balsamiq.com>

4. Ferreira, J., Noble, J., Biddle, R.: Agile Development Iterations and UI Design. In: AGILE 2007, pp. 50–58. IEEE Computer Society, Washington (2007)
5. Noble, J., Biddle, R., Martin, A.: The XP Customer Role in Practice: Three Studies. In: Agile Development Conference (ADC 2004), pp. 42–54. IEEE Computer Society, Salt Lake City (2004)
6. Ton, H.: A Strategy for Balancing Business Value and Story Size. In: Agile 2007 Conference, pp. 279–284. IEEE Computer Society, Washington (2007)
7. Lu, X., Wan, J.: Model Driven Development of Complex User Interface. In: The MoDELS 2007 Workshop on Model Driven Development of Advanced User Interfaces. CEUR-WS. (2007)
8. Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 16–31. Springer, Heidelberg (2005)
9. GUI Design Studio, <http://www.carettasoftware.com/guidesignstudio/>
10. Robles Luna, E., Grigera, J., Rossi, G.: Bridging Test and Model-Driven Approaches in Web Engineering. In: Gaedke, M., Grossniklaus, M. (eds.) Web Engineering. LNCS, vol. 5648, pp. 136–150. Springer, Heidelberg (2009)
11. Kruchten, P.: The Rational Unified Process: an Introduction. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (2003)
12. Beck, K.: Extreme programming explained: embrace change. Addison-Wesley Professional, Reading (2000)
13. Robles Luna, E., Garrigós, I., Grigera, J., Winckler, M.: Capture and Evolution of Web requirements using WebSpec. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 173–188. Springer, Heidelberg (2010)
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, Reading (1994)
15. Tolvanen, J., Kelly, S.: Domain Specific Modeling: Enabling full code generation. Wiley-IEEE (2008)
16. Using Layout Managers – The java Tutorials, <http://java.sun.com/docs/books/tutorial/uiswing/layout/using.html>
17. YUI Library, <http://developer.yahoo.com/yui/>
18. Ext JS – Javascript Framework and RIA Platform, <http://www.extjs.com/>
19. Acerbis, R., Bongio, A., Butti, S., Ceri, S., Ciapessoni, F., Conserva, C., Fraternali, P., Toffetti Carughi, G.: WebRatio, an Innovative Technology for Web Application Development. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, p. 779. Springer, Heidelberg (2004)
20. Pastor, O.: From Extreme Programming to Extreme Non-programming: Is It the Right Time for Model Transformation Technologies? In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 64–72. Springer, Heidelberg (2006)
21. Mahemoff, M.: Ajax Design Patterns. O’Reilly Media, Sebastopol (2006)

Model-Driven Web Engineering Performance Prediction with Layered Queue Networks

Alessio Gambi¹, Giovanni Toffetti¹, and Sara Comai²

¹ University of Lugano,
6904, Lugano, Switzerland
{alessio.gambi, toffettg}@usi.ch

² Politecnico di Milano,
20133, Milan, Italy
comai@elet.polimi.it

Abstract. This position paper describes an approach to predict the performances of a Web application already in the early stages of application development. It leverages the wealth of information of MDWE solutions to automatically obtain accurate representations of the running application in terms of layered queue networks (LQNs), i.e., analytical models simulating the behavior of the system and computing the performances mathematically. In particular, the paper discusses how a MDWE methodology can be exploited to generate such performance models and presents a proof of concept example.

Keywords: layered queue networks, performance, web engineering, model transformation, capacity planning, WebML.

1 Introduction

Being able to correctly size the resources needed by a Web application to its incoming workload is the ultimate goal of performance engineering [4,6]. For a company, both over- and under-provisioning turn into revenue loss: in the former case because of higher than needed operational costs, in the latter case because the Quality of Service is below clients' expectations. The current practice in performance engineering impacts on different phases of a Web application development cycle: early in the process, coarse high level specifications are used at design-time to build mathematical models (e.g., queue networks) to estimate system performances; as the application reaches a complete implementation, staged experiments provide numeric parameters for the performance models or direct measures of the complete system behaviour. Updating the mathematical models to reflect the design choices and implementation details that happen along the development cycle is a costly manual process and very seldom done, as a result performance models tend to be misaligned with the final system and their predictions are less accurate.

In this paper we propose an approach to leverage the wealth of information of MDWE solutions to automatically obtain accurate representations of the running applications in terms of layered queue networks (LQNs). The advantages of this approach with respect to the common practice are manifold: 1) formal models allow for the generation of LQNs at different levels of granularity and detail, from very coarse grained all the way to the actual code produced by model transformation, 2) they do not require manual intervention to be kept in synch with the development process, 3) any model update can be reflected directly in LQNs providing immediate feedback on expected performance degradation/improvement. These advantages are particularly significant in the context of Web engineering, since most Web applications are constantly under maintenance and evolution.

The paper is structured as follows: Sections 2 and Section 3 provide an overview of performance engineering and LQNs; Section 4 discusses how a MDWE methodology can be exploited to generate performance models. Section 5 presents a proof of concept example. Finally, Section 6 presents related work and Section 7 draws the conclusions.

2 Performance Engineering

Performance engineering deals with all the tools, practices, roles, and activities, that are applied to a system throughout its life-cycle in order to meet a set of non-functional (performance) requirements.

The first step of performance engineering consists in the identification of a) a set of performance objectives for a system given a well-specified set of running conditions (e.g., response time for operation X will be within 2 seconds 95% of the time under a median load of 100 concurrent clients) and b) the methods and plans to test them at staging time and monitor them in production. Then, design and development are conducted following performance and monitoring requirements (e.g., by considering task parallelism, by adding instrumentation to the code for monitoring, etc.). At the end of the design phase, an high-level representation of the main components can already be used to do preliminary performance analysis.

Analysts normally utilize a combination of analytical model, simulation model, and empirical analysis based techniques [1]. Analytical models can be used in the early phases of development, and they represent the system behaviour by means of mathematical equations. The solution of these equations results in the values of the performance metrics. Analytical models are generally fast to solve, however the set of the underlying simplifying assumptions and the tuning of their parameters greatly impact on their accuracy. Simulation models provide more accurate results but at higher costs. Those models represent the system behavior as a chronological sequence of events, where each event occurs at an instant in time and changes the state of the system. During the simulation statistical techniques are used to compute the values of performance metrics. Finally, empirical analysis returns the most accurate performance measures, since the actual system implementation is stressed with different workloads.

In this work, we concentrate on analytical models (in particular LQNs) and propose a wider adoption of these performance modeling solutions in the context of MDWE. The rationale behind our proposal is that, given a knowledge or representation of the code generation mechanism, the design phase in MDWE already provides a complete representation of the final system that can be combined with a deployment description to produce accurate performance models.

3 Layered Queue Networks

Queue network (QN) models are one of the main tools supporting performance engineering and are used to determine (as average values) important metrics such as *response time*, *system throughput*, and *resource utilization* [1]. QNs represent systems as collection of resources, called service centers, that multiple clients need to access. Clients compete to access resources and their contention is represented by means of queueing before the service center. The time spent by a service center to complete a client request is called service time (s), and the idle time spent by a client between two different requests is called think time (z). In real systems, a client request needs to access multiple resources, so requests in the model go through different service centers and queues. For example, an HTTP request for a dynamic page starts from the **Browser**, and it may lead to several requests to the **Front End** and the **Back End** servers. This example is depicted in the form of a sequence diagram in Figure 1-a. Notice that, during the computation, the browser is blocked waiting for the front end while the front end is waiting for the back end; this situation impacts on the final service time and it must be taken into account while computing the performance metrics.

This situation can be accurately modeled by means of a particular kind of QN model, called the layered queue network (LQN) [11]. LQN models extend plain queue networks and combine architectural information with consumption and load data; in this way, LQNs can capture the nesting of resources in the

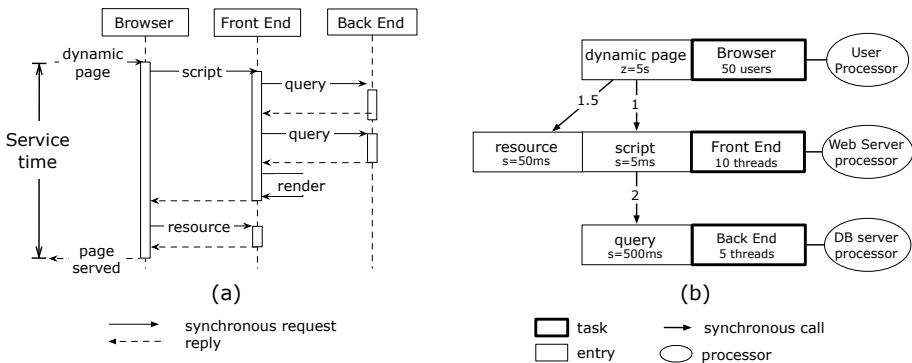


Fig. 1. A sequence diagram modeling a dynamic page request (on the left) and an LQN that models it (on the right)

system and their effects on the performance metrics. In the LQN notation, service centers are represented by **tasks** that may offer different services (or operations), which are modeled by means of **entries**. Each task has its own queue that is shared among the all requests for its different entries, and it is associated to a **processor**, which models the “physical” entity running it. Tasks may specify a multiplicity value to model their *multi-server* nature, and entries specify the service demand, which is the cost of running them on the associated processor. If an entry models a user activity then the think time is specified instead of the service demand. Entries can request services to entries of other tasks (represented by **calls**); calls may be synchronous (and therefore blocking) and can specify the mean count of requests per operation.

The LQN in Figure 1-b models a multi tiered system used to provide dynamic pages. The LQN contains three tasks, **Browser**, **Front End** and **Back End**, that are deployed on three different processors, respectively **User**, **Web server** and **DB server** processors. There are 50 concurrent users that request generic **dynamic pages** to the browser, with an average think time of 5 seconds. At the Front End task, each user request generates an average of 1.5 blocking requests for static **resources** and 1 blocking request for the dynamic content (the **script** entry) . Static resources are entirely served by the Front End, while to complete the **script** entry, 2 blocking requests are additionally issued at the Back End (**query** entry). The LQN specifies also multiplicities and service demands.

Layered queue networks are a powerful tool that can support the different phases of performance engineering. They can describe systems at different levels of abstraction: for example, tasks can be used to model software entities and to find *software bottlenecks*; entries can be used to model aggregate requests, at page level, or they can be further refined using **activities**; activities allow to model complex flows with loops, forks, and other constructs. For example they can be used to model accurately all the steps that an application server performs while building and rendering a dynamically generated page.

4 Model-Driven Performance Prediction

In this paper we advocate the usage of MDD for the generation of performance models at different levels of accuracy. For the sake of the exposition, we will use the Web Modeling Language (WebML) [2] as a reference language, since it is both well known in the community, and its runtime architecture is fairly straightforward. We stress however, how the process is general and can therefore be applied to any MDWE methodology.

As introduced in the previous section, a LQN uses primitives representing both the user behavior and the high-level system internal functioning. In particular, considering their main drivers, we can separate a LQN in different parts:

1. The tasks, entries, activities and request types that depend on the design of the application and architecture;
2. The hosts and tasks multiplicity that depend on the final deployment;
3. The think time and average request mix that depend on the user behavior;

4. The resource usage and configuration parameters that can either be obtained by system experts or measured with specific experiments on the system implementation.

In the following subsections, we discuss how, given a Web Engineering model with formal semantics and a known architecture (like, e.g., WebML/WebRatio), various queue models focusing on different aspects can be automatically obtained; an example of LQN derived from WebML/WebRatio will be provided in Section 5.

4.1 LQN Modelling Choices for the Model/Architecture

Figure 2 gives a high-level representation of the runtime architecture of a WebML application in WebRatio [2]. On the left-hand side of the figure, users make HTTP requests through their browser. HTTP requests are served by a Web server (e.g., Apache): static resources are directly served, while dynamic pages are forwarded to a load-balanced pool of application servers or servlet containers. Each servlet container runs an implementation of Struts, a common MVC2 framework for the Web. The framework controller forwards the requests to a specific action (page or operation), which in turn invokes the WebRatio runtime components (RTX) to compute the page along with its content units by querying one or more databases.

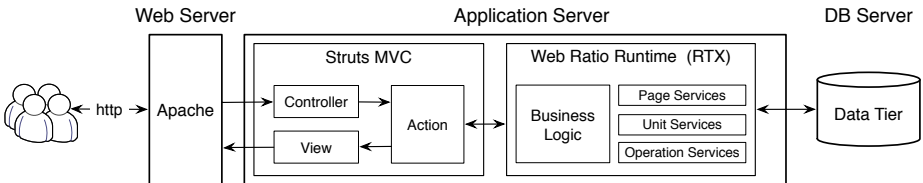


Fig. 2. High-level representation of the architecture of a WebML/WebRatio application

The previous architecture would be enough to create a simplified queue model reflecting the system layers, but delving deeper into the architecture's internals can provide more useful information. To serve dynamic WebML pages the RTX Page Service performs a different computation sequence depending on the link navigated by the user (for example, Link X in Figure 3). The page computation populates all the content units along with a link-specific parameters propagation to provide the correct query parameters to the database. Hence, each page computation is different and requires a computation time that is at least proportional to the number of units it contains. Our practical experience shows that the number of content units alone is a good metric of the response time of a dynamic page as this relates to the load at the database plus the waiting time for querying it.

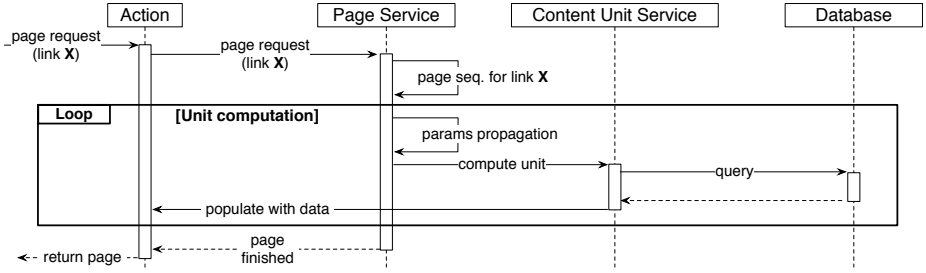


Fig. 3. Simplified sequence diagram of page computation

Since each link may trigger a different page computation sequence, even at the level of a single page the system may face a different load depending on the navigated link. For example, in WebML some units are not populated until they receive parameters through a *normal* (i.e. non-automatic) link, thus their computation does not always involve database operations. The model-to-model transformation to translate WebML to LQNs models can be summarized as follows: to model the general WebML/WebRatio architecture in LQN terms, we define the **App Server** task, representing the servlet container, and the **DB server** task, i.e., the data tier logic. Both tasks are application independent: they would be present in the LQN model of any WebML application. To model the application-specific logic, inside the App Server task we define an entry for each navigation link, then we create an activity for each content unit and operation, and finally we link all the activities according to page computation sequences. Each activity in the App Server task is connected to a corresponding DB server entry by means of synchronous calls.

At the DB server task, entries are mapped to SQL statements according to their functionality: for example, units with simple (attribute) selectors are mapped to plain selects, while units with selectors based on relationships are mapped to queries with joins.

Figure 4 depicts a simple WebML model fragment (on the left side) and the LQN sub-model obtained by applying the model transformation (on the right).

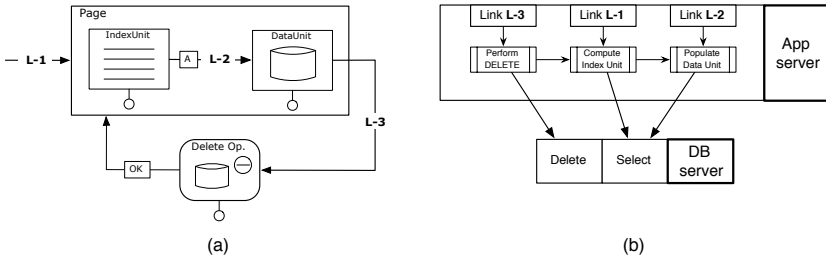


Fig. 4. A simple WebML hyperlink model (a) and the corresponding LQN model (b)

Considering this example, if a user navigates to the page using the non-contextual link L-1, the RTX framework has to compute the index unit and then (L-2 is an automatic link) the data unit, resulting in two select operations on the DB. The LQN model reflects this behaviour using the entry named **Link L-1** and the sequence of activities: **Compute Index Unit** and then **Compute Data Unit**. If the user navigates to the page using link L-2 instead, the computation sequence changes and only one select on the DB is required. Again the LQN reflects this situation using a different entry (**Link L-2**) connected only to the last activity. Finally, if the user issues the delete operation (link **Link L-3**) the runtime framework has to perform the delete operation first, and then it must recompute the whole page (as in the first case). The LQN models this using an additional entry triggering the sequence of all three activities. If we were to know with sufficient detail the typical user navigation pattern, we could predict with good accuracy the system load and the average response times for each page for a varying number of concurrent clients. Moreover, taking full advantage of the wealth of knowledge an application model contains, one could use additional information to estimate the computation time of each unit in a page just considering its type, multiplicity, and selectors and how they impact on query execution, state object population, and rendering time (e.g., selectors on relationship imply joins, hierarchical index units have a query with join per each level).

Clearly, the more complex the LQN, the more parameters are needed and the longer it takes to be computed. A possible measure to reduce LQN complexities is to sacrifice some detail by aggregating all the activities in a single entry per page, or by aggregating all units per type, providing a reference execution time per each type at the DB Server task.

4.2 User Behavior and Resource Usage

Two important parts of the LQN cannot be directly derived from the application models as they are: the user navigation model and an estimate of the resource usages (e.g., the time spent for computing a specific unit on a specific platform).

Concerning the former case the model can provide us, through transformations, with a representation of all possible navigation steps (e.g., a Markov chain with probability equally distributed among all navigable links in a page). But real applications have specific user roles with their own common navigational behaviour. These behaviours have to be extracted from real application scenarios (usually collected at beta application release) and represented consistently in the LQN using a combination of activities, OR-forks, and probability. An example of such user LQN is shown in the top part of Figure 6 where a user can choose among three different navigation paths.

With respect to resource usage, the LQN model parameters have to be set according to the experience of a system expert or have to be measured with ad-hoc experiments at staging time. In a possible future scenario, WebRatio could provide designers with a set of benchmark resource needs for units and deployments gathered from real applications.

Given user navigation paths and resource usages, the LQN will provide insightful information to an application designer in two different ways. First of all, it can be computed in the background (possibly at each model modification) at design time or during site evolution to provide immediate feedback on the expected response time of all application pages. This will indicate to the developer possible alternative designs, for example by suggesting to break a complex page into multiple navigational steps. On the other hand, at deployment time, the LQN can be used to provide prediction on the optimal tuning of different architectural configuration parameters (e.g., number of DB connections, number of threads in the AS) by manually setting them in the model and performing an exploration in the configuration parameter space.

5 A Proof of Concept Example

As a proof of concept, we will use a product catalogue and content management system of a small furniture manufacturing company, called Acme, to generate LQN models and discuss results. An excerpt of hypertext model is shown in Figure 5. We used a simple XSL stylesheet to transform the application model for the public siteview of Acme into a LQN representation for *lqns* (Layered Queueing Network Solver and Simulator¹).

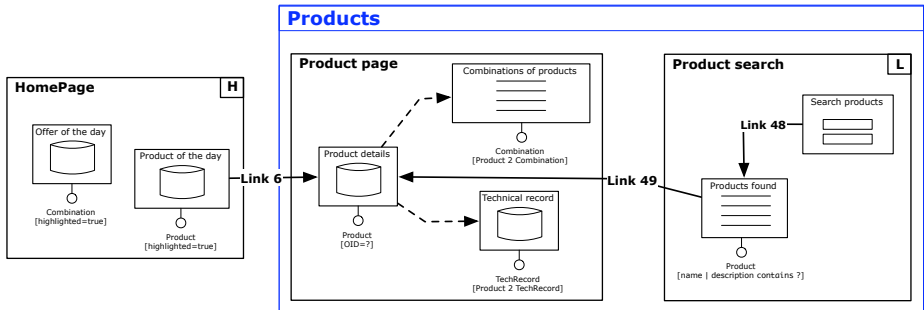


Fig. 5. Excerpt of the hypertext model for the Acme application

Each page computation sequence has been transformed into a sequence of activities at the application server level; we defined a simple non-intensive² navigation sequence for users (directly coded in the XSL), assigned a common resource usage for page and unit rendering (10ms each), and different db resource usages for queries (10ms for selects, 20ms for selects with joins³). Concerning the architecture, we are using a simplified solution with one application server (AS)

¹ <http://www.sce.carleton.ca/rads/lqns/>

² Users have 3 possible navigational paths asking around 5 pages over 20 seconds and repeat the cycle every 2 minutes.

³ The resource usages we chose are realistic but have no significance, our aim is to provide a numerical example.

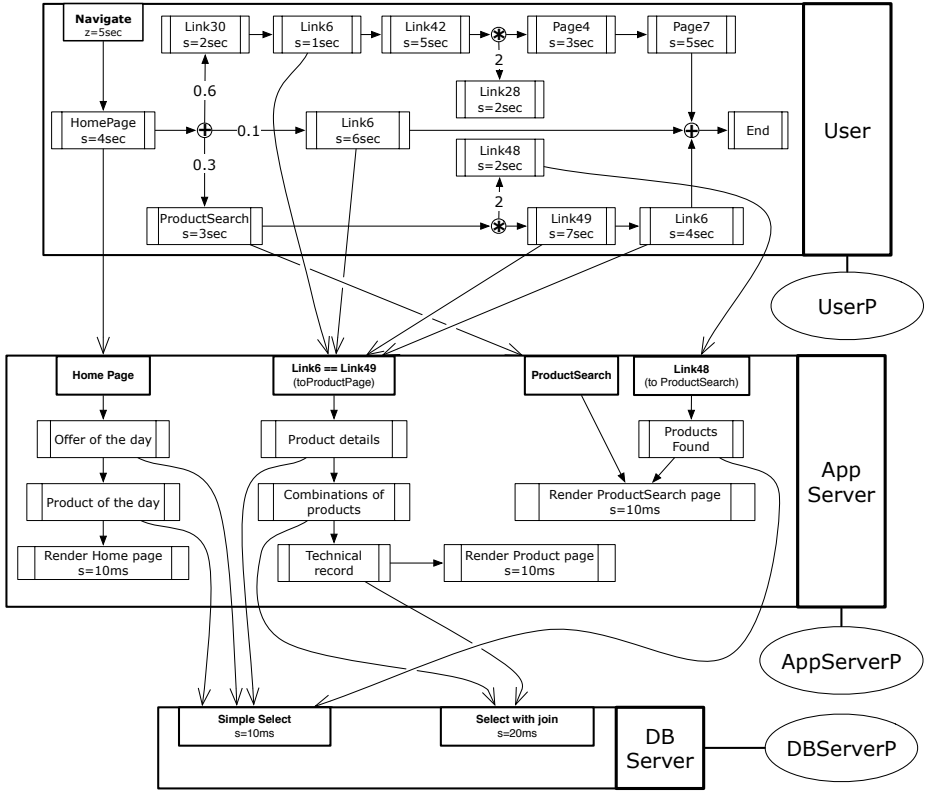


Fig. 6. A partial representation of the LQN for the Acme case study (limited to the Hypertext model in Figure 5)

and one database (DB) running on separate machines. A partial representation of the LQN we used is shown in Figure 6.

Figure 7 shows the plot of the predicted response times for pages *Home* (page1), *Product Details* (page2), *Product Search* (page19) and performing a search (ln48) in the Acme public siteview under varying user load with a fixed system configuration of 20 threads both on the AS and the DB. In the same graph, we also plotted the predicted CPU utilizations for both AS and DB. The reader should notice how as soon as the AS CPU is close to saturation response times raise abruptly. The bottleneck for this load and configuration is therefore the AS, while the DB could sustain higher load.

In Figure 8 instead, we use the LQN to explore one of the AS configuration parameters, in particular we try to identify the optimal number of threads to use under different numbers of concurrent clients. Our interpretation of the results is that for high loads having more than 10 threads has negative effects, while for low loads the differences in response times are minimal.

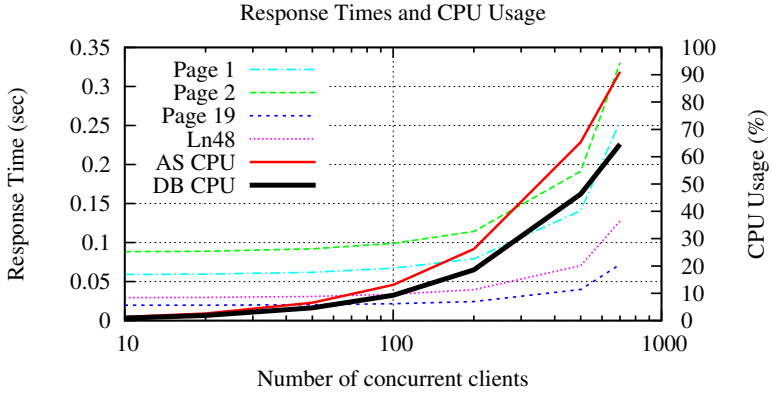


Fig. 7. LQN prediction of response times for two pages and CPU utilization for a synthetic workload

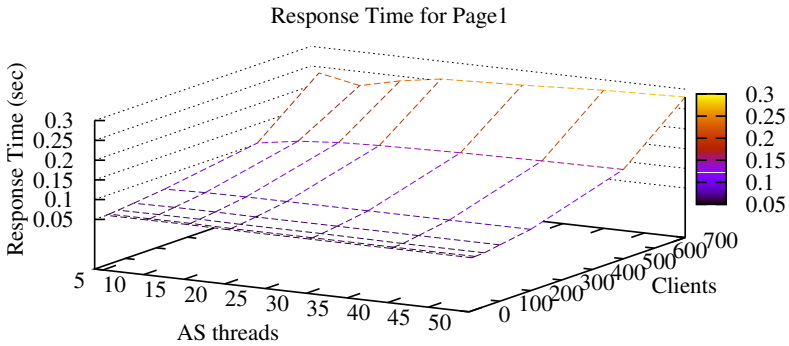


Fig. 8. LQN prediction of response times for Page1 varying concurrent clients and AS threads

6 Related Work

The concept of Model-Driven Performance Engineering was introduced by Fritzsche and Johannes in [6] and is defined as a process combining MDE and performance engineering: structural and behavioral models can be annotated with performance information and performance analysis can be applied at the different stages of refinement of the MDE process. In this context, several works have been proposed, typically considering UML design models that are transformed into a LQN or queueing network performance model, like, for example, [74] where XLST and ATL rules are used for the transformation, respectively. Compared to these works our approach is quite similar, since we translate pieces of WebML into LQN sub models and we use LQN for the exploration of parameters.

However, starting from a Web modeling language, we can choose different levels of details/abstraction specifically tailored to Web applications.

LQNs are one of the most adopted technique and have been efficiently applied also in other related contexts, like ERP applications [10] heavily adopting multi-threading and multi-core architectures, in the early phases of Software Product Lines [12], in the optimization of the deployment of multiple Web applications on a cloud [8], with replicated components [9], where LQN based approaches scale very well.

Considering the Web Engineering context, the problem of performances has been treated in different works: [3] discusses techniques for balancing load among multiple servers and shows how Web pages can be designed to improve performances; [5] shows a technique to minimize the access cost of a website by adding hotlinks that make the most accessed pages more accessible. In all these cases no formal models are employed and no performance prediction is applied.

In [13] a model-driven capacity planning tool suite, called Revel8or, for J2EE, .Net and Web services platforms is proposed to analyze and predict Web/Web services performance: this tool integrates performance analysis based on UML and Queueing Networks, with automated benchmark generation. Compared to this suite, our proposal goes in to deeper detail in the analysis of models specifically designed for Web applications and, to the best of our knowledge, this is the first time that LQN are obtained from a Web Engineering modeling notation.

7 Conclusions

In this paper we have described a model-driven approach that derives LQN models for the performance prediction of a Web application from a MDWE methodology. A proof of concept applied on WebML/WebRatio is presented. Although much work must be done to validate the applicability of the whole approach in practice, on the basis of this experience we retain that MDWE techniques can be efficaciously employed to automatically obtain accurate representations of the running applications behaviour. As future work we plan to apply the proposed approach to real case studies and to implement a framework for WebML/WebRatio based on LQNs.

Acknowledgments. We wish to thank Antonio Carzaniga for his comments and suggestions. This work is partially supported by the European Community under the IST programme of the 7th FP for RTD - project RESERVOIR contract IST-215605.

References

1. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: A survey. *IEEE Trans. Softw. Eng.* 30(5), 295–310 (2004)
2. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing data-intensive Web applications (2003)

3. Challenger, J., Iyengar, A., Dantzig, P., Dias, D.M., Mills, N.: Engineering highly accessed web sites for performance. In: Murugesan, S., Desphande, Y. (eds.) *Web Engineering*. LNCS, vol. 2016, pp. 247–265. Springer, Heidelberg (2001)
4. Cortellessa, V., Di Gregorio, S., Di Marco, A.: Using atl for transformations in software performance engineering: a step ahead of java-based transformations? In: *WOSP 2008: Proceedings of the 7th international workshop on Software and performance*, pp. 127–132 (2008)
5. Czerwicz, J., Kranakis, E., Krizanc, D., Pelc, A., Martin, M.V.: Enhancing hyperlink structure for improving web performance. *J. Web. Eng.* 1(2), 93–127 (2003)
6. Fritzsche, M., Johannes, J.: Putting performance engineering into model-driven engineering: Model-driven performance engineering. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) *MODELS 2007*. LNCS, vol. 4735, pp. 164–175. Springer, Heidelberg (2007)
7. Gu, G.P., Petriu, D.C.: Xslt transformation from uml models to lqn performance models. In: *WOSP 2002: Proceedings of the 3rd international workshop on Software and performance*, pp. 227–234 (2002)
8. Li, J.Z., Chinneck, J., Woodside, M., Litoiu, M., Iszlai, G.: Performance model driven QoS guarantees and optimization in clouds. In: *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp. 15–22 (2009)
9. Omari, T., Franks, G., Woodside, M., Pan, A.: Solving layered queueing networks of large client-server systems with symmetric replication. In: *Proceedings of the International Workshop on Software and performance (WOSP 2005)*, pp. 159–166 (2005)
10. Rolia, J.A., Casale, G., Krishnamurthy, D., Dawson, S., Kraft, S.: Predictive modelling of sap erp applications: challenges and solutions. In: *Proceedings of the International ICST Conference on Performance Evaluation Methodologies and Tools*, pp. 1–9 (2009)
11. Rolia, J.A., Sevcik, K.C.: The method of layers. *IEEE Transactions on Software Engineering* 21, 689–700 (1995)
12. Tawhid, R., Petriu, D.C.: Integrating performance analysis in the model driven development of software product lines. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) *MODELS 2008*. LNCS, vol. 5301, pp. 490–504. Springer, Heidelberg (2008)
13. Zhu, L., Liu, Y., Bui, N.B., Gorton, I.: Revel8or: Model driven capacity planning tool suite. In: *ICSE 2007: Proceedings of the 29th international conference on Software Engineering*, pp. 797–800 (2007)

Models and Meta Models for Transactions in Web Applications

Mark Douglas Jacyntho^{1,2,3} and Daniel Schwabe¹

¹ Departamento de Informática, PUC-Rio, Av. Marquês de São Vicente, 225,
Rio de Janeiro, Brazil

{mjacyntho, dschwabe}@inf.puc-rio.br

² Universidade Candido Mendes, Núcleo de Pesquisa e Desenvolvimento,
Rua Anita Pessanha 100, Campos dos Goytacazes, RJ, Brazil

³ Instituto Federal de Educação, Ciência e Tecnologia Fluminense,
Av. Dário Vieira Borges, 235, Bom Jesus do Itabapoana, RJ, Brazil

Abstract. In this paper, we present a DSL to specify business and Web transactions in a systematic way, addressing both informational and behavioral perspectives. Our meta-model is based on the reification of transactions, where transactions are modeled as first-class types, supporting attributes, associations, operations and state machines. Treating transactions as domain type instances facilitates the interplay with other models, such as the navigational model, which is a view of the domain type model.

Keywords: Business Process, Workflow, Business Transaction, Workflow Transaction, Web Transaction, Conceptual Modeling, Process Modeling, Transaction Modeling, Meta-model, Ontology, Semantic Web, DSL, MDE, MDWE, MDD, MDA.

1 Introduction

Web applications are multi-concern systems. Since each concern has its own characteristics and peculiarities, a good modeling strategy is to promote their clear separation. This approach is adopted in the majority of Web development methodologies. In general, for each concern, there is a separate model: domain model, navigation model, interface model, behavior model, and so on.

Web application behaviors have evolved from simple request handling mechanism to highly complex behavior, within which data and transactions must be managed to support intra- and inter-organization business transactions.

Current Web applications are developed to carry out tasks or goals that are part of a well-defined business transaction workflow, in accordance to its rules and constraints, serving different users whose joint work needs to be coordinated. Issues like distribution and concurrency must be considered, and conflicts solved. For example, in an e-commerce Web site two concurrent sessions can be simultaneously checking out an order containing the last unit of the same product, and last session to send the confirmation will not have success in acquiring the product (that was available only a fraction of

time earlier). Another typical situation could be the concurrent reservation of the same seat in a flight at an airline Web site.

Following [7], within a business transaction one or more systems are used and for each system one or more system transactions are triggered. In the specific case of Web applications, these system transactions can properly be called Web transactions. Based on [3] a Web transaction can be defined as “a sequence of activities performed by a user to achieve a specific goal related to a step of a business transaction, through a Web application”.

Furthermore, another very interesting issue in Web applications is how to combine Web transactions and navigation. Strictly speaking, navigation is a behavior just like any other application behavior. The only difference is that its semantics are known ahead of time, as provided by the standard hypertext node-and-link navigation models, instantiated by the Web. Because of this, specialized “navigation models” have been proposed, to simplify the specification of this portion of the application’s behavior. Typically, navigation is implemented as a read-committed nested transaction to support the selection of items to be processed by a subsequent sibling transaction. For example, the Web transaction “Place Order” has two sequential nested transactions: “Items Selection” and “Checkout”. The former consists in navigation over the catalog to select items to buy and the later creates an order with the previous selected items. Like any other transaction, navigation is subject to concurrence interferences - the information can become obsolete. This fusion of paradigms is a complex task that needs to be carefully analyzed during the process of Web application design, at the risk of unwanted behavior and results.

As the transactional concern is clearly present, and thus a specific model for Web transactions is needed. The lack of this model makes transactions to be relegated to a secondary level in the implementation code – e.g., in the controller’s code in MVC architectures. Each designer creates his ad-hoc controllers based only on the notion of user session and requests along this session. In fact, along the user session there are several Web transactions taking place, with well defined properties, parameters and execution flow, nevertheless these issues are not explicitly modeled. Which request starts the transaction? Which commits? Inside the scope of which transaction a request occurs? In addition, the necessary information (objects) to carry out the transaction are commonly stored, indiscriminately, inside the user session, using a dictionary style cache mechanism, where objects are stored using an associated key. Since these objects are the transaction parameters, why not reify the transaction concept and model these parameters as transaction properties? The transaction object would be itself the information cache and this object could be treated as any other domain object (persistable, navigable, updateable, and so forth).

In this paper, we propose a DSL (defined by a meta-model and a notation) to address the explicit modeling of business and Web transactions, reifying the concept of transaction in a first-class meta-model type. Transaction-type instances correspond to transaction occurrences and store all the state of that occurrence. At runtime, transaction instances are activated via one controller, and implement the transaction logic using its instance data. It is important to say that our focus is Web transactions, not information (data) tier transactions (e.g. Databases). We assume the existence of a transactional information tier.

The challenge of this work is to offer a meta-model that is complete, but without restricting the transactional behaviors that the designer may want to model. It should be possible to model from traditional ACID transactions to modern non-ACID transactions (for example, transaction which permits lost updates that could be solved semantically by another model).

In the same manner we have been treating the navigation concern, from now on, with the transaction primitives defined by the meta-model, we can explicitly model the transactional issues that were formerly addressed only as an implementation detail.

This paper is structured as follows: in section 2 we introduce the meta-model of the DSL. Next, in section 3, we present an example and some observations regarding the benefits of using the DSL. Section 4 presents some related work and, finally, section 5 concludes this paper with final remarks and future work.

2 DSL Meta-model

Our meta-model is based on the notion of reification, by promoting the concept of transaction to a first-class type, responsible to keep the execution state (actual parameters, local variables, execution status, etc), controlling all the execution steps. Reified transaction types can be used like any other domain type, and may also be persisted.

Another significant characteristic of this meta-model is the clear separation, into distinct meta-classes, of the transaction specification ("what") and its possible enactments ("how"). Specification here means a contract given the pre/post-conditions added through domain invariants. Enactment means one or more execution protocols that satisfy the contract, achieving the same post-condition. Each enactment is described by a flow model, where the transaction is further detailed into increasingly finer transactions until the lowest level of abstraction is reached, i.e. operation calls. Note that to reuse one transaction as a nested one inside an enactment's flow of a parent transaction, we need only to know the specification of the invoked transaction ("what"), and any enactment ("how"), instantiated at runtime, will work fine, because, by definition, it satisfies the contract specification.

The meta-model has two orthogonal perspectives. The first is the informational or static view regarding the properties (attributes and associations) of the transaction and the second is the behavioral or flow view, where the focus is on the execution flow of a possible enactment that fulfills the transaction contract.

Along the explanation, the concepts will be clarified using excerpts of a case study about a business transaction dealing with analysis of artifacts (*papers, workshops, and talks*) submitted to a conference edition, encompassing: submission, review and selection of artifacts.

2.1 Informational Perspective

The informational perspective is concerned with the necessary information - parameters and local variables used to carry out the transaction - and objects manipulated

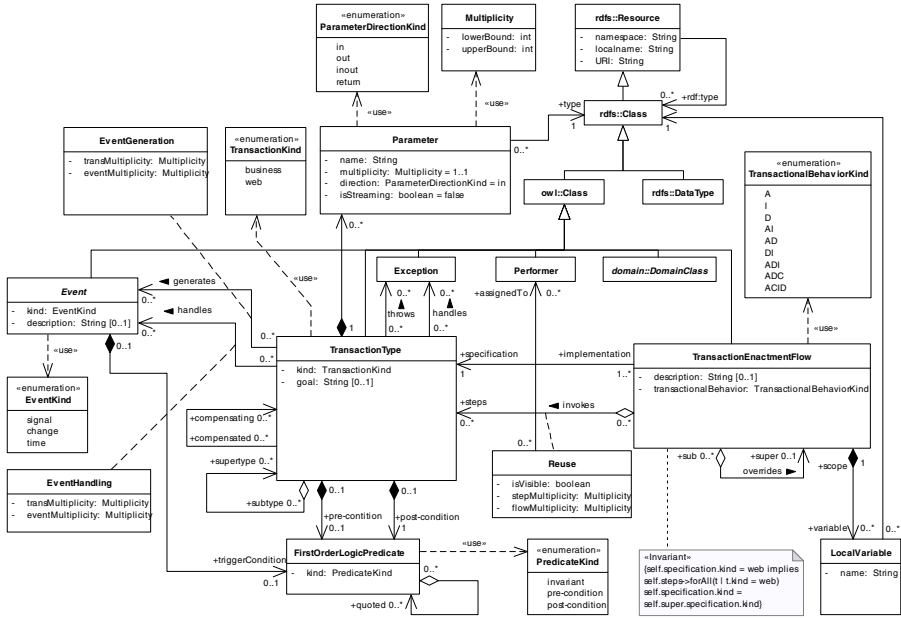


Fig. 1. Meta-model informational perspective

during the course of the transaction. Figure 1 shows the informational portion of the meta-model as a UML-style class diagram.

The core concept of this meta-model is *TransactionType*, which models a transaction specification ("what"). There are two kinds of transactions: *business* and *Web*.

To specify a transaction, we must define its contract, encompassing *Parameters*, *Exceptions*, and *pre/post-conditions*. *Parameter* is used to model the input/output parameters of the transaction. Following the design by contract approach, *post-condition* is a first order logic predicate (Boolean expression) that specifies the effect of the transaction that must be guaranteed by any enactment.

Sometimes the post-condition makes sense only under certain initial conditions, so called *pre-conditions* which are also Boolean predicates. If the pre-condition is not true when a transaction starts, this specification does not apply to it, so the outcome is unspecified, and the post-condition is not guaranteed. Invariants may also be defined that are automatically added (logic operator AND) to the pre/post-conditions of all defined transactions.

Finally, the exceptional outcomes of a transaction can be specified using *Exceptions*. An exception can be considered a special kind of return parameter. For each exception there may be a transaction that plays the role of exception handler.

To illustrate these primitives, the Web transaction used to submit the artifact to a conference edition - "Artifact Submission Transaction" - has three input parameters (*authors*, *first author*, *conference edition*), one output parameter (*new artifact*), one exception (*artifact wrong format exception*). The pre-condition is that the current date

has to be before the edition's deadline submission date and the post-condition is that a new artifact instance has been created, with status *submitted*.

Another important primitive is the notion of *Event*, which models events that the application sends or receives addressed to or originated from external entities (actors). An event can be compared to a trigger that initiates a transaction. For instance, the author decides to withdraw his artifact. An incoming "Withdrawal Event" triggers the Web transaction "Artifact Withdrawal Transaction", suspending any other transaction that may be in progress ("Reviewers Assignment", "Obtaining Artifact To Review", "Artifact Review Submission").

The transaction type may have subtypes. All specifications of the supertype transaction are inherited by the subtype transactions. This is a type or specification inheritance and, thus, overriding is not permitted, only extensions. Based on the design by contract rules, a subtype post-condition can only extend the supertype post-condition, not redefine it. In addition, the subtype pre-condition must not extend the supertype pre-condition. The subtype pre-condition can be weaker but not stronger than the supertype pre-condition. In other words, like any other kind of subtype, a transaction subtype must satisfy the *Liskov Substitution Principle* [6], so that a subtype instance can properly be used in any context where we reason just using the supertype specification, without knowing anything about possible subtypes. In our case study, for each subtype of *Artifact* (*Paper*, *Workshop* and *Talk*), there is a corresponding Web transaction for submission which is a subtype of the "Artifact Submission Transaction".

Finally, a transaction may have one or more alternative compensating transactions. A compensating transaction is a way to "reverse" the effects of a terminated transaction in case of a subsequent abort. It is different from rollback because the effect is not rolled back, but replaced by another effect that compensates the first.

Once the transaction is specified, the next step is to model at least one enactment (*TransactionEnactmentFlow*) of "how" to achieve its post-condition, that is, one possible realization or implementation of the transaction ("what"). Statically speaking, an enactment defines the ACID properties and its constituent nested transactions (invoked/reused) that define the steps of the referred transaction. It is perfectly possible to define ACID and non-ACID enactments. Each enactment defines what nested transactions are (re)used to carry out the specified parent transaction and what *Performers* (actors) can execute them. For example, as mentioned before the business transaction of artifact analysis - "Artifact Analysis Transaction" - encompasses submission, review and selection of artifacts. Therefore, one possible enactment of this business transaction - "Artifact Analysis Enactment" - has the following nested Web transactions as steps: "Artifact Submission", "Reviewers Assignment", "Obtaining Artifact to Review", "Artifact Review Submission", "Artifact Decision", "Artifact Final Version Submission", "Artifact Withdrawal", and "Wrong Format Exception Handling". These last two ones are, respectively, the event handler associated with the "Withdrawal Event", if the author decides to withdraw his artifact, and the exception handler associated with the "Artifact Wrong Format Exception", exception thrown by the "Artifact Submission" Web transaction, if the submitted artifact does not adhere the conference required format.

An important point is the constraint that Web transactions can only be composed of other Web transactions. It does not make sense to put a business transaction inside a Web system transaction. Another important point is to indicate whether the outcome of the terminated nested transaction is visible or not beyond the parent transaction boundary.

Finally, an enactment can be defined via inheritance of another enactment. This is an "implementation" inheritance and, therefore, extensions and overriding, both are permitted. Nevertheless, there is a restriction that an enactment can only inherit from another enactment of the same transaction kind (business or Web). Again, in our example, for each subtype of *Artifact* there is a corresponding enactment which inherits from "Artifact Analysis Enactment", because of the extra constraints, like for example, the corresponding Web transaction "Artifact Submission" subtype: for *Paper* must be "Paper Submission", for *Workshop* must be "Workshop Submission" and for *Talk* must be "Talk Submission".

To specify the dynamic perspective, *TransactionEnactmentFlow* allows modeling the flow logic of the transaction's steps, describing the sequence in which the nested transactions take place. This is the subject of the next section, where we explain the behavioral perspective of the meta-model.

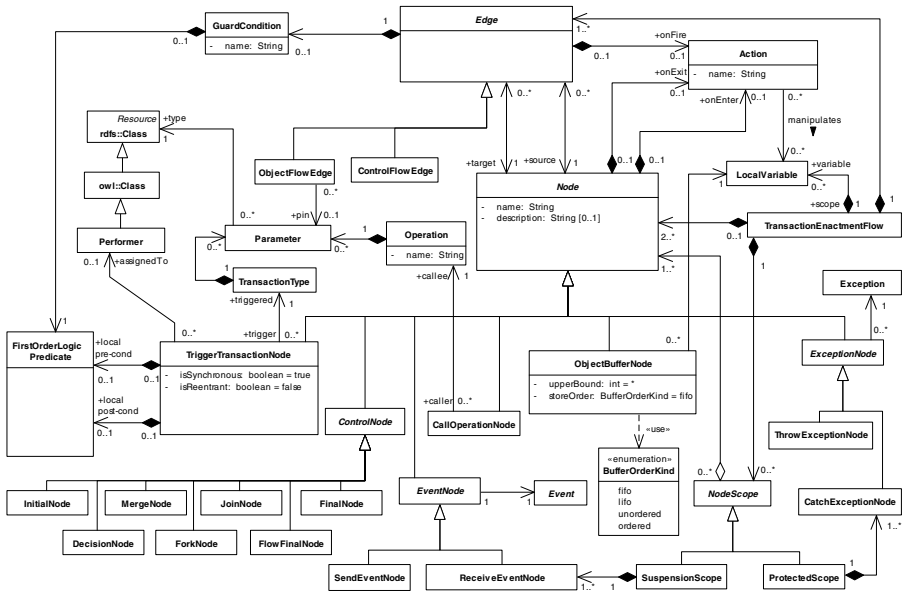


Fig. 2. Meta-model behavioral perspective

2.2 Behavioral Perspective

This view defines the possible execution flows between steps (nested transactions or operations) of a transaction enactment. This part of the meta-model is a specialized version of the UML 2.0 activity meta-model, which is based on Petri nets [4].

Basically, the *TransactionEnactmentFlow* primitive contains nodes connected by edges forming a complete flow graph, and control and object (data) tokens flow along the edges and are operated on by nodes, routed to other nodes, or stored temporarily. Control token is a Boolean data that indicates that that target activity node cannot start until the source activity node terminates. Figure 2 shows the behavioral part of the meta-model. There are meta-concepts in common between the informational and behavioral diagrams of the meta-model. To simplify the diagram, some relationships already shown in Figure 1 were omitted in Figure 2.

The informational part of the *TransactionEnactmentFlow* models only the static composition of the enactment; its constituent transactions might happen in parallel, in sequence, there might be alternative paths, repetitions, etc. The role of the behavioral meta-model is to complement the definition of a transaction enactment, establishing the execution flow.

There are six kinds of node: *TriggerTransactionNode* - represents a transaction trigger. There may also be a *Performer* associated who is the actor (role) that executes this transaction; *CallOperationNode* - represents a call to an operation; *Control Nodes* - route control and objects (data) through the graph; *ObjectBufferNode* - holds data values (objects). Represents the usage of a *LocalVariable* as a buffer; *Exception Nodes* - throw or catch an exception; *Event Nodes* - send or accept an event.

In addition to node and edges, there is the notion of *Scope*, which is a set of nodes grouped for some purpose. There are two kinds of *Scope*. *SuspensionScope*: this scope gathers *ReceiveEventNodes* so that when the event happens any activity inside the scope is suspended. *ProtectedScope*: this scope is similar to the try/catch mechanism in programming languages. The nodes inside the scope are protected against exceptions thrown within the scope. The scope has one or more *CatchExceptionNodes* associated and the scope is protected from each corresponding *Exception*.

Given this meta-model, it should be clear that it is possible to define an enactment which implements a standard hypermedia navigation engine, which keeps track of the navigation state of the application at any point. From this, it straightforward to see that navigation becomes just another transaction in the application.

3 An Example

As we know, given an application domain, there exist common features present in the vast majority of the models that may be used to build a standard reusable framework model. Therefore, in general, the modelers do not instantiate a meta-model from scratch, but create their models by extending pre-existing standard models (abstract meta-model instances).

To assist the meta-model instantiation, we provide a standard base instance of our meta-model to be extended by a particular model. This standard abstract model, not shown for the sake of space, defines the *Transaction* abstract class from which all transaction classes can be derived. In addition there is the class *Flow (Thread)* that represents the execution flows of the transaction. One transaction can have one or more executions flows or threads. Two or more flows are created when there is a fork control node in the behavioral model. Each *Flow* maintains the steps (Transactions

instances) executed in its scope, including its current step. One Transaction can have many flows in execution, but only one flow is effectively running (progressing), per user session.

The classes have some standard properties and an execution status attribute. The Flow class has five lifecycle statuses: *ready*, *running*, *blocked*, *finished*, and *interrupted*. The Transaction class has duration attribute and the following statuses: *definingActualParameters*, *executing*, *committed*, *aborting*, *aborted*, *compensating*, *compensated*, and *terminatedByException*.

We next show our case study for a business transaction of analysis of artifacts submitted to a conference as an extension of the standard model. In this example, this business transaction is called *ArtifactAnalysis* and one possible enactment has seven Web transactions: *ArtifactSubmission* - submission of the artifact executed by one of the authors; *ReviewersAssignment* - assignment of reviewers to assess the artifact, carried out by the pc-chair; *AcquiringArtifactToReview* - each reviewer obtains the artifact to evaluate; *ArtifactReviewSubmission* - each reviewer submits the respective evaluation; *ArtifactDecision* - the pc-chair accepts or rejects the artifact; *ArtifactFinalVersionSubmission* - if the artifact was accepted, the author submits the final version; *ArtifactWithdrawal* - the author withdraws the artifact.

For the sake of space, the domain model will not be shown. The main domain concept is *Artifact* and its subtypes (*Paper*, *Workshop*, and *Talk*). Among other properties, *Artifact* has *Authors* and status (*submitted*, *waitingReviewers*, *inReview*, *reviewed*, *acceptedWaitingLastVersion*, *withoutLastVersion*, *accepted*, *rejected*, and *withdrawn*). Each *Artifact* is associated (submitted) with one conference *Edition*. Each *Artifact* undergoes three *Reviews*, each made by one *Reviewer*. Finally, the *Artifact* is accepted or rejected by the *PC-Chair* of the *Edition*. If accepted, a final version must be submitted, otherwise the *Artifact* is not considered accepted. The first step is to model the informational view of the transactions. Figure 3¹ shows informational view of the business transaction *ArtifactAnalysis* and of the Web transaction *ArtifactSubmission* (suppressing its pre/post-conditions, for lack of space) and its subtypes. To simplify, we show only one possible enactment (*ArtifactAnalysisEnactment*) of the *ArtifactAnalysis* specification and we do not show the ACID enactments of the *ArtifactSubmission* Web transaction and its subtypes. The ACID behavior of the *ArtifactAnalysisEnactment* has only "D" (Durability) and, remember that this enactment must honor the post-conditions "note" associated with the *ArtifactAnalysis* transaction. The dashed line subtype symbol between *ArtifactAnalysisEnactment* and *ArtifactAnalysis* denotes realization (one possible implementation) of the transaction. This is the type classification of the enactment, not an inheritance, that is, all instances of one enactment are instances of the corresponding transaction. Note that *ArtifactAnalysis* and *ArtifactSubmission* are subclasses of the *Transaction* class defined in the standard abstract model.

¹ DSL notation: double oval - *Business Transaction*; simple oval - *Web Transaction*; rounded rectangle - *Enactment*; up/down arrow - *Exception*; rectangle - *Domain Object*; diamond arrow - *Step*; solid line/triangle - *Subtype/Overriding*; dashed line/triangle - *Realization of a specification by one enactment*.

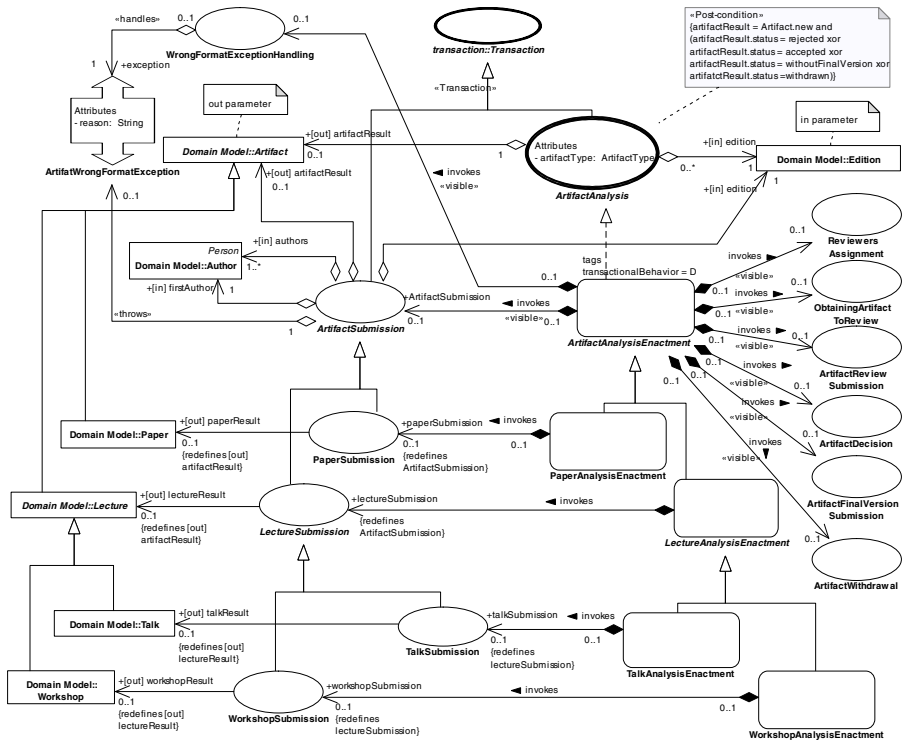


Fig. 3. Artifact Analysis business transaction informational model

The *ArtifactAnalysisEnactment* is composed (composition diamond symbol) by all Web transactions mentioned above (whose enactments are not shown). An interesting aspect is the overriding of *Artifact AnalysisEnactment* for each subtype of *Artifact*. This overriding is necessary because the *ArtifactSubmission* Web transaction is different for each subtype and, thus, the relationship with *ArtifactSubmission* must be redefined.

Similarly, the relationship with *Artifact* via the output parameter *artifactResult* is also refined for each subtype of *ArtifactSubmission*. Other remarkable point is the abstract refinement from *ArtifactAnalysis* to *ArtifactSubmission*. At the *ArtifactAnalysis* level of abstraction there are only two parameters (*artifactResult* and *edition*) but at the level of *ArtifactSubmission* (and its subtypes) there are two extra parameters (*authors* and *firstAutor*). If the artifact is in the wrong format, the *ArtifactSubmission* Web transaction throws the *ArtifatWrongFormatException* that is handled by the Web transaction *WrongFormatExceptionHandling*, where the artifact is rejected.

Assuming that this conference works with multiple pc-chairs, we have to avoid concurrence interferences when one pc-chair allocates reviewers to an artifact (*ReviewersAssignment* Web transaction). Specifying that this Web transaction enactment ACID behavior has at least "I" which means that this Web transaction must be isolated elegantly solves this issue. This implies some lock implementation mechanism

in the running application, optimistic or pessimistic. Theoretically, "I" means *serializable*, but, in practice, this can be relaxed to augment concurrency and one may simply specify the level of isolation: *read uncommitted*, *read committed*, *repeatable read* and *serializable*. In this particular Web transaction, for example, we may use *repeatable read*, indicating that this Web transaction will have success only if no other concurrency session updates the artifact.

To complete the modeling, Figure 4² shows the flow model for the *ArtifactAnalysis* business transaction that defines execution flow of the transaction steps. Note that there are two scopes: one protected scope and one suspension scope. The first catches the *ArtifactWrongFormatException* thrown by the *ArtifactSubmission* step and sends to the *WrongFormatExceptionHandling* step. The other scope monitors the arrival of the external *WithdrawalEvent* that is generated by the user (author) when he wishes to withdraw the artifact. The rest is the traditional flow modeling. It is important to say that for each Artifact subtype (Paper, Workshop, and Talk) the corresponding *ArtifactSubmission* Web transaction subtype substitutes the *ArtifactSubmission* step in the flow model.

Finally, to illustrate the interplay between the transactional and navigational concerns, imagine that the designer wishes to permit the reviewer, when he is executing the *ArtifactReviewSubmission* Web transaction, to navigate to other artifacts that he has reviewed earlier. To achieve this, we have three independent models: the transaction model, where the *ArtifactReviewSubmission* Web transaction is specified, the navigational model, where the Artifact navigational node is, and an interface model, where there is an element that, when activated, sends two events, one to start *ArtifactReviewSubmission* Web transaction and another to change the navigation state to the Artifact navigational node. The resulting interface is the composition of the two executing "transaction" states: *ArtifactReviewSubmission* and navigation, which can be interpreted as a special Web transaction that never commits. In this way, the reviewer can navigate via the Artifact navigational node, suspending the *ArtifactReviewSubmission* Web transaction, which can be resumed later in the same form it has started.

At the end of this example, it becomes clear that following the roadmap suggested by the meta-model, the designer defines, explicitly, "all" properties and flow of each transactions, being forced to think about several important nuances that formerly were forgotten, and were only revealed in test (or production) phase, in the form of error (or defect).

In terms of software architecture, we can conjecture one transaction execution controller component - "Transaction Engine" - whose logic would already be defined by the models: "execution flow" - flow model pre-loaded; "(http) parameter cache" - reified transaction objects' attributes, and "concurrency control and lock management" - ACID properties of the transactions.

² DSL notation: rounded rectangle - *TriggerTransactionNode* or *CallOperationNode*; dashed rounded rectangle - *Scope*; arrow - *Edge*; diamond - *Decision/Merge Nodes*; bar - *Fork/Join Nodes*; solid circle - *InitialNode*; X circle - *FlowFinalNode*; solid/white circles - *FinalNode*; up arrow rectangle - *ThrowNode*; down arrow rectangle - *CatchNode*; pentagon - *SendEventNode*; chevron rectangle - *ReceiveEventNode*; little rectangle - *ObjectBufferNode*.

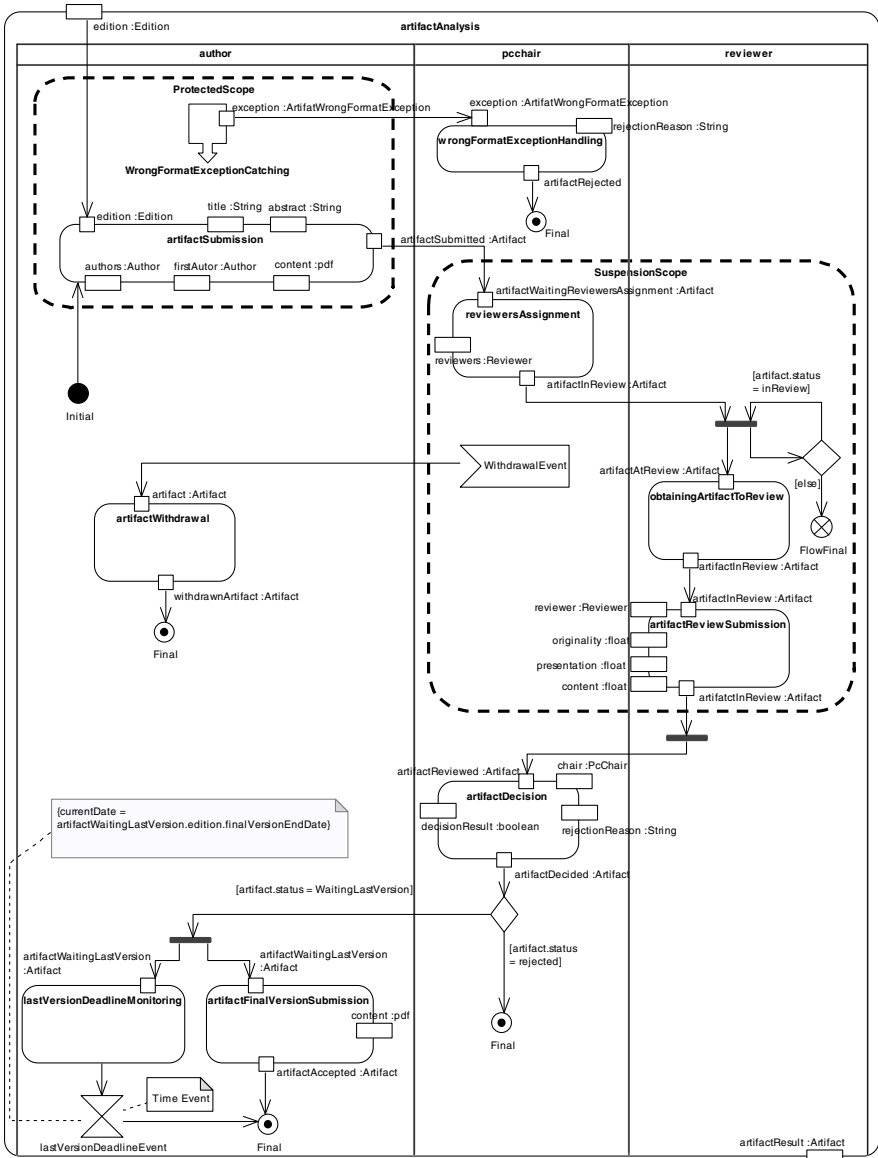


Fig. 4. Artifact Analysis business transaction behavioral model

4 Related Work

A number of well-known Web design methods have been extended to address transactions: Brambilla et al [2], OO-H and UWE [5], UWA [1], not detailed here for lack of space. None treat navigation as a special Web transaction, mixing up the navigational

and transactional concerns, with no clear separation among the specification and its various possible enactments. None of these proposals include all of the features present in our meta-model.

There are some transaction proposals for Web services (e.g. BPEL [8]) but these are for specifying Web services orchestration, based on Web services standards. For us, Web services are just one implementation choice of domain types operations.

5 Concluding Remarks

In this paper we have presented a reification approach to model business and Web transaction, covering the informational and behavioral perspectives of transactions. We have proposed a meta-model and graphical notation, defining a complete DSL to model transactions. We intend in future work to improve this DSL via many prototypes and to create other important Web application DSLs (e.g security DSL), which must be combined, without mixing the concerns involved.

Acknowledgement: This research was partially funded by CNPq, process number 142192/2007-4 and 302.352/85.6.

References

1. Baresi, L.: Ubiquitous Web applications. In: The eBusiness and eWork Conference (e2002), Prague, Czech Republic (2002)
2. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 15, 360–409 (2006)
3. Distanto, D., Tilley, S.: Conceptual Modeling of Web Application Transactions: Towards a Revised and Extended Version of the UWA Transaction Design Model. In: 11th International Multi-Media Modeling Conference (MMM 2005). IEEE Computer Society Press, Los Alamitos (2005)
4. Hee, K.V., Aalst, W.V.D.: *Workflow Management: Models, Methods, and Systems*. and Systems. MIT Press, Methods (2002)
5. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Modeling Web Business Processes with OO-H and UWE. In: 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003), Oviedo, Spain (2003)
6. Liskov, B., Guttag, J.: *Program Development in Java: abstraction, specification, and object-oriented design*, 6th printing. Addison-Wesley, Reading (2004)
7. Papazoglou, M.P.: Web Services and Business Transactions. *World Wide Web: Internet and Web Information Systems* 6(1), 49–91 (2003)
8. Web Services Business Process Execution Language,
<http://docs.oasisopen.org/wsbpel/2.0/wsbpel-v2.0.pdf>

Using Actions Charts for Reactive Web Application Modeling

Nina Geiger, Tobias George, Marcel Hahn, Ruben Jubeh, and Albert Zündorf

University of Kassel, Software Engineering ,
Department of Computer Science and Electrical Engineering,
Wilhelmshöher Allee 73,
34121 Kassel, Germany
{nina,tge,hahn,ruben,zuendorf}@cs.uni-kassel.de
<http://seblog.cs.uni-kassel.de/>

Abstract. Building a rich internet application (RIA) requires the programming of various callbacks and listeners. AJAX like server requests require callback handler objects that react to the asynchronous server response. Active Graphical User Interface (GUI) elements like buttons or menu entries require action handlers. Using a timer queue requires appropriate event handlers, too. Programming all these handlers is tedious and error prone. Subsequent steps of e.g. initialization or of a protocol of server requests are scattered over multiple separated blocks of code. The control flow between these blocks is hard to retrieve. Some common variables have to be introduced in order to pass the application state between the different handler blocks. To overcome these problems, we propose to use an extension of UML statecharts, called *Action Charts*, dedicated to the modeling of callbacks and listeners. All kinds of handlers are modeled in a common uniform statechart notation. States become actions or handlers. Transitions represent the flow of execution. Variables are shared between actions providing a simple mechanism for passing the application state from one handler to the next. From such Action Charts we generate sourcecode that is compliant with the Google Web Toolkit¹ (GWT). The generated code is pure Java code that facilitates validation and debugging of the modeled behavior. It can be translated to JavaScript and run inside the web browser using the GWT crosscompiler. The Action Charts and code generation are implemented as part of the open source CASE tool Fujaba.

1 Introduction

Modern web applications shift more and more application logic and even the data model to the client. This allows more interactive web applications (following the AJAX pattern). Furthermore, the traditional page-by-page application flow dissolves and is replaced by fully interactive widget based user interfaces with windows, buttons, lists, trees etc. This shift is possible due to several reasons: the runtime environment, the browser, has a powerful, fully programmable

¹ <http://code.google.com/webtoolkit/>

JavaScript engine, and technologies like DOM and CSS allow the visible content to be manipulated in a very flexible manner during runtime.

Furthermore, development tools for web applications improve accordingly. Our work is based on the Google Web Toolkit (GWT), which allows the application developer to implement the whole application, that is client and server code, in Java. Browser-understandable artifacts like HTML, CSS and ECMA-/JavaScript are generated on application deployment, the developer is faced with just a simple implementation language. The drawback of shifting the application more to the client is that we end up with a distributed application consisting of client and server components: our components have to communicate over an unreliable network connection with arbitrary latency. Dropping the server components completely is not possible, as the browser doesn't provide reliable persistent storage or even should not access or store data. If you take for example web shops, the provider wants to store order details and user information on server side for being able to finish the ordering process. Another case could be online homework systems, that can be used by schools and universities to administrate lectures. Here we have many privacy issues to solve. Only those parts of the application data model the client is allowed to change can be shifted to the client, restricting the access to the other parts. A student, for example, is only allowed to view his own grades, but is not allowed to view the ones of other students attending the same course. The access restrictions also have to ensure, that the student isn't able to change his grades.

The programming environments for web applications usually support client-server-communication following an asynchronous communication pattern, like GWT-RPC services. Because the underlying HTTP connection might fail and the server response time for a request is unknown, the client continues to work after issuing a server request, and will be informed by its local JavaScript Engine when the server response was received. That is called the asynchronous callback pattern. Technically, the client side of web applications is single-threaded, so this approach is necessary to have an always responding interactive application. But the asynchronous callback pattern is cumbersome for the application developer. When issuing a server request one has to pass a callback object and on server response the control flow continues in the corresponding callback object. Furthermore, when executing a sequence of server requests, the callback of the first request has to issue second request passing a callback that has to issue the third request and so on. Thus the code for this logically related sequence of steps is scattered over separated code blocks, in the case of Java even over separated classes. The control flow between these code blocks / classes is hidden in some callback parameters. In addition, these separated code blocks / classes need to provide extra means to enable the passing of a common application state between them.

A similar problem arises when developing the user interface: User input events are usually delivered through the observer pattern. So again, there is no sequential control flow which can be observed by looking at the code, but the event handling and application logic is scattered over or at least invoked by listener objects.

The approach presented in this paper shows Model Driven Engineering of Rich Internet Applications (RIAs) integrated into the CASE-Tool Fujaba (see: section 2). Using this Tool-Suite, which is build upon the Eclipse Platform, one is able to model the structure and behavior of an application via class diagrams and story diagrams. These diagrams are translated to Java code, afterwards. To enable the modeling of web applications, we adapted the standard Fujaba Tool and its diagrams. We are using statecharts with a special transition semantic to transparently map different kinds of events occurring in web applications. Because the execution semantics differs from statecharts, we call them *Action Charts*. This paper also includes an early proposal to bind the application's data model to user interface components (following the MVC pattern) with the same approach: data model changes are replicated to the client side and are able to update the corresponding UI component. This change mechanism is modeled with Action Charts based on adapted statecharts, too. Using adapted code generation mechanisms, we are able to generate completely functional web application code out of the Fujaba Tool. Formally, our Action Charts rely on the usual metamodel of uml statecharts. We use a stereotype `<<ActionContainer>>` to flag the different semantics and to trigger the special codegeneration.

1.1 Running Example

We have modeled a small application covering all the handlers and Action Chart events, as well as the persistent data storage and multi-user capabilities, called the *ToDoListApplication*. Figure 1 shows the user interface of the application. Multiple users can share a *ToDoList*, add new items, prioritize them and mark them as done. The GUI consists of several widgets: A tree with an input field and a Button per item. The Button is associated with a `ClickHandler`, the label is updated via a change event on the underlying `Entry` object residing in the shared data model. The application state is stored persistently on server side and replicated to all connected clients. This way, a user can close the application without losing data. One example Action Chart implementing the application behavior can be seen in Figure 3, the corresponding application model can be seen in Figure 2.

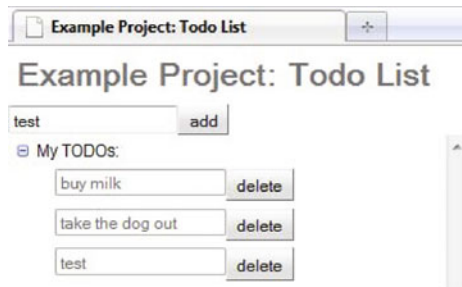


Fig. 1. Todo List User Interface

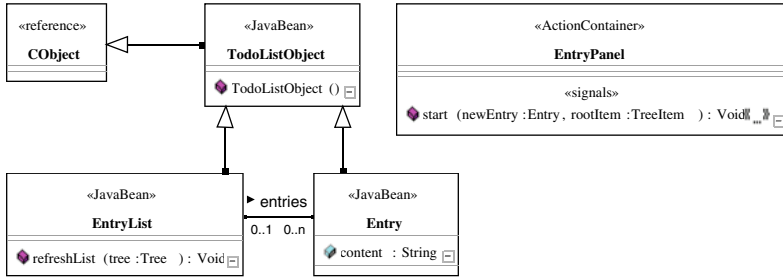


Fig. 2. Example Application: TodoList Class Diagram excerpt

This paper is structured as follows: First, in section 2 we will introduce the Fujaba Tool-Suite on which our approach is built. We will show the modeling and code generation capabilities of this tool and introduce our transformation chain. In section 3 we introduce our modeling approach using a running example. Section 4 describes in more detail the technical issues we have implemented in our Action Charts. Section 5 compares our approach to work that has already been carried out in the area of Web Engineering for Rich Internet Applications. Section 6 finally concludes and focuses on work we have planned for the future.

2 Modeling with Fujaba

The approach presented in this paper is implemented to extend and modify the existing statechart modeling capabilities of the Fujaba Tool Suite 2. Fujaba is an open source CASE Tool, which offers software engineers to develop their applications completely model driven. It enables the design of an application using UML class diagrams, but also provides methods to create the application logic on model level, using story diagrams as visual programming language 4.

After modeling the application using the diagrams introduced above, the CodeGen2 Fujaba plugin 5 generates Java source code out of it. The code generation mechanism used in Fujaba is template based, which makes it easily extendable, this is done for our approach of web development, for example. Another part of the Fujaba Tool which is extended for the use in web applications is the persistency and versioning library CoObRA 13. This library works directly on the runtime object graph and saves it persistently as a stream of change events. We adapt this mechanism to provide persistency for our web clients, see below.

2.1 Story Driven Modeling

The most important part of the Fujaba notation are graph rewrite rules that allow to query and modify the application's object graph on a high level of

² <http://www.fujaba.de>

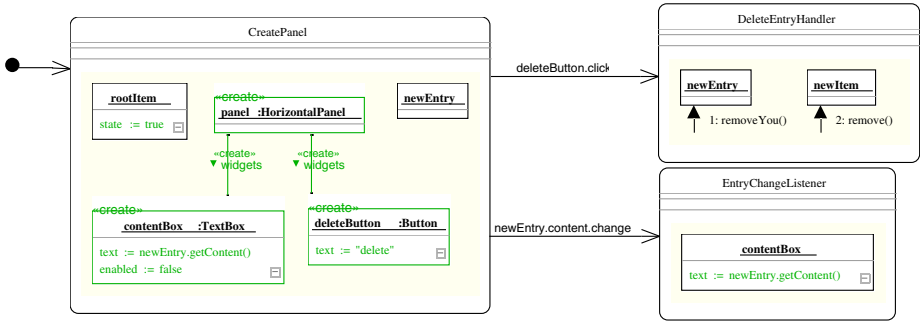


Fig. 3. Example Action Chart Diagram

abstraction. Figure 3 shows a cutout of an Action Chart used to initialize the applications user interface on the client side.

Fujaba allows to embed graph rewrite rules into statechart states resp. actions of an Action Chart. These graph rewrite rules carry out its operations. In Fujaba, a graph rewrite rule consists of an object graph that is used as a query pattern to be matched in the current runtime object graph. For example, the action `CreatePanel` of Figure 3 has a query pattern of three nodes `panel`, `contentBox`, and `deleteButton`. Our graph patterns distinguish bound objects and unbound objects. Bound objects are already matched to runtime objects and need not to be searched any more. In our notation, bound objects show only their name and omit their type. At execution time, unbound objects are matched onto runtime objects such that the overall match conforms to the search pattern graph. In addition to querying the runtime graph with a pattern graph, a graph rewrite rule also allows to model modifications of the matched elements. In our example, the `text` attribute of the `contentBox` object is changed to a new value by an assign expression. Graph elements may also be created or deleted using `<<create>>` and `<<destroy>>` stereotypes. Finally, a graph rewrite rule may contain collaboration messages allowing to do computations and to call methods on the matched objects.

Graph rewrite rules are an excellent means to model an applications behavior in terms of operations on its underlying object graph. The graph patterns allow to express complex graph queries that are (with some exercise) easy to read and to understand. Thereby, the programs are easier to extend and to maintain.

2.2 Model Transformation Chain and Application Architecture

As mentioned in the introduction, we try to shift as much as possible components of our application to the client side. Ideally, just the persistency layer remains on the server side. GWT supports that approach very well, because all code capable of running on the client also runs on the server, just the client-server communication (GWT RPC services) require a certain implementation convention. Because GWT already creates client artifacts (HTML, CSS, JavaScript)

out of Java source code, we just had to adapt our code generator to generate GWT compliant code and invoke GWT's Java-to-JavaScript compiler.

At runtime, the model is instantiated on both sides of the application and is synchronized, even between multiple clients and the server, using the Web-CoObRA features introduced in [1]. This holds for the application data model which is generated out of the class diagram and its associated story diagrams. All the parts that are tightly related to the Graphical User Interface (GUI) part of the application are only run on the client side. This is the case especially for the Action Charts introduced in this paper, as these refer and instantiate GUI widgets, which are only available on the client side.

3 Modeling Approach and Application Architecture

As discussed, our typical web applications deploy model objects at runtime in the web client. GUI events are handled by listeners that query and modify the runtime model within the client. In addition, server requests may be issued. We use a MVC pattern to update the GUI in case of model changes. For persistency and multi user support, the runtime model is replicated on a server. These features can all be modeled using our Action Charts. Each action of such an Action Chart may contain a graph rewrite rule, which can call another Action Chart via collaboration statements. Furthermore, objects created or assigned in an action are action-chart-global, that means they can be referred or accessed in any successor action. Because each action is capable of receiving events at any time, our proposal doesn't follow a strict sequential execution. Reactive event-triggered actions resp. states can be seen as a short-hand notation for a parallel and-state with a waiting state preceding the actual reaction state. An Action Chart that has multiple reactive states (and thus many hidden wait states), actually treats all these event handlers as additional and-substates. However, all these parallel substates listen to disjoint events: each parallel substate listens to its own GUI or change or timer or RPC event. Because the client side JavaScript execution model is single-threaded, only one event is handled at a time and thus, although we deploy so many parallel substates, we don't observe any concurrency problems between these parallel substates. The following four sections discuss the distinct reactive features of our modeling approach:

3.1 RPC Services

Remote Procedure Calls (RPCs) are used in GWT applications to call methods on the server side of the application. Because of the non-blocking behavior of AJAX like applications, a callback object has to be passed when making the call. This callback object is used to inform the client side about the result of the call. The corresponding Interface of the GWT RPC framework distinguishes between a successful call and a call failure. RPC calls can be modeled in Fujaba graph rewrite patterns as collaboration statements.

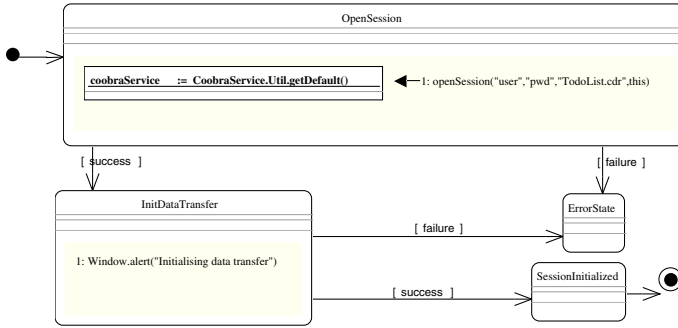


Fig. 4. RPC Calls with Action Charts (excerpt)

The collaboration statement number 1 in action `OpenSession` shown in figure 4, calls the `openSession` method of the `CoobraService`. The last parameter of this method call indicates the callback object. As discussed in the realization section 4, at runtime each action of our Action Chart is turned into an inner class and a specific `FAction` object (cf. section 4). Thus, the `this` variable in the RPC call in state `OpenSession` refers to a runtime action object representing the `OpenSession` state. This makes the calling `OpenSession` action a callback. The action has two outgoing transitions, marked with `success` and `failure`. A successful RPC call will trigger the `success` transition to the next action, the `failure` transition could be used to display an error message, cf. section 4.

3.2 User Interface Events

During the execution of the initial state(s) of the application, one might initialize the user interface by instantiating GUI widget components and by building up the DOM tree. This can be done by adding widgets to the `RootPanel` consecutively. Figure 3 shows the initialization of some GUI widgets in the `CreatePanel` action. The definition of a `ClickHandler` for the instantiated button is shown in action `DeleteEntryHandler`, cf. Figure 3.

Click handlers are created by adding transitions labeled with `<objectname>.click` to the action where the object is instantiated. The target action of this transition now works as Click handler and the designated behavior can be implemented by adding graph rewrite rules to this action. This graph rewrite rules may also use collaboration messages to invoke other methods that may implement more complex behavior. Such methods may be modeled using the usual Fujaba story diagram language, cf. 4. Selection Events and Handlers for UI Components can be modeled the same way. The label of the outgoing transition has to conform `<objectname>.select` and the target action will be transformed to a Selection Handler, then.

3.3 Model Changes via PropertyChange Events

Fujaba's code generation provides JavaBeans conforming setters and getters that also fire property changes. This enables our approach to react to model changes with PropertyChange events and the according handlers. In our Action Charts, we model the reaction to property changes as shown in action `EntryChangeListener` in Figure 3.

Action `OpenSession` of Figure 3 refers the `newEntry` object. This object has a property called `content:String`. To react to changes occurring to this property, we define a transition similar to the ones shown in section 3.2. The label of the transition now has to conform to `<propertyname>.change`. The target action of the transition now can be used to model the handling of the change event. Again, graph rewrite rules and method invocations can be used, here. The example shows how to set the text of the textbox to the value stored in the `content` property.

3.4 Timer Events

Timers are provided by the runtime environment and are currently the only possibility to achieve quasi-parallel execution of application logic. The application on the client side is limited by it's environment to utilize only a single thread. Using a timer, actions can be deferred or executed periodically, so this is an acceptable workaround and our proposal directly supports time-triggered actions. To schedule a timer callback object, we use `after <integer expression>` transitions. When the source action is reached, a GWT Timer is scheduled with the target action as handler object. The target action may again contain a graph rewrite rule and / or method invocations to model the desired handler operation.

4 Realization

To realize the modeling of web applications with our Action Chart approach it was necessary to adapt the existing Fujaba code generation concepts for statecharts.

Our new code generation concept turns each action into an inner class of the class owning the Action Chart. These inner classes have a `doAction` method that implements the corresponding graph rewrite rules. In addition, these inner classes inherit from our runtime library class `FAction` that in turn implements the interfaces of various GUI listeners, property change listeners, RPC callback classes, and GWT timer classes. Cf. Figure 5, which also shows an excerpt of the classes of the runtime library. Class `FAction` implements the different listener methods such that the action specific `doAction` is called.

In addition, we still generate an `init` method into the owning class that creates at runtime an object structure that reflects the Action Chart structure. We also generate a `start` method that lazily calls the `init` method if the Action Chart object structure is not yet in place and that invokes the `doAction` of the initial action thus starting the Action Chart execution. At the end of each `doAction` a `doAutoTransitions` operation is invoked. This operation is generated within

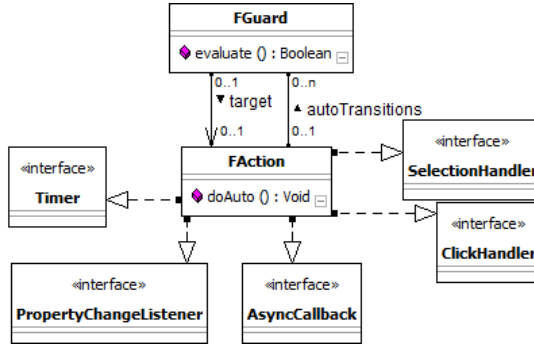


Fig. 5. Classdiagram showing parts of the runtime library

each `FAction` subclass. The `doAutoTransitions` operation evaluates the auto transitions leaving the corresponding action, evaluates the guard conditions and if true, the `doAutoTransitions` operation calls the `doAction` of the following action object. This realizes the synchronous execution of auto transitions in our approach.

If a listener transition leaves an action, we generate appropriate `addListenerers()` method calls within the `doAction` of that action, which adds the necessary listeners to the modeled objects at runtime. If e.g. a GUI event occurs, the corresponding listener object is informed via the usual listener mechanism and this eventually invokes the `doAction` of that listener action object.

As discussed, when calling a server operation via a RPC, we pass the current action object as the callback object parameter. Thus, on server response, the RPC runtime mechanism calls either method `onSuccess` or `onFailure` on the current action object. In class `FAction` we implement method `onSuccess` to lookup `success` links leaving the current action / callback object and to invoke the `doAction` of the reached action object. The `onFailure` method follows `failure` links, accordingly. In addition, the `resultValue` of the reached action object is set appropriately.

In our existing code generation concept for statecharts, each runtime statechart interpreter was running in its own thread. In a web browser, the whole Javascript application of a single web page runs in one single thread. The Javascript engine of this single thread handles all GUI events, timer events and RPC callback events. Thus, all these events are sequentialized by the Javascript engine and only one event is handled at a given time. This relieves us from many concurrency issues.

5 Related Work

The model driven development of web applications is analyzed by many research groups over the world. From our perspective there have been four that seem to be the most relevant ones.

First of all, there is the Object-Oriented Hypermedia Design Method (OOHDM) [14]. This method comprises of four steps: conceptual modeling, navigational design, abstract interface design and implementation. A second method, is UML-based Web Engineering (UWE) [10]. UWE consists of five different phases and uses UML notations to model the web application. Another modeling approach mostly used for the development of e-commerce applications and hypermedia information systems is given with the Object-Oriented Web-Solutions (OOWS) [11]. This approach is supported by a commercial tool called OlivaNova. Lastly the Web Modeling Language (WebML) [2] introduces a notation to specify at a conceptual level complex websites. This notation is supported by a development environment called WebRatio in which web applications can be modeled and automatically generated.

Nevertheless, the four approaches mentioned before were intended to develop traditional page based web application, while the approach presented in this paper clearly focuses on Rich Internet /Ajax Applications. Model driven development of these kind of applications has become a major research activity in the area of model driven web engineering. There already exist a few approaches in the literature that deal with the special issues raised by Rich Internet Applications. For example there exists an extension for the UWE method intended for the development of RIAs [8]. Also the OOHDM modeling approach has been extended for the new needs of Rich Internet Applications, as can be seen in [9].

One main difference between these existing approaches and the modeling approach introduced in this paper is our ability to model application logic. This can be done even inside the Action Charts which are used to create event listeners, asynchronous remote procedure calls and other client side logic. The whole application can be modeled, no switch to code is needed.

[9] and [12] introduce modeling approaches which seem very close to the one introduced here, first. But in [9] and [12] it is necessary to have different models for navigation, application logic and the orchestration of user interface components. The orchestration model uses statecharts as well, but is different in the handling of these. Using story diagrams inside our states we are able to clearly express inside the statechart which actions to the application model and the user interface have to be invoked when a state is reached. Additionally, applications created using our approach only have one entry point. This way, we completely avoid page changes inside the application. [7] are able to create events and timers using statecharts, as well. Again we see two benefits using the Fujaba approach. First we again have graph rewrite rules, which enable us to model complex object graph queries and modifications. Secondly, using our code generation, we provide the complete MVC pattern in our web applications. This means we are able to react to property change events occurring on client side. Such property changes can be modeled using statecharts, too.

6 Summary and Future Work

We have presented a new modeling approach for web applications. This approach allows to model RPC callbacks, GUI listeners, property change listeners,

and timers in uniform way within Action Charts. The actual model query and modification operations are modeled using graph rewrite rules embedded in the actions. These rules share their object variables. Thus, it is easy to pass common model objects from one (construction) action to a subsequent (callback) action or to listener actions. Due to our experiences this is a tremendous improvement of the old situation, where each callback and each listener had to be modeled in its own class and passing model elements required explicit attributes in those classes. Challenging issues for the future of client-side web applications arise from the more and more growing capabilities of the browser runtime environment. We are exploring client-side persistency mechanisms to finally build a distributed persistence layer covering many client nodes and the server, which should introduce redundancy only when necessary or beneficial. With our new Action Chart approach, we are now able to model all parts of a web application, its GUI construction, the GUI handlers, RPC callbacks, timers, property change listeners, persistency and replication mechanisms, and client side model data in a concise and uniform notation. This paper presents a very simple web application. In the context of our FAST project [3] we have used this approach to build a relatively complex data transformation tool and a data type definition tool. Thus, we are confident that the approach scales to real world applications. Full support for all user input events (mouse actions, CSS events) is work in progress while our examples evolve. However there is still more future work to be done: Defining the user interface using story diagrams is somehow more intuitive than textually coding it, but is still cumbersome. With graphically modeled GUI widgets, one can see more clearly the relations, but the graph approach doesn't satisfy the hierarchical property of GUI components (there's always a root component with child components). A lot of components have to be created and placed in order to form a graphical user interface that is capable of doing its designated tasks. To ease this step of the design phase for web applications, we currently develop a What-You-See-Is-What-You-Get Editor for GWT and SmartGWT widgets. You can review the whole designer idea in [6].

Acknowledgments

This work is partially supported by the European Commission under the first call of its Seventh Framework Program (FAST STREP Project, grant INFSO-ICT-216048).

References

1. Aschenbrenner, N., Dreyer, J., Hahn, M., Jubeh, R., Schneider, C., Zündorf, A.: Building Distributed Web Applications based on Model Versioning with CoObRA: an Experience Report. In: Proc. 2009 Intl. Workshop on Comparison and Versioning of Software Models, pp. 19–24. ACM, New York (May 2009)
2. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Elsevier, Amsterdam (December 2002)

3. The FAST Project (2010), <http://fast-fp7project.morfeo-project.org/>
4. Fischer, T., Niere, J., Torunski, L., Zündorf, A.: Story diagrams: A new graph rewrite language based on the unified modeling language. In: Proc. of the 6th International Workshop on Theory and Application of Graph Transformation, Paderborn, Germany (1998)
5. Geiger, L., Schneider, C., Reckord, C.: Template- and modelbased code generation for MDA-Tools. In: 3rd International Fujaba Days, Paderborn, Germany (September 2005)
6. Geiger, N., Eickhoff, C., Hahn, M., Witzky, I., Zündorf, A.: Future web application development with fujaba. In: Gorp, P.V. (ed.) Proceedings of the 7th International Fujaba Days, pp. 51–55 (November 2009)
7. Koch, N., Pigerl, M., Zhang, G., Morozova, T.: Patterns for the model-based development of rias. In: Gaedke, M., Grossniklaus, M., Díaz, O.(eds.) ICWE 2009. LNCS, vol. 5648, pp. 283–291. Springer, Heidelberg (2009)
8. Machado, L., Filho, O., Jo, R.: Uwe-r: an extension to a web engineering methodology for rich internet applications. WSEAS Trans. Info. Sci. and App. 6(4), 601–610 (2009)
9. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A model-driven development for gwt-based rich internet applications with ooh4ria. In: Schwabe, D., Curbera, F., Dantzig, P. (eds.) ICWE, pp. 13–23. IEEE, Los Alamitos (2008)
10. Nora Koch, A.K.: The expressive power of uml-based web engineering, pp. 105–119 (2002)
11. Oscar Pastor, J.F., Abrahao, S.: An object-oriented approach to automate web applications development. In: Electronic Commerce and Web Technologies, pp. 16–28 (2001)
12. Pérez, S., Díaz, O., Meliá, S., Gómez, J.: Facing interaction-rich rias: The orchestration model. In: Schwabe, D., Curbera, F., Dantzig, P. (eds.) ICWE, pp. 24–37. IEEE, Los Alamitos (2008)
13. Schneider, C.: CoObRA: Eine Plattform zur Verteilung und Replikation komplexer Objektstrukturen mit optimistischen Sperrkonzepten. PhD thesis (2007)
14. Schwabe, D., Rossi, G.: The object-oriented hypermedia design model. ACM Commun. 38(8), 45–46 (1995)

Modeling Search Computing Applications

Alessandro Bozzon, Marco Brambilla, Alessandro Campi, Stefano Ceri,
Francesco Corcoglioniti, Piero Fraternali, and Salvatore Vadacca

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
V. Ponzio 34/5, 20133 Milano, Italy
{bozzon,mbrambill,campi,ceri,corcoglioniti,
fraterna,vadacca}@elet.polimi.it

Abstract. Search Computing defines a new class of applications, which enable *end users* to perform exploratory search processes over multi-domain data sources available on the Web. These applications exploit suitable models, supported by a framework, that make it possible for *expert users* to configure the data sources to be searched and the interfaces for query submission and result visualization, by using for such source and interface configurations mash-up tools which do not require programming. This paper presents Search Computing design process and developer roles, together with the conceptual models that represent the foundation of a model-driven approach to Search Computing.

Keywords: Search Computing, development process, search engine, models.

1 Introduction

Although Internet search is primarily performed by means of search engines, not all information needs are satisfied by individual Web pages. Web search queries become more and more complex: their formulation does not fit in few keywords, the result is richer than a simple list of Web pages, and general-purpose search engines perform poorly upon them. *Vertical search engines* (e.g., *Expedia* or *Lastminute* in the travel sector) provide a partial solution: they aggregate domain-specific information from a fixed set of sources; they do a better job than general-purpose search engines *in their domain*, but cannot be used or extended for other topics.

Complex *search processes* [1] addressing different domains remain to be supported. For example, a user may wish to find a place where an interesting event occurs, that has good weather in a given period, with travel and accommodation options that match given budget and quality standards, maybe with close-by trekking opportunities. A new class of applications, *search computing applications* [8], aims at responding to **multi-domain queries** like the one above, i.e., queries over multiple semantic fields of interest, by helping users to decompose queries and assemble complete results from partial answers; this class of applications fills the gap between generalized search systems, which are unable to find information spanning multiple topics, and domain-specific search systems, which cannot go beyond their domain limits. Examples of Search Computing queries are: “Where can I attend an interesting

scientific conference in my field and at the same time relax on a beautiful beach nearby?” or “Where is the theatre closest to my hotel, offering a high rank action movie and a near-by pizzeria?”. Further motivating examples (and online demonstrations) can be found online at: <http://demo.search-computing.com/>.

Data source integration is a nontrivial task *per se*, as its solution requires solving schema matching problems, providing suitable coercion functions in the context of mismatching concrete types, and applying data cleansing and entity identification[15]. Ranked data source integration adds the problem of defining global rank functions over heterogeneous ranking attributes. Multi-domain queries are answered by selecting a subset of the available search services, by individually extracting their responses, and then by joining them thereby building combinations that collectively constitute the results of the query; combined and ranked results must be presented to the user for inspection and query refinement.

In this paper, we advocate a model-based stepwise refinement approach to the development of search computing application, which distinguishes the role of the developer, domain expert and end user, and adopts models to progressively abstract the complexity of query processing and data source wrapping.

The contribution of this paper is to discuss how to: (1) organize the development tasks in a coherent process, (2) identify the actors involved in such process, and (3) illustrate the models of the essential concepts that constitute a search computing application.

The paper is organized as follows: Section 2 gives an overview of the Search Computing framework; Section 3 presents the roles and process for SeCo application development; Section 4 discusses the main models managed by Search Computing framework; Section 5 discusses the related works and Section 6 concludes.

2 The Search Computing Framework

A Search Computing application supports users in asking multi-domain queries; for instance, “Where can I attend a scientific conference close to a beautiful beach reachable with cheap flights?” This paper proposes an approach for building Search Computing applications by exploiting a configurable framework that delegates domain-neutral computation to a generic architecture which is configured with domain-dependent information and with the help of suitable models. As shown in Figure 1, several sub-frameworks constitute the Search Computing framework.

The *Service Mart Framework* provides the scaffolding for wrapping and registering data sources. Data sources can be heterogeneous (examples include Web site wrappers, Restful data sources, WSDL web services, data sources exposed by means of the Yahoo! Query Language [<http://developer.yahoo.com/yql/>], etc.).

A service mart is defined as an abstraction (e.g., Hotel) of one or more concrete Web services (e.g., Bookings and Expedia), each capable of accepting queries and of returning results, possibly ranked and chunked into pages. Registration metadata describes the service mart signature (input and output data types) and connection information. A connection is defined as an input-output relationship between pairs of service marts that can be exploited for joining them.

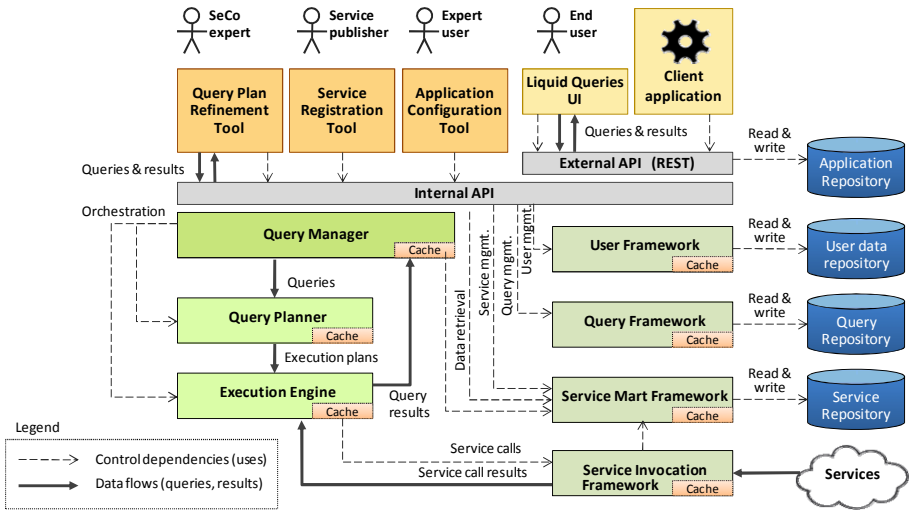


Fig. 1. Overview of the Search Computing framework

The *User Framework* provides functionality and storage for registering users, with different roles and capabilities. The *Query Framework* supports the management and storage of queries as first class citizens: a query can be executed, saved, and modified.

The *Service Invocation Framework* masks the technical issues involved in the interaction with the registered service marts, e.g., the Web service protocol and data caching issues.

The *Query Processing Framework* provides the service for executing multi-domain queries. The *Query Manager* splits the query into sub-queries and binds them to the respective relevant data sources; the *Query Planner* produces an optimized plan with the sequence of steps for executing the query; finally, the *Execution Engine* executes the query plan, by submitting the service calls to designated services and then building the query results by combining the outputs produced by the called services.

Information persistence is delegated to dedicated object stores: the *service repository* (storing service mart descriptions), the *query repository* (storing queries saved by users), the *user data repository* (storing profiles information), and the *application repository* (storing application configurations). To obtain a specific application, the general-purpose architecture of Figure 1 is customized with the help of models and tools targeted to programmers, expert users, and end users.

3 Development Process and Roles

The development process of Search Computing applications involves actors with different roles and expertise:

- **Service Publishers** are in charge of wrapping data sources, to make them compatible with the service mart standard interface, and register service mart definitions and their connections in the service repository.

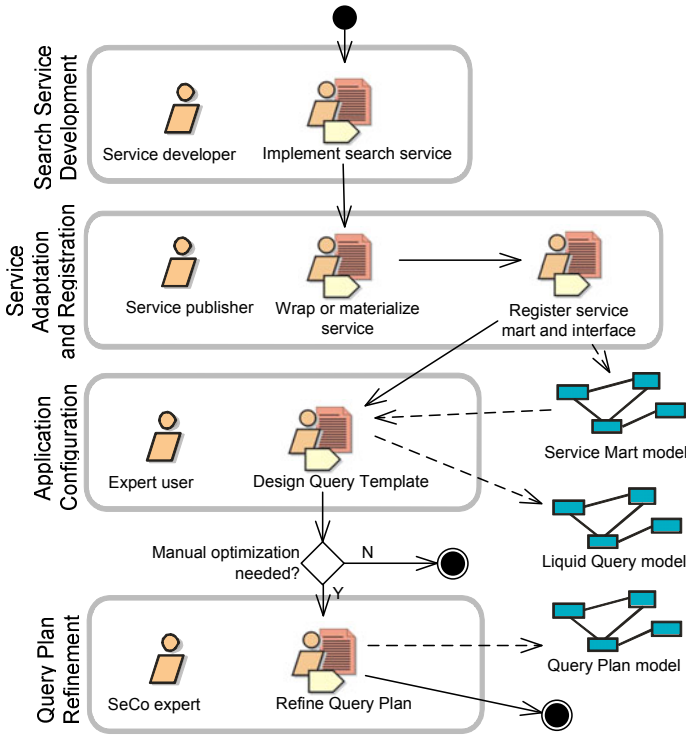


Fig. 2. Development process for SeCo applications (SPEM notation)

- **Expert Users** configure Search Computing applications, by selecting the services of interest, choosing the connection patterns to be used, and configuring the complexity of the user interface.
- **End Users** use Search Computing applications configured by expert users according to an exploratory information seeking approach [3].

These user's roles operate according to the SPEM process scheme shown in Fig. 2:

- **Search service development:** this phase consists in the development of Web services exposing on the Web the data requested by the application; it must be done in all those situations in which the data consumed by the Search Computing application are not available as Web services and must therefore be exposed, e.g., by wrapping a legacy system by means of a Web service interface; the activity is done by programmers, since it involves the production of the adapter code necessary to expose legacy data sources as Web services.
- **Service adaptation and registration:** this phase is necessary in the majority of cases, in which the data required by the application are already present on the Web, but the Web services that expose them do not adhere to the service mart interface and protocol. Therefore, several activities are needed for adapting the existing services to the Search Computing framework: creation of service wrappers, possible design of data caching policies, normalization of the data, and registration as service

marts in the service repository; these tasks are performed by service publishers, which need programming skills for the construction of data and protocol adaptation components. However, since the service mart APIs and protocol are fixed, the development of adapters can be supported by a template-based approach, in which standard adapter components are tailored to the specific Web service to wrap.

- **Application Configuration:** this phase consists in configuring the Service Invocation Framework and the User Interface and is conducted by an expert user. The former task requires selecting from the service repository several service marts and service implementations (for those service marts associated with multiple sources) and in declaring their configuration, including the input and output parameters (in case a service can be called in different ways) and the connection patterns (used for joining pairs of services). Notice that multiple implementations could be available for the same service mart (e.g., Expedia and eDreams) and the same pair of services could be connected in different ways (e.g., theatres offering musical events could be connected to GIS data sources by using location names or geographic coordinates).

The GUI configuration activity requires customizing the structure of a generic interface, by choosing: 1) optional selection predicates to restrict the objects retrieved by the query (e.g., a maximum price target for events or flights); 2) default ranking criteria for the results; 3) visual preferences on the display of the result set (e.g., sorting, grouping, and clustering attributes or the size of the result list); 4) a set of extra service marts usable by the end user to expand the current query (e.g., to expand a query on movies by means of a service that joins selected movies to their reviews and the theatres where they are programmed to nearby restaurants).

- **Query plan refinement:** this phase, in charge to the SeCo expert, consists in manually refining the optimized query plan produced by the SeCo platform starting from the query specification. In general, query plans are self-tuned, and therefore this phase is usually not needed. However, an expert can decide to manually override the optimal plans to comply with complex queries, or manually select alternative data sources, or improve scalability through parallelism, or provide customized choices not covered by the optimization.

The previous development steps lead to the final application accessed by the end user. The GUI, instantiated during the application configuration phase, supports the “search as a process” paradigm, based on the continuous evolution, manipulation, and extension of queries and results; the query lifecycle consists of iterations of the steps of **query submission**, when the end user submits an initial query; **query execution**, producing a result set that is displayed in the user interface; and **result browsing**, when the result can be inspected and manipulated through appropriate interaction primitives, which update either the result set (e.g., re-ranking or clustering the results) or the query (e.g., by expanding it with additional service marts or requesting for more results) [3].

The described development process takes into account the trend towards empowerment of the user, as witnessed in the field of Web mash-ups [10]. Indeed, only service development and service mart adaptation require programming expertise. All the other design activities are moved to service registration time and to application configuration

time, so that designers only need a conceptual understanding of services and queries, and do not need to perform low-level programming.

4 Search Computing Object Models

In this section we define the models describing the main objects involved in Search Computing applications. For illustration, throughout the paper we use a running example, in which a user plans a leisure trip, and wants to search for upcoming concerts (described in terms of music type, e.g., Jazz, Rock, Pop, etc.) close to a specified location (described in terms of kind of place, like beach, lake, mountain, seaside, and so on), considering also availability of good, close-by hotels. Additionally, the user can expand the query with information about available, nearby good quality restaurants for the candidate concert locations, the news associated to the event, photos that are taken close to the location, and possible options to combine further events scheduled in the same days and located in a close-by place.

4.1 Service Marts Model

Service marts are abstractions that represent the properties (attributes and data types) of Web objects in a standardized way, according to the concepts of the Marts package shown in Fig. 3. Every service mart definition includes a name and a signature (a collection of exposed attributes). Attributes are strongly typed and can be atomic (i.e., defined by single-valued, basic types), composed (i.e., defined by a set of sub-attributes), or multi-valued (i.e., allowing multiple instances). The association between service marts is expressed by **connection patterns** (composition package of Fig. 3). Every pattern is described by a conceptual Name and a set of comparison predicates between pairs of attributes of the two services, which are interpreted as a conjunctive Boolean expression.

The Patterns package of Fig. 3 describes a lower level characterization, whereby each service mart is associated with one or more access patterns. An **access pattern** is a specific signature of the service mart in which each attribute is denoted as either input (I) or output (O), depending on the role that it plays in the service call. In the context of logical databases, an assignment of I/O labels to the attributes of a predicate is called predicate adornment [7]. Moreover, an output attribute can be designed as ranked (R).

Finally, **service interfaces** (Interfaces package of Fig. 3) describe the physical implementations of services. Two categories of service interfaces are possible: *search service* (i.e., one producing ranked sets of objects) or an *exact service* (i.e., services producing unranked sets of objects that satisfy a condition). Service interfaces are characterized by *chunking size* (number of result instances in the chunk), *caching properties*, and *cost descriptors* (e.g., the *response time* and/or as the *monetary cost of invocation*). Examples of access patterns for the running case are:

```

Concert(location[I], radius[I], minDate[I], maxDate[I], genre[I], name[O],
         date [O][R], lat[O], long[O], distance[O][R], Price[O][R], address [O])
Restaurant(category[I], minRating[I], location[I], radius[I], name[O],
            address[O], lat[O], long[O], Rating[O][R], distance[O][R], url [O])

```

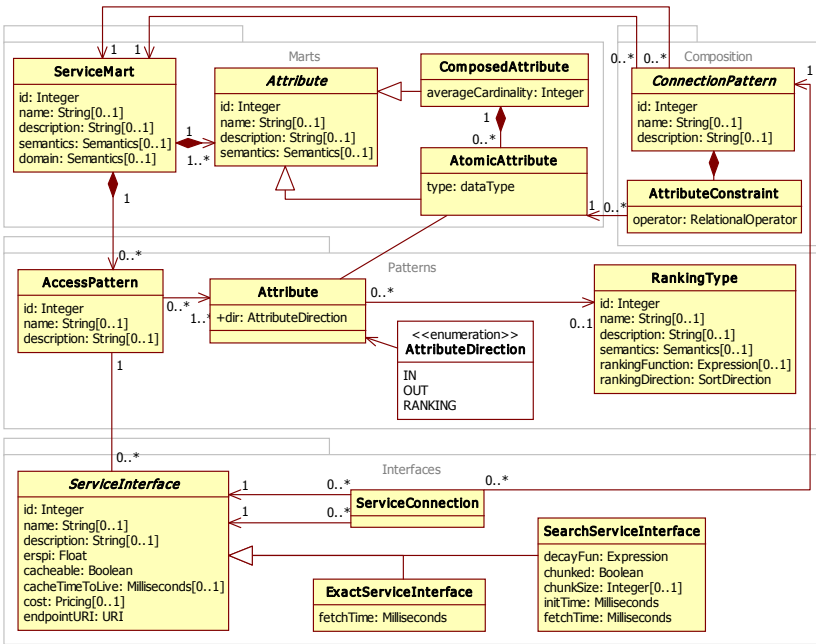


Fig. 3. Service mart model

4.2 Query Model

A *Search Computing query* is a conjunctive query over services, which includes two main aspects: the logical query clauses over the relevant data sources and the result ranking criterion. Fig. 4 shows that a query clause can refer to the service mart level (and thus requires automatic translation down to the concrete service interfaces) or at the Service Interface level. A clause may describe the service to invoke, conditional predicates over service attributes, join operations, and ranking of results.

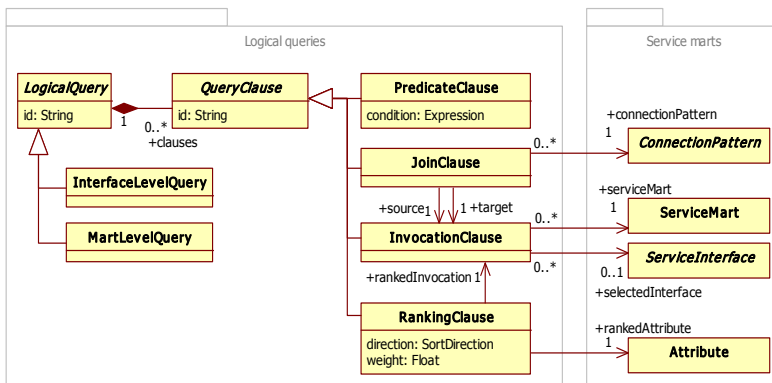


Fig. 4. Query model

An example of complex query is “Where can I find a jazz concert in San Francisco close to a nice vegetarian restaurant?” represented as:

```

Concert("San Francisco", "1.0 m", "10/30/2010", "11/30/2010", "Jazz", name [O],
date [O] [R], lat [O], long [O], distance [O] [R], Price [O] [R], address [O])
Restaurant("vegetarian", "3stars", {Concert.lat, Concert.long}, "1.0 m",
name [O], address [O], lat [O], long [O], Rating [O] [R], distance [O] [R], url [O])
    
```

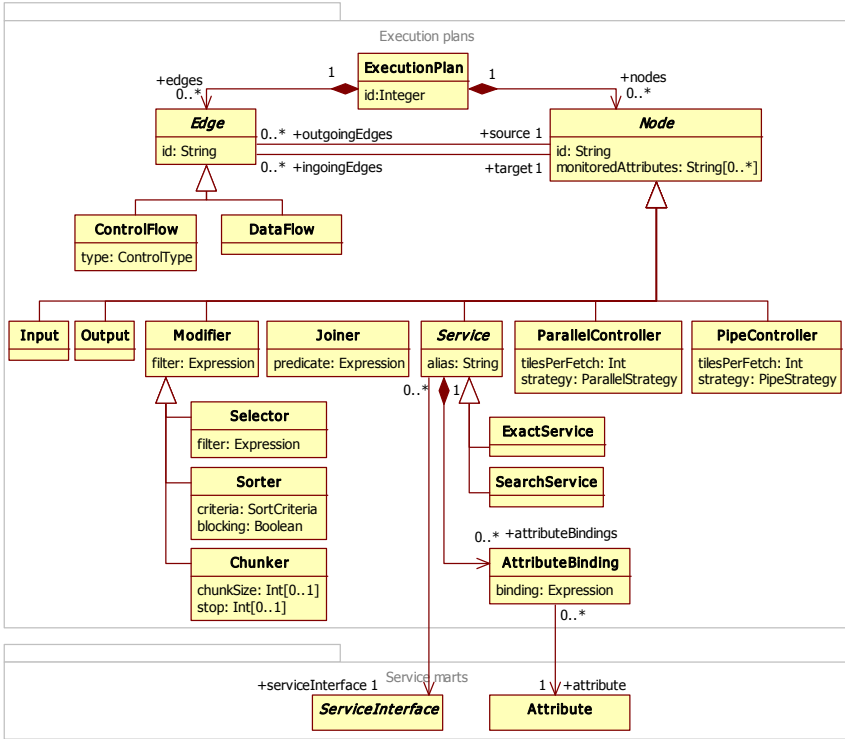


Fig. 5. Query plan model

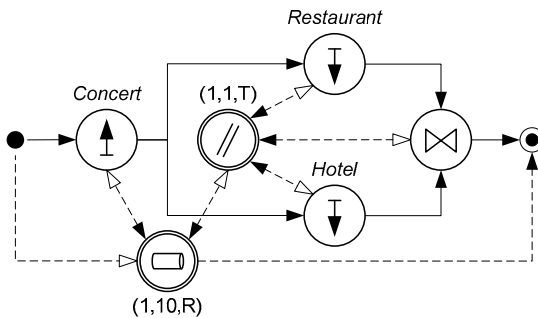


Fig. 6. Example of query plan according to the *SeCo Panta Rhei* DSL

4.3 Query Plan Model

A *query plan* is a well-defined scheduling of service invocations, possibly parallelized, that complies with their service interface and exploits the ranking in which search services return results to rank the combined query results. A query plan (see Fig. 5) is a graph composed by nodes (invocations of services or other operators) and edges (control flow and the data flow between nodes). Several types of nodes exist, including service invocators, joiners, controllers and modifiers (chunkers, sorters, selectors), according to the *Panta Rhei Domain Specific Language* [5].

As an example of query plan model, Fig. 6 shows a pipe join which first extracts a list of *Concerts* from a search service and then uses these results to retrieve close-by *Restaurants* and *Hotels*, whose invocations are performed in parallel according to a predefined join strategy. Finally, results are joined by suitable join units, which execute the join of search engine results [4]. Two strategy nodes—one for the outer pipe join and one for the inner parallel join—orchestrate the execution of the query by observing the output of data units (such as service invocators and joiners) and deciding the service to be invoked accordingly.

4.4 Interaction Model

Fig. 7 shows the model that describes the set of abstractions supporting exploratory search within the Search Computing user interface. Following the distinction of the development process into application configuration and usage, the model represents interaction at two levels. At the specification level, when configuring an application, a user’s query is defined as a selection of an underlying *logical query*, which is in turn

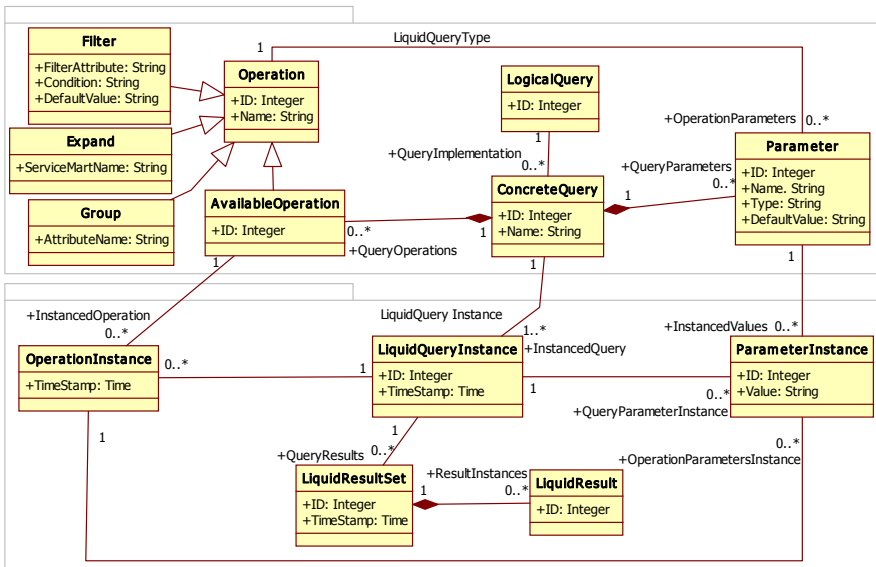


Fig. 7. The interaction model

Combinations		Events					Hotels			Restaurant		
S ID	Score	C Title	C Distance	E Start Date	E Start Time	V Venue name	H Title	H Distance	H Avg rating	R Title	R Distance	R Avg rating
0_0	1.138	Dee Dee Bridgwater	0.03	2009-10-30	20:00:00	Herbst Theater	Embassy Hotel	0.33	4	Bonobus Restaurant	0.48	4
0_1	1.040	Dee Dee Bridgwater	0.03	2009-10-30	20:00:00	Herbst Theater	Commodore Hotel	0.72	4	Nine Restaurant	0.48	4
0_2	1.018	Dee Dee Bridgwater	0.03	2009-10-30	20:00:00	Herbst Theater	Villa Fiorance Hotel	0.83	4	Nine Restaurant	0.48	4
0_3	0.998	Dee Dee Bridgwater	0.03	2009-10-30	20:00:00	Herbst Theater	Embassy Hotel	0.23	4	Bonobus Restaurant	0.7	4
0_18	0.989	Nicholas Payton	0.87	2009-10-30	20:00:00	Grace Cathedral Episcopal Church	Commodore Hotel	0.23	4	Bonobus Restaurant	0.26	4
0_21	0.982	Nicholas Payton	0.87	2009-10-30	20:00:00	Grace Cathedral Episcopal Church	Commodore Hotel	0.23	4	Postrio Restaurant	0.27	4
0_19	0.979	Nicholas Payton	0.87	2009-10-30	20:00:00	Grace Cathedral Episcopal Church	Hotel Vertigo	0.20	4	Bonobus Restaurant	0.26	4
0_22	0.972	Nicholas Payton	0.87	2009-10-30	20:00:00	Grace Cathedral Episcopal Church	Hotel Vertigo	0.28	4	Postrio Restaurant	0.27	4
0_6	0.954	Dee Dee Bridgwater	0.03	2009-10-30	20:00:00	Herbst Theater	Embassy Hotel	0.23	4	Spoko's Restaurant	0.77	4
0_20	0.953	Nicholas Payton	0.87	2009-10-30	20:00:00	Grace Cathedral Episcopal Church	Villa Fiorance Hotel	0.41	4	Bonobus Restaurant	0.26	4

Fig. 8. Example of resultset retrieved by the system for the Concert-Restaurant-Hotel case (see the online demo at <http://demo.search-computing.org>)

mapped to a *concrete query*. Such concrete query is then associated with: 1) a set of input parameters; 2) a set of *AvailableOperations* (e.g., Filter, Group, Expand, etc.). Operations might be parametric, thus requiring the end user to supply parameters.

Once specified, a logical query can be instantiated at runtime (as denoted by the *QueryInstantiation* class). When a query is instantiated, the query parameters assume a value, thus allowing the execution engine to process the associated logical query, which in turn produces a *ResultSet*, composed by a set of *Results*. Then, a user can decide to further explore the results by applying one of the *AvailableOperations*, thus altering the *ResultSet* according to the semantic of the selected operation. Some operations simply apply local transformations to the extracted results (e.g., re-sorting, grouping, and so on), while others need the intervention of the query execution engine, e.g., to ask for more results or to expand the result set with information on additional service marts; in the latter case, in particular, some additional pieces of query plans can be activated. Fig. 8 shows an example of result table for the running case. An online demo is available at: <http://demo.search-computing.org>.

5 Related Work

Model-driven engineering is the main research field SeCo takes inspiration from. All the SeCo artifacts are modeled as conceptual entities and are managed by model-driven transformations (a coarse approach to the transformations is described in [6]). The existing works in the field, which include general-purpose MDA techniques and web-specific languages and tools (e.g., WebML [9], OO-HMETHOD [11], UWE [13], HERA, and others), do not consider the search-specific aspects of application development and in particular completely ignore the need for an intermediate role between the user and the developer (that we call expert user) that has the duty of configuring search applications starting from basic services.

None of the **tools for object-oriented design**, including the ones focusing on databases (e.g., Oracle JDeveloper 10g [<http://www.oracle.com/tools>]), on Web applications design (e.g., Code Charge Studio [<http://www.codecharge.com>]), WebRatio

[www.webratio.com]), and standard application design (e.g., Rational Rapid Developer or jABC, Java Application Building Center), explicitly support the design of search applications. From various tools, SeCo borrows the ideas of *visual composition* of the applications and *automatic deployment* of the running prototype.

Mashup approaches are even more inspiring in this sense, since they exactly comply with the SeCo expert users need of an easy online toolsuite to quickly configure and deploy applications. The most famous mashup environments are Yahoo Pipes and Microsoft Popfly (recently discontinued), together with several scientific investigations still ongoing (e.g., [12] proposes a spreadsheet based mashup development framework).

The **service engineering** field provides service description languages and protocols such as WSDL and SOAP, with limited support in mechanizing service recognition, combination, and negotiation. Proposals like WSFL, DAML-S, OWL-S, and WSMF attempt at formalizing the semantics of Web services using ontology technology. Data exchange formats (e.g., SOAP-XML, JSON, and others) and service orchestration languages (BPEL, BPMN, and so on) inspired our approach. Other ongoing efforts such as W3C's Web Services Architecture and OASIS standards aim at creating a framework for service modeling. With respect to the SOA terminology of orchestration and choreography, we adopt a centralized orchestration approach.

Finally, **search-based application development** has largely inspired SeCo too. The Symphony platform by Microsoft enables non-developers to build and deploy search-driven applications that combine their data and domain expertise with content from search engines and other Web Services [14]. Other approaches to search-based development (like Google Base API [<http://code.google.com/apis/base/>] and YQL, Yahoo Query Language [<http://developer.yahoo.com/yql/>]) target the skilled software developer. These solutions, combined with mashup approaches, can get close to the SeCo approach, although several critical aspects would still be different (e.g., join and ranking would be missing anyway).

With respect to our previous work, this paper advances in the direction of exploring the development process of search applications. A preliminary proposal for process and roles was introduced in [2], while model transformations are discussed in [6]. In this paper instead we refined the process, we redefined the user roles and provided a detailed the conceptual models (with a set of sample instances).

6 Conclusions

Search Computing is an emerging paradigm for supporting complex search. Search Computing design choices are supported by a cohesive framework which integrates several interacting models, thereby partitioning the design space and responsibilities to the different roles and involved expertise, in a non-trivial way; the objective is to replace programming with model driven development wherever possible, yielding to flexibility and efficiency.

Acknowledgements. This research is part of the Search Computing (Seco) project, funded by ERC, under the 2008 Call for "IDEAS Advanced Grants".

References

- [1] Baeza-Yates, R., Raghavan, P.: Next Generation Web Search. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 11–23. Springer, Heidelberg (2010)
- [2] Bozzon, A., Brambilla, M., Ceri, S., Corcoglioniti, F., Gatti, N.: Building search computing applications. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 268–290. Springer, Heidelberg (2010)
- [3] Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid Query: Multi-Domain Exploratory Search on the Web. In: World Wide Web Conference (WWW 2010), Raleigh, USA, pp. 161–170. ACM, New York (April 2010)
- [4] Braga, D., Campi, A., Ceri, S., Raffio, A.: Joining the results of heterogeneous search engines. *Information Systems* 33(7-8), 658–680 (2008)
- [5] Braga, D., Ceri, S., Corcoglioniti, F., Grossniklaus, M.: Panta Rhei: A Query Execution Environment. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 225–243. Springer, Heidelberg (2010)
- [6] Brambilla, M., Ceri, S., Tisi, M.: Search Computing - A Model-Driven Perspective. In: Tratt, L., Gogolla, M. (eds.) Theory and Practice of Model Transformations. LNCS, vol. 6142, pp. 1–15. Springer, Heidelberg (2010)
- [7] Cali, A., Martinenghi, D.: Querying Data under Access Limitations. In: ICDE 2008, pp. 50–59 (2008)
- [8] Ceri, S., Brambilla, M. (eds.): Search Computing. LNCS, vol. 5950. Springer, Heidelberg (March 2010)
- [9] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, USA (December 2002) ISBN 1-55860-843-5
- [10] Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing* 11(3), 59–66 (2007)
- [11] Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. *IEEE MultiMedia* 8(2), 26–39 (2001)
- [12] Kongdenfha, W., Benatallah, B., Vayssière, J., Saint-Paul, R., Casati, F.: Rapid development of spreadsheet-based web mashups. In: WWW 2009, Madrid, pp. 851–860. ACM, New York (April 2009)
- [13] Knapp, A., Koch, N., Moser, F., Zhang, G.: ArgoUWE: A CASE Tool for Web Applications. In: EMSISE Workshop (2003)
- [14] Shafer, J.C., Agrawal, R., Lauw, H.W.: Symphony: Enabling Search-Driven Applications. In: USETIM (Using Search Engine Technology for Information Management) Workshop, VLDB Lyon (2009)
- [15] Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. *Information Systems* 26(8), 607–633 (2001)

Developing Security Assessment Models in Web² Mobile Environments

Bong Gyou Lee, Hyunsik Seo, Giseob Byun, Keon Chul Park,
Soo Kyung Park, and Taisiya Kim

Graduate School of Information, Yonsei University,
134 Shinchondong, Seodaemungu, Seoul, Korea
{bglee, seohs, parkkc, tgbgs, sk.park, lucky8619}@yonsei.ac.kr

Abstract. The purpose of this study is to develop and present security assessment models to manage the quality of Web² mobile environments. Web² is an evolved concept of web from Web 1.0 and Web 2.0 which means, diverse services and technologies including Real-time service, Social Network Services, Augmented Reality technology and Location Based Service are realized in mobile environment. However, compared to Web 1.0 and Web 2.0, few studies have been conducted for the security issues of Web² mobile environment. To be prepared for such issues, this paper reviews the characteristics of security in Web² mobile environments and present security assessment models in perspectives of Sensor-generated threats and User-generated threats. This study is significant in that it presents the directionality of the security issues to be discussed later in Web² mobile environments. This study will have implications for businesses and researchers preparing Web² mobile services and marketing.

Keywords: Web² Mobile Environment, Security Assessment, Sensor-generated Threats, User-generated Threats.

1 Introduction

Recently, the development of mobile network infrastructures, mobile communication technologies, and terminal technologies; the advent of diverse applications; and the activation of application markets have led the global increases in the supply and use of smart phones. That is, with the evolution of mobile Internet networks such as WiFi and 3G that enable users to freely access and use the Internet anytime and anywhere; the upgrading of the performance of smart phones into all-in-one devices that enable document works and support replays of diverse multimedia; the advent of diverse applications that satisfy users' needs; the development of an operating system (OS) that supports the applications; and the activation of application markets where those applications can be bought and sold, smart phones are providing new opportunities for businesses and revenue creation in the saturated mobile market, where it has become difficult to create revenues with voice services [1]. However, diverse factors threatening these new opportunities provided by smart phones are also appearing. Of these, the fact that the possibility and areas that are exposed

to security threats are increasing is a serious problem [2]. In particular, it is considered that this will be a larger threat in Web² mobile services that enable attribute information collected from various kinds of sensors in real-time to be communicated from device to users and from device to device. Such services will also enable augmented reality technologies utilizing such information and Social Network Services (SNS) between users as information exchanges and access to and opening of networks increase. To review security related studies, a study that analyzed technical trends and the vulnerability of security in Web 1.0 and Web 2.0 environments [3], a study on the trend of security technologies and the direction to standardize the technologies [4], and a study on security issues and efficient measures to inspect the web [5] have been conducted. However, in reality, where as mobile Web² services are constantly evolving, the studies that have been conducted to prepare for the situation are insufficient. Therefore, the purpose of this study is to develop and present security assessment models in Web² mobile environments. To present a model used to prepare for security issues in Web² environments, it is necessary to assess the different types of security and the potential threats to each of them. In order to accomplish this, a framework will be developed through three broad stages. First, the characteristics of Web², which has evolved through Web 1.0 and Web 2.0, will be discussed. Second, based on these characteristics, security issues that might occur will be presented. Third, various threats to security that could occur in mobile environments will be reviewed; these will be mapped onto the security issues presented in the second stage. By illustrating the types, channels of occurrence, and the contents of damage of expected security issues, the security assessment model for Web² will be refined. This study has implications for businesses and related researchers preparing diverse services for Web², where paradigms are rapidly changing.

2 The Concept of Web²

2.1 Evolution of the Concept of Web

With the development of information technologies and user demands, the web has evolved from 1.0 to 2.0, and is now developing into Web² [6]. Web 1.0 consisted of text-centered HTML documents, information or services were produced by a few experts, and users had to accept them unilaterally. Later, the web became Web 2.0, thanks to the advent of sites that provided services using diverse data made into platforms through the participation of many people. In addition, technical changes that occurred at this time enabled the supply of web production tools to the public [7]. With these evolutions, the sharing and openness of information content were emphasized. Now, the availability of mobile Internet and smart phones is working as a catalyst for the development web applications. Web², a term coined by Tim O'Reilly (2009), refers to interactions between the web and the world, that is, the cyber world and the real world. In addition, the exponential expansion from Web 2.0 to Web², rather than an arithmetical increase to 3.0, means that the web is now becoming a reality in itself instead of an aggregate of static HTML pages intended to describe reality [3]. In Web², augmented realities, location information, social networks, etc.

are highlighted. Thus, it is becoming possible to obtain useful information on a user's surroundings in real time and manage personal connections [8]. Therefore, more sensors and people are introducing platform data and applications in order to actively induce sharing and participation.

2.2 Features of Web²

2.2.1 Sensor-Generated Information

In Web² environments, unique and diverse services are created and provided through sensor-generated information. Based on the microphones, cameras, motion sensors, proximity sensors, and location sensors that are built into smart phones, applications are appearing that utilize sensor-generated information [6]. Sensor networks such as digital cameras, mobile phones, LBS (Location Based Services) equipment, RFID (Radio-Frequency Identification), etc. are creating the phenomenon of an "Internet of Things" that connects humans with things and things with things. This phenomenon enables consumers to easily obtain historical or geographical information by attaching tags to information [9]. Representative utilizations of sensor-generated information include "augmented reality" (AR) which refers to technologies that combine virtual information with the real world in real time to give the information [10]. Unlike virtual realities, where all environments are produced as 3-dimensional computer images, here virtual information is superimposed on real images. Thus, the sense of reality is improved. In the case of the iPhone's Sekai Camera application, letter information (names, reputations, etc.) called "air tags" are provided along with photos and other information buildings, restaurants, etc. These are shown through the iPhone's camera. These services create added values such as convenience, sympathy in experiences, safety, efficiency, etc., and are actively applied in the mobile broadcasting, advertising, education, game, and medical manufacturing industries [11]. Large volumes of data in the real world come to meet the web in real time through sensor-generated services.

2.2.2 User-Generated Information

Web² has a feature called "crowd sourcing" that enables large groups of people to create collective works with higher values than the works of individual participants [12]; as it has become possible to share information by exchanging contents and services in real time through smart phones, the web has become much more interactive. The mapping between non-structured data generated by users and structured datasets is becoming a core capability of Web². Examples of this include Twitter, which is a network of micro-blogs; the "information cascades" posted on Twitter spread from Twitter to become fundamental sources of information for many people who want to know what happened. Operations such as adding data points on maps through web mapping applications take the form of collective intelligence through users' searches and responses in real time and information sharing in real time [6]. User-generated services are currently enjoying the limelight, as they provide relation-type services based on the characteristics of speed and novelty.

3 Security Issues in Web² Mobile Phones

3.1 Changes in Security Issues in Relation to the Evolution of the Web

As the web evolves, the technologies and services are rapidly converging and being standardized. However, matters related to security are not keeping pace [13]. As a result of this weakness, there have been attempts to illegally obtain numerous datasets provided in web services [14]. The early Web 1.0, designed to receive the information and services unilaterally provided by portal service businesses, was composed of HTML, URL, and HTTP. It provided the elementary interactions of simple clicks through links. To overcome the early static HTML environments, technologies such as the Java applet, Java script, and ActiveX appeared. However, they were directly related to the vulnerability of security [15] because using Java script, which had many weaknesses in security, or imprudently using ActiveX with unidentifiable providers, was fatal for security. Meanwhile, Web 2.0 services that provided user participation, sharing, and high openness included all the security issues in the existing web while having wider areas that could be attacked than the existing web services. This was due to the additional methods of accessing services and the interactive and asynchronous operation method of Web 2.0 [16]. For instance, in Web 2.0, an XML content feed that uses the RSS and Atom standards is used. This feed not only enables both users and web sites to obtain the headlines and texts of contents, but also basically enables users to see the summaries of relevant web sites. Unfortunately, there is a problem with this: it is not perceived that the applications and added systems used in these processes are vulnerable [3]. It is expected that issues of web security will not go away in the course of evolution into Web². In the case of Web² represented by AR, Real-Time Web, and SNS, it is expected that not only will security issues in the existing Web 2.0 be included, but the scope will also be widened due to the expansion of mobile services, the strengthening of mutual connectivity between users, and the use of sensor networks. Thus, security is becoming more important. This can be identified through recent research such as studies on security vulnerabilities in the mobile web [17]; these analyzed the mobile ubiquitous sensor network (USN) technology and cases of security vulnerabilities [18]; threats to security in SNS environments and related countermeasures, based on Europe ENISA reports [19]; and the sharing and protection of identities in SNS environments [20]. Therefore, it is expected that in Web², along with the security issues that have appeared in Web 2.0, additional security problems resulting from the fusion of diverse technologies and services will occur. In Web² mobile environments, threats to security will exist due to rapid increases in the volume of data communications using mobile sensors such as AR, followed by the volume of data created by users resulting from increased use of SNS. The changes in security issues resulting from the evolution of the web are shown in Table 1.

In the existing Web 2.0, it is considered that the data created through the participation of a small number of professionals would represent only a small part.

Table 1. Changes in security issues resulting from the evolution of the web

Classification	Web 1.0	Web 2.0	Web ²
Characteristics	Information and services unilaterally provided by portal service businesses	User participation, sharing, and high openness	Expansion of mobile services, strengthened mutual connectivity between users, and the use of sensor networks
Representative Technology and Services	HTML, ActiveX	AJAX, FLAX, XML, RSS, Atom, Tagging, LAMP	AR, Real-Time Web, SNS
Security	Weakness due to using ActiveX and dependence on OS/browser	Wider areas that may be attacked compared to Web 1.0 services due to the additional methods of accessing services and the interactive and asynchronous method of operation	Wider areas that may be attacked compared to Web 2.0 services due to the fusion of diverse technologies and services, such as the expansion of mobile services and the use of sensor networks

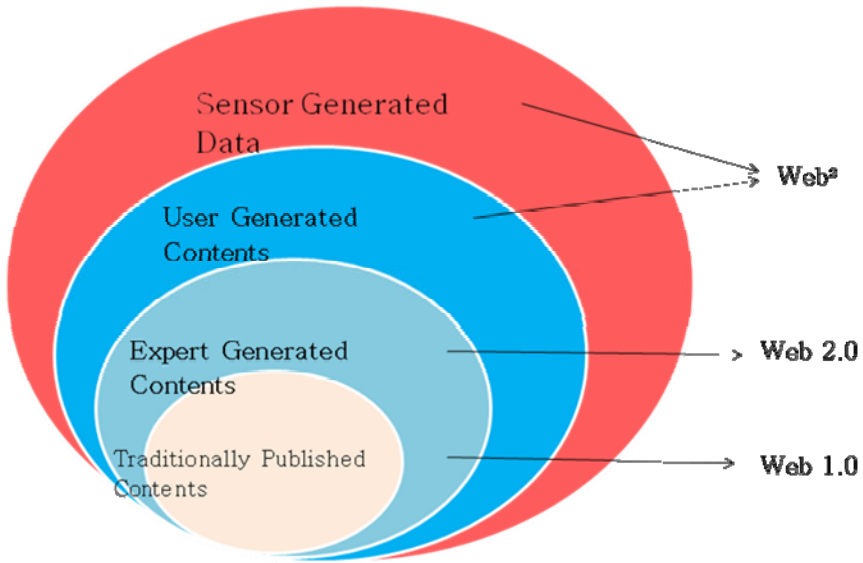


Fig. 1. Data types generated by Web²

3.2 Security Issues in Web²

For Web², it is expected that the security issues related to smart phones will further increased. In this study, the most important characteristics of Web² services were seen as sensor-generated information for the implementation of AR, etc., and user-generated information for the implementation of SNS. Therefore, the security issues that will appear in new forms in these two dimensions will be discussed as the most pertinent issues for Web².

3.2.1 Security Issues Related to Sensor-Generated Data

One of the most important characteristics of Web² is that information can be generated by sensors. In particular, smart phones may create unique services using 3-axis accelerometers, proximity sensors, and ambient light sensors. They may also provide location information services. In addition, as shown in the case of the sport kit combined with Nike, smart phones provide unique services that enable users to access their exercise information and history on the web, along with other sensors. One problem with these sensor-based services is that they can be vulnerable in terms of security. For instance, a small USB receptor module can be produced in order to detect a Nike+iPhone (or iPod) sensor UID. The Nike+iPhone serial communication tool will provide functions such as simultaneous traffic logging for two serial ports, receiver initializations, rink command transmissions, and operation time logging with sensors in operation; thus, when many traffic logs overlap, the protocol between the iPhone and the receiver, as well as the data exchanged, can be accessed by illegitimate users. In addition, embedded modules that log and track of Nike+iPhone sensors can be developed using Intel's iMote2; malicious persons will be able to collect real-time location information from GPS sensors through the embedded modules. In addition, it will become possible to monitor the sensor data transmitted from Nike+iPhone sensors utilizing Linux-based Gumstix and the existing iPod. In this case, it will be possible to track many people with sensors through the above mentioned equipment. Sensor-generated security issues include methods of attacking the above-mentioned smart phones. For instance, with the information disclosure type of attack, users' jogging routes and daily schedules can be stolen and, if similar sensors are systematically used by businesses later, the businesses may infringe customers' privacy by tracking customer information when customers visit stores for the first time. In addition, batteries may be exhausted through continued communications with sensors. Moreover, cross-platform-type attacks that infect computers or induce the deterioration of their performance when users check their personal exercise information will be also become possible.

3.2.2 Security Issues Related to User-Generated Data

Security issues related to user-generated include "social engineering hacking," which is an attack method by which an individual uses people's vulnerable points to get them to act in accordance with the attacker's intention so that he or she can obtain information. As social networks such as Twitter, mini home pages, blogs, and messengers have attracted huge numbers of users, this hacking method take advantage of the fact that hacking using social characteristics is easier than technical methods. As a result, damages such as the spread of malignant codes and phishing are rapidly increasing. The attacker collects information related to not only to family relations but also to friends, workplace lives, or social meetings. Attackers disguise themselves as friends or acquaintance in order to form trust when approaching the subject of attack. Later, when it is judged that trust has been formed, the attackers execute an attack based on the information collected. The subject of attack, that is, the victim, fails to understand the seriousness of the request of the attacker due to compensation or a sense of moral obligation and comes to accept and execute the request of the attacker [21]. Since open-type OS-based smart phones store a lot of confidential and personal information that is pursued by attackers, such as certification keys related to various

kinds of personal information transmitted through SMS, as well as e-payment information, if the information is leaked, the impact of the damage will be serious. The figure on the right side of Fig. 2 shows a case in some countries where the attacker sent SMS to victims and asked them to pay money to protect their information; as a result, many iPhone users suffered financial harm. As such, it is expected that hacking attempts targeting domestic smart phone users will frequently occur; since serious infringements on privacy may occur due to the hacking of malignant attackers, countermeasures are required.

4 Security Assessment Model for Web² Mobile Environments

To be prepared for security issues in Web² environments, the types of security issues and their potential threats should first be assessed. For instance, Kim and Kang (2009) divided the attacks that could be made in mobile environments into six types: wireless attacks, overcharging attacks, viruses and worms, break-in attacks, DoS (Denial of Services) attacks, and loss or theft [22]. Furthermore, Jang (2010) divided such threats into four types: threats of mobile malignant codes, attacks on the vulnerable points of mobile applications, attacks on mobile platforms, and access to networks without considering security [2]. In addition, Guo et al. (2004) divided the types of threats into attacks on smart phones and attacks through smart phones; they divided attacks on smart phones into attacks through the Internet, infections in synchronization with PCs, and attacks or infections from other smart phones through Bluetooth or UWB [23]. Since the security of smart phones is achieved through organic relations that involve the responsibilities of developers, distributors, and users, it is destined to be affected by mobile OS, applications, mobile platforms, and networks. Since this study is assessing the new threats to security that are expected to appear in Web², the threats will be divided according to the characteristics of Web² for review.

4.1 Threats to Sensor-Generated Data

It is expected that information installed in smart phones that is generated by many sensors, such as 3-axis accelerometers, proximity sensors, and ambient light sensors, will mainly be vulnerable to threats through mobile OS and mobile applications as follows.

- **Threats through mobile OS** use the characteristics of OS. Whereas the iPhone and Blackberry have closed-type platforms, the Symbian, Android, and Windows mobiles have open type platforms. Functions that block untrusted information are limited or lacking; in particular, iPhones show serious security problems due to Jail Break. Although the security of iPhones is considered better than those of other OS, Jail Break phones are vulnerable to attacks by hackers and can be remotely controlled by others. These are called zombie phones. In addition, the user will not only suffer monetary damage, but his/her privacy will also be infringed on because his/her current location can be grasped through information generated by sensors such as GPS.
- **Threats through mobile applications** can be divided into two types. The first inserts viruses, worms, malignant codes, etc. into applications downloaded by smart

phone users from the App Store. These interfere with the smart phone users' control over information or seize personal information. This type may falsify files internal to smart phones or exhaust batteries. The second type attacks the vulnerable points of mobile applications and applications that have not been carefully verified in relation to security have many vulnerable points. Abusing this, the attackers cause problems such as illegally accessing networks or obtaining the authority to control mobile devices [2].

4.2 Threats Involving User-Generated Data

Threats involving user-generated data are security issues arising from human relations in connection with the provision of SNS among the characteristic services of Web². It is considered that the services will be vulnerable mainly to threats made through access to mobile platforms and networks, as follows.

- **Threats through mobile platforms** exist in the App Store. In the case of Google's Android Market, separate processes to examine the registrations of applications have not been put in place; thus, it is difficult to block the registration of illegal applications and, in the case of iPhone, due to the closed OS operation, some users refuse the operation. In case the phones go through Jail Breaking, which refers to arbitrary device transformations or modifications, applications that have not gone through security checks can be easily downloaded so that the phones will have serious vulnerable points. Serious problems may result such as the rapid reduction of the life of batteries, controlling the devices remotely, and the leakage of personal information.
- **Threats through access to networks** occur in the representative networks used by smart phones such as mobile communication company networks (3G), WiFi, and Bluetooth. Recently, with the development of wireless interfaces such as Bluetooth and wireless LAN, free access to networks is guaranteed anytime and anywhere. However, due to careless management, cases of malignant uses such as leaking or monitoring personal information by abusing social closeness occur frequently. Through smart phones, which have limited functions compared to PCs, networks can be contaminated with viruses or malignant codes. The risk that smart phones will be exposed to attacks such as packet sniffing and phishing will be larger in future. Based on the issues mentioned so far, the routes and contents of damage are organized by the type of security issue in Table 2. As reviewed so far, it is expected that threats to sensor-generated data will mainly arise through the characteristics of smart phones, and threats to user-generated data by the users participating in the relevant network. In Web 1.0, although there were no sensor-based services, there were system-centered services provided that involved many vulnerable points, mainly in OS and applications. In Web 2.0, since participation and openness-centered services were provided, in addition to the existing problems in mobile OS and applications, security problems in terms of access to mobile platforms and networks were larger. Since systems have become more sophisticated though the use sensors, and because the active role of participants is emphasized, it is considered that for Web², there will be problems at higher levels than those in Web 1.0 and 2.0; furthermore, sensor- and user-based issues should also be emphasized. Therefore, as shown in Fig. 2, all security issues will be taken into account and a framework to develop security assessment models for Web² will be presented.

Table 2. Web² mobile security threats

Type of threat	Threatened route	Result of threat
Sensor Generated Threat	<ul style="list-style-type: none"> Identify theft program Permission system Signature authentication system Jail Break 	<ul style="list-style-type: none"> Remote controls Acquisition of authority to control mobile devices
	<ul style="list-style-type: none"> Virus worm, applications downloaded with malignant codes Application attack of vulnerable points 	<ul style="list-style-type: none"> Reduction in the life of batteries Zombification of contaminated smart phones
User Generated Threat	<ul style="list-style-type: none"> Malignant access by careless management of wireless interface Network contamination 	<ul style="list-style-type: none"> Illegal access to network Packet sniffing
	<ul style="list-style-type: none"> Illegal remodeling and transformation Application market's weak security 	<ul style="list-style-type: none"> Phishing Spoofing

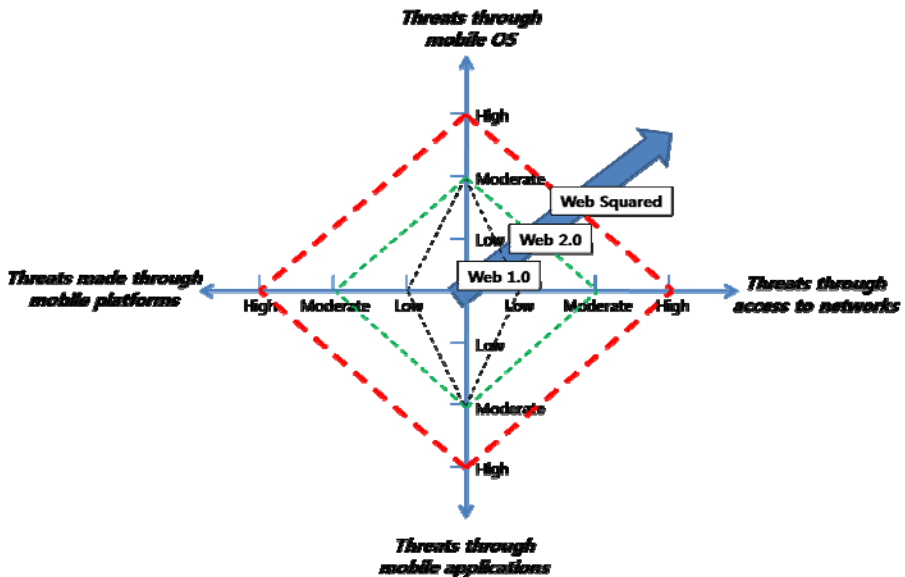


Fig. 2. Security assessment model for web² mobile environment

5 Conclusion and Implications

Unlike security in general information communication technologies, security issues in mobile environments are thought to be relatively less important; thus, research on them has remained at a conceptual level. However, in reality, smart phones are being supplied as a result of the rapid development of mobile environments; thus, it is expected that Web² mobile security issues will be intensively examined when monetary losses occur or when the profitability of new business models in Web² are emphasized. The goal of this study was to prepare for this very issue; its major purpose was to analyze security assessment models for Web² mobiles. To draw the model, along with defining Web² as interactions between the web and the world, the characteristics of services provided in Web² environments were identified as sensor- and user-generated services. Then, the security issues of Web² mobile phones were reviewed based on the characteristics of Web² already delineated. Various types of threats to security related to mobile environments were reviewed by type; these were organized and presented as threats related to sensor- and user-generated data in the context of Web². This review also included the types of security issues, the routes of damage, and the contents of damage. As an outcome of the study, a security assessment model for Web² mobiles was presented. It was argued that threats related to sensor-generated data mainly involved mobile OS and mobile applications; threats related to user-generated data mainly occurred through mobile platforms and access to networks. In this framework, it is judged that, in Web 1.0, importance was attached to systems and sensors; in Web 2.0, the threats to security in Web 1.0 remained, while threats involving user-generated data appeared as new vulnerable points. It is expected that, for Web² mobiles, all these levels of threats to security will be included, while threats related to sensor- and user-generated data will become larger. This study has implications for businesses and related researchers that provide diverse services, and can help them to prepare themselves for Web². First, potential threats to security in mobile environments such as threats made through mobile OS, mobile applications, mobile platforms, access to networks were reviewed. Thus, discussions from diverse viewpoints related to smart-phone security are possible, including those of developers, distributors, mobile communication businesses, vaccine businesses, the government, and research institutions. For instance, users should pay attention to autonomous security such as safety rules in relation to the use of smart phones. Mobile communication businesses should develop and apply technologies that would fundamentally block and eradicate malignant codes that are transmitted as SMS. Manufacturers should apply security technologies when sending/receiving financial or personal information, and should support the encoding of personal information stored in terminals. Mobile platform businesses should establish criteria and systems for the verification of security of the S/W registered in the market and support environments for the development of safe application programs. Vaccine businesses should develop vaccines optimized by considering the storing devices and battery capacities of terminals, which are different from those in general PC environments; they should research and develop vaccines for different OS, which vary with their terminals. The government and research institutions should establish systems of response and cooperation with related organizations; they should also make efforts to develop procedures to respond to smart phone security accidents. In this study, it is suggested that security issues in

Web² cannot be completely solved by simple countermeasures and preventive measures for the devices themselves. For instance, not only threats by sensors, but also threats by users of Web² where participation is emphasized have been presented. Therefore, if threats to mobile security are ignored and attention is concentrated only on the growth of the market and the visible growth of smart phones, technical and political issues that can be solved now may be missed; thus, great social costs may have to be paid later. There is a concept called “techno science” that refers to the idea that revolutions in digital information communications will affect all parts of our daily lives, forming a new culture. This means that new technologies such as smart phones will penetrate into our daily lives and change our cultures accordingly. To establish safe mobile ecosystems for Web² mobiles where the paradigms have changed, we should continuously prepare and develop our mobile cultures.

Acknowledgement

"This research was supported by the KCC(Korea Communications Commission), Korea and KISA(Korea Internet & Security Agency) under the Security System Development for New IT Service program."

References

1. Ji, S.J., Jung, S.Y., Lee, J.H.: Opportunities and Threats of Smart-phone. In: Internet & Security Issues, Korea Information Security Agency, Seoul, Korea (2010)
2. Jang, S.K.: Security Threats in Smart-phone environment. Korea Information Processing Society Review 17, 64–69 (2010)
3. Park, J.H.: Web 2.0 Technical Trend and Web Security Threat Analysis. In: Korea Information Security Agency (2006)
4. Youm, H.Y., Lee, J.S.: Web 2.0 Security Technology Trends and Promoting Standardization. Journal of Telecommunications Technology Association 117, 21–29 (2008)
5. Lee, C.Y., Kim, C.H., Lee, J.H.: Security Problems and Effective Measures of Inspecting Web in Web 2.0. Korea Institute of Information Security and Cryptology 18, 25–33 (2008)
6. O'Reilly, T., Battelle, J.: Web Squared: Web 2.0 Five Years on. O'Reilly Media, Inc., Sebastopol (2009)
7. Kim, S.H., Kim, H.D.: A Study of Web 2.0 Trend & Service View. Research on Digital Policy 5 (2007)
8. Salomon, M.: Would you consider using online virtual worlds for meetings. Telecommunications Journal of Australia 59 (2009)
9. Graham, M.: Transparency and Development: Ethical Consumption and Economic Development through Web 2.0 and the Internet of Things (2010)
10. Wagner, D., Schmalstieg, D.: First Steps Towards Handheld Augmented Reality. In: Proceedings of the 7th IEEE International Symposium on Wearable Computers (2003)
11. Jung, D.Y.: The changes of future that the augmented reality will bring, Samsung Economic Research Institute, Seoul, Korea (2010)
12. Geser, H.: Augmenting things, establishments and human beings (2010)

13. Hong, K.Y., Hong, K.W., Park, J.W., Lee, K.H.: Web Service Security Technology Standardization Trends. Korea Institute of Information Security and Cryptology 14 (2004)
14. Lim, C.G., Ahn, D.S., Kim, K.H., Lee, K.Y.: A Study on the Web Service-Hacking Pattern Recognition System. Webcasting Internet and Telecommunication Review 9 (2009)
15. Lee, W.T., Lee, J.E., Yang, S.C., Hwang, Y.S.: The Changes of Citizens' E-participation and Political Implication in the Age of Convergence between Broadcasting and Telecommunication. Korea Information Society Development Institute (2008)
16. Ritchie, P.: The Security Risks of AJAX/Web 2.0 Application. Network Security 200, 4–8 (2007)
17. Kim, W.J., Moon, Y.J., Lee, S.J.: A Study on the Security Vulnerability and Countermeasure in the Mobile Web 2.0 Environments. Journal of Electrical Engineering and Information Science 34 (2007)
18. Lee, H.D., Park, N.J., Choi, D.H., Chung, K.I.: A Case Study on Mobile USN Technology and Weakness. Korea Institute of Information Security and Cryptology 18 (2008)
19. Korea Information Security Agency, Threats and Countermeasures on Social Network Environment. Information Security Issue Report, Korea Information Security Agency, Seoul, Korea (2007)
20. Lee, H.H., Choi, H.C., Kim, J.H., Cho, S.R., Jin, S.H.: A Study of Sharing Identity and Protect on SNS Environments. Korea Institute of Information Security and Cryptology 19 (2009)
21. Ahn Lab, <http://kr.ahnlab.com/info/securityinfo>
22. Kim, K.Y., Kang, D.H.: Smart-phone Security Solutions in Open-mobile Environment. KIISC Review 19, 21–28 (2009)
23. Guo, C., Wang, H.J., Zhu, W.: Smart-phone Attacks and Defenses. In: Third Workshop on Hot Topics in Networks HotNets-III, San Diego, CA, USA (2004)

Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications

Daniel Mican and Nicolae Tomai

Babes-Bolyai University, Dept. of Business Information Systems,
Str. Theodor Mihali 58-60, 400599, Cluj-Napoca, Romania
{Daniel.Mican,Nicolae.Tomai}@econ.ubbcluj.ro

Abstract. Personalization systems based upon the analysis of users' surfing behavior imply three phases: data collection, pattern discovery and recommendation. Due to the dimension of log files and high processing time, the first two phases are achieved offline, in a batch process. In this article, we propose Wise Recommender System (WRS), an architecture for adaptive web applications. Within this framework, usage data is implicitly obtained by the data collection submodule. This allows for the extraction of usage data, online and in real time, by using a proactive approach. For the pattern discovery, we efficiently used association rule mining among both frequent and infrequent items. This is due to the fact that the pattern discovery module transactionally processes users' sessions and uses incremental storage of rules. Finally, we will show that WRS can be easily implemented within any web application, thanks to the efficient integration of the three phases into an online transactional process.

Keywords: Adaptive web-based applications, Web usage mining, Recommendation systems, Web personalization, Association rules.

1 Introduction

The ability of a web application to offer personalised content and to adapt is determined by its ability to anticipate users' needs and to provide them with the information and content they need. Adaptive web applications [10] can do this only after having analysed data resulted from the users' current and former interaction with the system. Based upon the similarities discovered between different types of content and different user groups, one can make a series of recommendations enhancing the web applications' capacity of adaptation and personalisation. Personalisation systems based upon the analysis of the user's surfing behaviour imply three phases [9]: data collection and preparation, pattern discovery and content recommendation. Thus, a new research branch, called Web Usage Mining came into being, its goal being to discover useful information and knowledge as a result of the analysis of these interactions. The WUM techniques use data extracted from log-files and provide information about activities undertaken by users while surfing. In order to discover new useful information, WUM applies a series of techniques, like classification, clustering, discovery of association rules or sequential patterns [9].

In our research, we have used the technique of association rules, in order to discover correlations between the pages of a web application, based upon the analysis of the user's surfing sessions. Our efforts were channelled towards finding an efficient solution for implementing a recommendation system within a web application capable to synthesize and to store only those data that are relevant for the recommendation process and within which all three phases could be realized online. Thus, we have proposed a new framework to fulfill the high personalisation needs of actual web applications. The Wise Recommender System (WRS) uses a proactive approach, allowing the extraction of data about the users' interaction with the web application. The collecting of usage data is being implicitly achieved, without the need of an explicit request from the part of the users' opinion. Our approach allows the incremental finding and storing, both of the existing connections between frequently visited pages and those less frequently visited. As a result, the WRS is able to offer a list of personalised pages to each user, depending on the pages he is currently surfing on, without the need for a surfing history or a minimal number of visited pages.

2 Adaptive Websites and Web Personalisation Systems

The concept of Adaptive Websites was proposed by Perkowitz and Etzioni in [10]. Adaptive websites are defined as those sites using information about the way in which users access them, in order to improve their organisation and presentation. The Web Personalization System is defined in [9] as any action that adapts information and services provided by a web application to the needs of one user or of a group of users. A personalisation system must be able to provide users with the information they need, without them having to explicitly request it.

During the last years, more and more researchers paid a special attention to WUM domains and to the personalisation of web applications. Among the first researchers who channelled their efforts towards Web Usage Analysis and WUM were Cooley et al. [7], who proposed WebMiner, one of the first systems offering an overview of WUM. The PageGather [10], is a synthesising algorithm that uses clustering in order to find collections of similar pages within a website. The WebPersonalizer [9] has the goal to make recommendations to the users, based upon the similarity of the surfing behaviour with that of users in the past and contains an offline module, whose role is to filter log files data and to extract the most interesting surfing models. We can also mention SUGGEST [2] and Smart-Miner [3] which can efficiently process terabytes of web log files.

In order to be able to use data residing in log files, it is absolutely necessary that these be cleaned and filtered. The analysis of the log files raises a series of problems, namely: the existence of a high number of irrelevant records for the process of web usage mining, the difficulty in identifying users and sessions, the lack of information about the content of accessed pages, and the fact that data processing is a batch processing, which takes up time and resources. Literature mentions several methods helping to identify and delimit a user's sessions. The most popular approach is the 30 minutes time limit threshold [4], followed by reference length [7], and maximal forward reference [6]. The multiple drawbacks connected to the users and sessions identification have led to the development of reactive and proactive strategies. Reactive

strategies want to associate requests with users, based upon web server logs, following their interaction with the website. On the other hand, proactive strategies want to associate requests with users, during their interaction with the website [11].

The goal of the association rules mining [1], [5] is to discover correlations or relations of association between existing records in a dataset. In [5] fundamental association rules have been mentioned, from their emergence and up to the present moment. These works present both classic algorithms like: Apriori, Eclat, Clique, FP-Growth, a.s.o., as well as the generic optimisations they were provided with. In [8], practical and efficient methods are presented, whose aim is to find association rules in the case of less frequent items.

3 Wise Recommender System (WRS), the Proposed Architecture for Web Personalisation

In order to increase the capacity of adaptation and personalisation of web applications, we have integrated several submodules in an innovative manner. Thus, the collecting submodule allows the extraction of usage data, online and in real time, by using the proactive approach. The extraction of data about the users' surfing behaviour, preferences and activities is implicitly accomplished, without the necessity of explicitly involving these into the collection process. The data extracted in this manner are quality data, complete, noiseless and error-free. Moreover, WRS also takes into consideration the content very rarely or occasionally accessed. In figure 1, one can see the architecture of the WRS system we propose.

The crawler identification submodule allows the identification of search engines from human users. This submodule has access to a table which contains the names and the IPs, respectively the IP intervals that are allotted to the main web crawlers. Its role is to filter web crawlers and to send to the users' identification submodule only the traffic generated by human users.

The goal of the user identification submodule is to identify, within a web application, human users exclusively. In order to achieve this goal, the submodule implemented by us successfully uses the newest web technologies allowing session work and uniquely identifies each user, by the means of the IP address, while also taking into consideration the fact that it could be behind a proxy. Thus, it will associate to each user a unique session ID, valid from the moment the user accesses the application and up to the moment the user will close the web browser.

The content identification submodule must uniquely identify the content accessed by the user. The identified content will be stored in a table, together with the number of hits it had over time. Should a page be accessed several times by a user within a surfing session, its accessing value will be incremented by a unit.

The goal of the session identification submodule is to identify surfing sessions for each user. This is quite simple, due to the fact that these ones are already uniquely identified by the users' identification submodule. Furthermore, all pages the web application generates for a user will be accompanied by their session ID. In order to identify and delimit users' sessions, we based our implementation on the W3C approach, one session being made of the totality of pages accessed by a user, from the moment the user opens the web browser and up to the moment the user closes it.

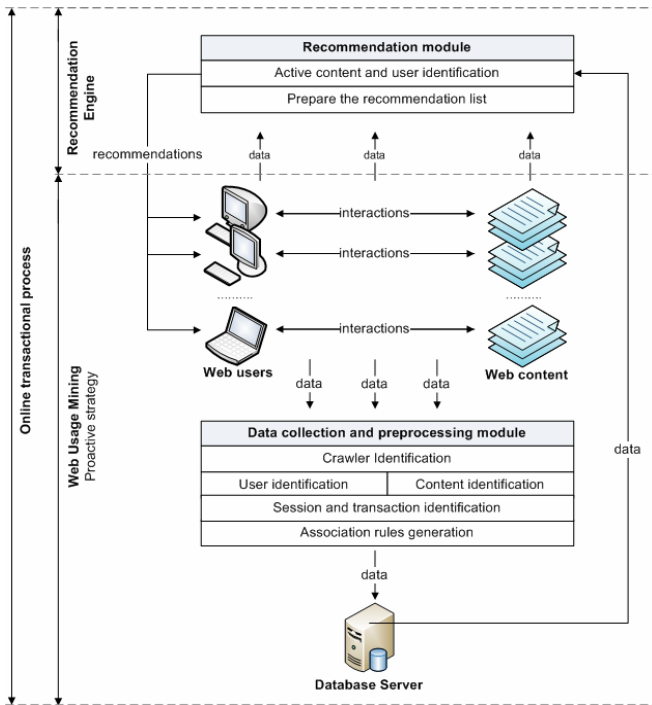


Fig. 1. WRS. The proposed architecture for web recommendation and personalization

The submodule generating association rules can access in real time data taken over by the other submodules. Thus, it will connect to the sessions identification submodule and will take over from this one data related to the users' sessions, in order to generate association rules. The generating of association rules is being done online, in real time, in a transactional process. Once the session has been processed and rules extracted and inserted, it will be deleted from the sessions table. Due to the innovating method of processing sessions and storing of association rules, we succeeded in achieving a scalable model, able to work in real time with a large volume of data.

The active content and user identification is a submodule whose role is to identify the active user and the page this one is visiting. The moment a user accesses a web page, an identification ID is sent, in order to identify the active page of the recommendation list generation submodule. The recommendation list generation receives an identification ID of the current page and has the role of selecting from the database the recommendation list that will contain the pages in a descending order, according to the degree of confidence.

4 Description of Experiments We Have Carried Out

In order to prove the scalability of the proposed model, we have undertaken a series of experiments on two popular Romanian websites: Intelepiciune.ro and BizCar.ro.

The two websites total over 380.000 unique online visitors, respectively over 1,6 million page views per month. We have implemented the proposed model on the two above-mentioned sites and in figure 2 one can notice the dynamics of content, sessions and rules over a period of 32 days in classifieds section.

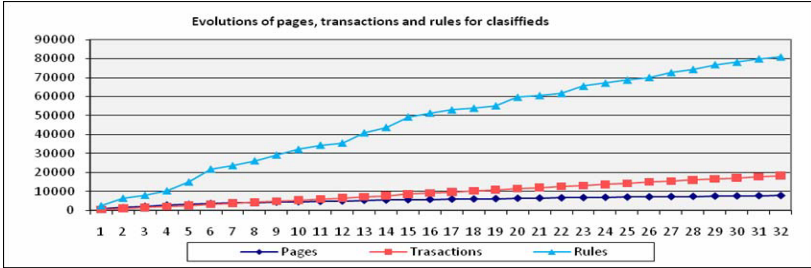


Fig. 2. The evolution of pages, sessions and rules in classifieds section

Unlike other approaches, in the model we are proposing, the recommendation period is not influenced by the number of recorded sessions. In table 1, we can notice the time necessary in order to generate a recommendation list depending on the evolution in time of the number of pages, sessions and rules generated for Intelepciune.ro.

Table 1. The time, number of pages, sessions and generated rules

Page No.	1072	3589	5234	7525	9669	17230	20453	22564
Sessions	667	2687	4922	9723	10799	19952	25637	33628
Rules	3497	18774	33337	53558	63711	99135	116110	142131
Time	0.002	0.008	0.013	0.039	0.052	0.085	0.139	0.159

By looking at the data from the table 1, one can notice that the time necessary to generate a list of recommendations is influenced by two factors: the number of generated pages and rules. As a result, we undertook to analyse the dependence between the recommendation time and the two factors of influence with the help of a regression model. As a result of a statistical analysis, we obtained Multiple R = 0.988572, which shows that there is a strong connection between the two variables, the number of generated pages and rules. We obtained R Square = 0.977274, which shows that 97% from the variation of the recommendation time is explained by the two variables. The average square variation (Standard Error) = 0.009048, the result being that the points on the regression are approaching a straight. Due to the fact that in the case of the number of pages, the P-value is $0.00399 < 0.05$, the result is that this coefficient is of significance. For the generated rules of the P-value = $0.884357 > 0.05$, the result translates in the fact that the coefficient is insignificant. From here, we can conclude that this variable can be eliminated from the model, thus resulting a simple linear regression model.

5 Conclusions

One of the most important goal of an adaptive web application is the content recommendation in a period of time as short as possible. In this article, we proposed WRS for content recommendation and we used association rules in order to model existing connections between the pages of a web application. The proposed system brings an additional benefit, because it allows the finding and maintaining in the system of the rules existing between those pages that are not frequently accessed by users, too. In this article, we proposed a different approach for a recommendation system, by integrating the pattern discovery phase and that of data collection and filtering into a single module. Following the undertaken experiments, it resulted that the proposed model is very efficient in recommending the content and can be easily implemented within any web application. In the future, we wish to continue the optimisation of the recommendation system by incorporating different particularities resulting from the type of content existing in different web applications. Likewise, we would like to exploit knowledge extracted over a longer period of time, in order to see the evolution of the intensity of connections discovered over time.

Acknowledgement. This work is supported by the Romanian Authority for Scientific Research under project IDEI_2596.

References

- [1] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C, pp. 207–216 (1993)
- [2] Baraglia, R., Silvestri, F.: Dynamic personalization of web sites without user intervention. *ACM Commun.* 50(2), 63–67 (2007)
- [3] Bayir, M.A., Toroslu, I.H., Cosar, A., Fidan, G.: Smart Miner: a new framework for mining large scale web usage data. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 161–170. ACM, New York (2009)
- [4] Catledge, L.D., Pitkow, J.E.: Characterizing browsing strategies in the World-Wide Web. *Comput. Netw. ISDN Syst.* 27(6), 1065–1073 (1995)
- [5] Ceglar, A., Roddick, J.F.: Association mining. *ACM Comput. Surv.* 38(2) (2006)
- [6] Chen, M.S., Park, J.S., Yu, P.S.: Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 209–221 (1998)
- [7] Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining World Wide Web browsing patterns. *Knowledge Information Systems* 1(1), 5–32 (1999)
- [8] Ding, J., Yau, S.S.: TCOM, an innovative data structure for mining association rules among infrequent items. *Comput. Math. Appl.* 57(2), 290–301 (2009)
- [9] Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on Web usage mining. *ACM Commun.* 43(8), 142–151 (2000)
- [10] Perkowski, M., Etzioni, O.: Adaptive sites: Automatically learning from user access patterns. In: Proc. of the Sixth International WWW Conference, Santa Clara, CA (1997)
- [11] Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. *INFORMS J. on Computing* 15(2), 171–190 (2003)

Quality in Use Model for Web Portals (QiUWeP)

Mayte Herrera, M^a Ángeles Moraga, Ismael Caballero, and Coral Calero

Alarcos Research Group, Department of Technologies and Information Systems,
University of Castilla – La Mancha,
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
mayteherreranoa@gmail.com,
{MariaAngeles.Moraga, Ismael.Caballero, Coral.Calero}@uclm.es

Abstract. Web Portals are increasingly being used for important tasks, at a work, personal or leisure related issues. Since Web Portals provide with applications, services and information, their levels of quality are an important prerequisite for their success, and hence, a way to ensure that they do not disappear as a consequence of their not having been used. According to the ISO, there are various perspectives of quality: internal, external, and in use. Although issues related to external quality and internal quality have been widely studied in literature, we consider that it is also necessary to take account the importance of quality in use of web portals. This paper is aimed at defining a quality model to assess the level of quality in use of Web Portals. The model is founded on the ISO/IEC 25010 standard, and on some related works found in literature.

Keywords: web portal, quality in use, quality model, quality characteristics.

1 Introduction

Web Portals appeared in the late 1990s as a new type of Internet website architecture specifically designed to provide personalized online services [1]. The success of Web portals depends on their ability to provide accurate content and useful services specifically tailored to individual users according to their requirements [1]. Web portals offer a single point of access to a wide range of information, applications and services in a single environment [2], thus enabling and facilitating the collaboration and interaction between users [3] in such a way that all the services could be adapted to the preferences or necessities of each user [4]. Since web portals allow a suitable work environment to be created for any organization [5], companies establish and promote their web portals to complement, replace or expand their services to their customers, and may even have the intention of providing new services to new customers.

Several researchers have highlighted the importance of developing Web portals which could successfully attract new users and maintain existing ones by serving as a gateway to information and internet services [6]. One important success factor is, therefore, the need to warranty the levels of quality of the web portals as software products [7]. It is thus no longer sufficient to simply provide technically excellent web portals, but it is also necessary for them to fit the best practices and activities of their consumers [8].

As with any other kind of software product, ISO defines several perspectives with which to analyze the level of quality of a web portal: internal and external quality, which are related to the characteristics or properties of the portal, and quality in use, which is aimed at evaluating users' opinions of the portal.

Unfortunately, the relevant literature has not dealt with quality in use in web portals in sufficient depth, despite its having been proved to be an important aspect [1]. In the domain of the web portals specifically, if users do not feel safe when using a portal, if it is difficult for them to achieve their goals through the Web portal, or if they are not satisfied when using it, they may easily decide to use another different one and the web portal could consequently disappear. In the context of organizational web portals, workers are the final users. If the web portal does not fit their specific tasks, it could be very difficult for them to achieve their goals. Assessing the quality in use will allow web portal owners to estimate how usable a web portal might be and the user's satisfaction.

To assess quality in use, it is first necessary to define a model. The main contribution of this paper is, therefore, to define a quality in use model for web portals taking the ISO/IEC 25010 as a basis. ISO/IEC 25010 is the new standard of software product quality that is awaiting publication, and is a part of the new series of SQuaRE (Software product Quality Requirements and Evaluation) standards [9].

This paper is structured as follows. Related literature is presented and analyzed in Section 2. This section is divided into two subsections, the first of which is related to the perspectives of quality in SQuaRE, and is specifically focused on the quality in use model, and the characteristics and sub-characteristics proposed in the standard [9]. The second subsection analyzes certain works that study quality in use in web portals. In section 3 a discussion about the topics and the proposal of a model for quality in use for web portals based on ISO/IEC 25010 is presented. Finally, Section 4 shows some of our conclusions and proposed future work.

2 Related Literature

This section reviews certain concepts related to the perspective of quality in software. More specifically, we shall focus on the perspective of quality in use provided by the ISO/IEC 25010 standard. We shall also analyze how the concept of quality in use in the context of web portals has been studied in literature, along with the set of characteristics that affect quality in the use of portals.

2.1 Perspective of Quality in SQuaRE

The ISO/IEC 9126 standard [10] for software product quality has recently been superseded by the new ISO/IEC 25000 series: the Software product Quality Requirements and Evaluation (SQuaRE) set of standards [9]. According to both standards, the quality of a system can be assessed as the extent to which the system satisfies the stated and implied needs of its various stakeholders. These stated and implied needs are represented in the SQuaRE by means of different models: the software product quality model, the data quality model and the system quality in use model.

The main purpose of the software product quality model is to specify and assess the level of quality of a product through internal measures of inherent properties of the software, and through external measures of the behavior of the system of which the software is part [9].

External quality can be assessed as the result of the combination of the behavior of both the software and the computer system [8], whereas quality in use is the extent to which a product which is being used by specific users meets their needs to achieve specific goals with effectiveness, efficiency, flexibility, safety and satisfaction in specific contexts of use [9]. Quality in use therefore corresponds to users' views of the quality of a system containing software. In this respect, the perception of quality in use must be measured in terms of the result of using the software, rather than the properties of the software itself [8]. In order to make the assessment process easier, the quality in use model defined in ISO/IEC 25010 [9] defines three main characteristics: **usability, safety and flexibility**.

The characteristic of usability has been defined [9] as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. The users' goals may be pragmatic (to be effective and efficient), and/or hedonic (to achieve stimulation, identification and/or evocation). The pragmatic goals are: acceptable perceived experience of use (pragmatic aspects including efficiency), acceptable perceived results of use (including effectiveness), and acceptable perceived consequences of use (including safety). The hedonic and pragmatic satisfaction in this model is achieved through likability, pleasure, comfort and trust. User performance and satisfaction is determined by qualities including attractiveness, functional suitability and ease of use [9].

The characteristic of safety has been inherited from ISO/IEC 9126-1 [10]. It is defined as the degree of expected impact of harm to people, to businesses, to data, to software, to property or to the environment in the intended contexts of use [11]. Safety could be broadly interpreted as the capability to measure the potentially negative outcomes which could be generated from incomplete or incorrect input [11]. For a consumer of a product, negative business consequences may not only be associated with poor performance, but also, for example, with the lack of pleasurable emotional reactions, or of achievement of other hedonic goals [11]. Safety has the following sub-characteristics: commercial damage, operator health and safety, public health and safety, and environmental harm.

The need to consider the context has been made explicit in the new definition of quality in use in ISO/IEC CD 25010 [11]. This is owing to the importance of considering that a product which is usable in one context of use may not be usable in another context with different users, tasks or environments. A new characteristic of flexibility has therefore been included, with the sub-characteristics of context conformity, context extendibility and accessibility [9]. Flexibility also includes aspects related to the users' ability to modify their means of interaction with other users, and the appearance of the web portals' interface to suit their individual needs and preferences [9].

The quality in use model defined by ISO/IEC 25010 is shown in Fig. 1, and a complete definition of the quality characteristics and sub-characteristics can be found in [9].

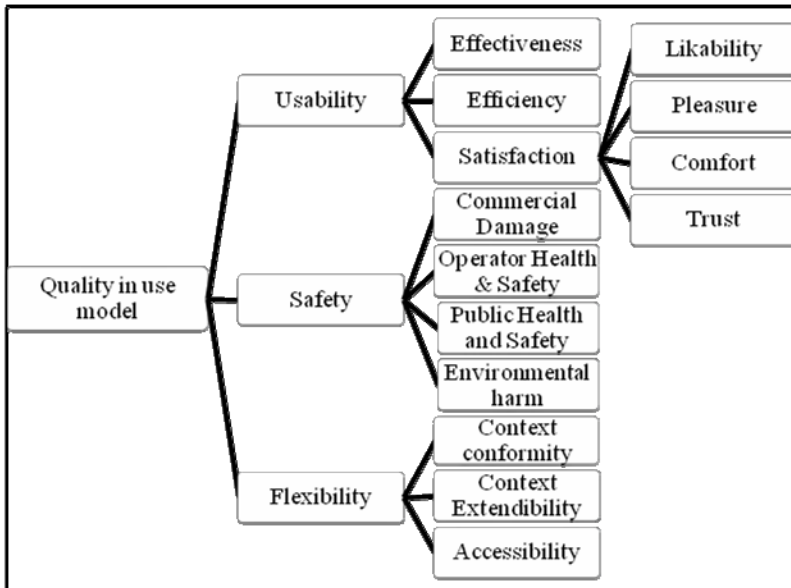


Fig. 1. Model for quality in use in ISO/IEC 25010

2.2 Quality in Use in Web Portals

As was previously mentioned, ISO/IEC 25010 is an evolution of the ISO/IEC 9126 and defines a more complete and detailed quality in use model. Nevertheless, some characteristics regarding quality in use have already been studied in previous works and will be analyzed in this section. The main aim of this section is to study how the characteristics and sub-characteristics for quality in use have been defined in recent studies. This will be done in order to adapt the quality in use model defined in ISO/IEC 25010 [9] to the context of Web portals and to complete the model proposed in this study. It is important to emphasize the fact that the quality characteristics to be analyzed are those concerning the quality in use and those that are of interest to the end users of web portals. These characteristics directly evaluate the effects on users as a result of using the portal, and the effectiveness and efficiency in carrying out their tasks using it. With this in mind, it is not the interest to this study, to analyze aspects or characteristics of the web portals themselves, as these are features of external quality.

The characteristic of Satisfaction has been considered as an element of quality for web applications in general, and has already been introduced in the area of web engineering [12]. The work presented in [6] specifically studied user satisfaction with regard to the usage of web portals. This study outlined the initial results of how a multi-criteria evaluation framework was applied to synthesize the multiple dimensions related to users' satisfaction with a web portal. In their work, the authors studied user satisfaction from the point of view of [6]:

- the *web portal content*, in terms of contained information, or when accessing external information resources.

- the *web portal design*, in terms of providing users with a pleasant, usable and stable environment,
- the *web portal personalization capabilities*, in terms of serving users' specific preferences and needs,
- the *web portal support for the formulation of virtual communities of users*, in terms of bringing together users with similar interests and needs.

WebQual[13] is an instrument for assessing characteristics such as usability, information quality, and service interaction quality. The current version of WebQual (4.0) has been renamed as E-Qual. This version is based on the analysis of the results of its previous versions complemented by an extensive discussion of the literature that supports each dimension. The main change is an increased emphasis on usability rather than site quality. The focus of attention is on user experience when using a Web Portal rather than site characteristics per se [2]. The dimensions included in E-Qual are:

- *usability*, which encompasses aspects such as navigation, appearance and general ease of use,
- *information quality*, which encompasses accuracy, timeliness, relevance, granularity and general believability of the information,
- *service interaction quality*, which encompasses service quality constructs such as security, trust, personalization and access to the organization.

Works of [1, 2, 7, 14, 15] analyzes the perceived quality of service in web portals. It is necessary to point out that this term refers to portal users' perceptions of the quality of all the information, applications and services supplied. These studies point out that ensuring that these services meet quality requirements is essential to ensure business operations and user satisfaction [2].

The works presented in [14] and in [15], both based on E-Qual, study the perceived portal quality. The factors that were confirmed in these studies are similar to those of the previous studies on which they are based, with slight differences in terminology and emphasis that will be shown as follows. It is important to specify that in some studies the term "factor" is used with the same meaning as that of "*quality characteristics*".

In [15] the authors carry out an empirical study of organizational portals. This proposal identifies four factors of service quality:

- *empathy*, which is similar to service interaction quality in E-Qual,
- *ease of use*, which is similar to usability in E-Qual,
- *information quality*, which maintains the same name as that used in E-Qual,
- *accessibility*, which is similar to aspects of usability and service interaction quality in E-Qual.

Similarly, in[14] a study of business-to-consumer portals in Greece identified three factors for service quality in this type of portals:

- *customer care and risk reduction benefit*, which is similar to service interaction quality in E-Qual,
- *information benefit*, which is similar to information quality in E-Qual,
- *Interaction facilitation benefit*, which is similar to usability in E-Qual, but extended by adding considerations of technical design and speed.

In [1] the authors have made a study of the perceived quality of service in a University Web portal. The authors initially based their study on E-Qual, and then added some other factors that they considered might be significant for web portal users. They therefore propose to extend their understanding of user perceived e-service quality to include *transaction quality*, and they include some additional items hypothesized from the literature. Transaction quality includes aspects of complete transactions that are useful for the portal users.

Finally, it is important to consider the study of [16]. In this work a revised list of characteristics was developed into an Internet based questionnaire for measuring user satisfaction with web sites. Questions were placed in different groups. These groups or factors were [16]:

- *Ease of use*, which is related to the ability to easily navigate through a site and find required information. Important aspects include simple, intuitive and consistent navigation,
- *Experience*, which includes the visual and personal experience of visiting the site. Issues include design, use of colors and style, along with building interest and a sense of community,
- *Information*, which is aimed at accessing content with an adequate levels of quality information. This information is appropriate for consumption by the user, and should typically be easy to read and understand, relevant, current, reliable and provided via an appropriate level of detail and format,
- *Communication and integration*, which are related to the way in which the site is integrated with the external environment and communication with the user. This includes being able to find and return to a site, integration or links with other sites, the speed and security of communication, and provision for feedback and other contact.

3 Proposal for a Quality in Use Model for Web Portals (QiUWeP)

This section provides a definition of a model for quality in use for web portals. In order to achieve this goal, and as a starting point, the quality characteristics and sub-characteristics defined in the quality in use model defined by the ISO/IEC 25010 standard [9] have been analyzed and selected.

The standard defines usability, safety and flexibility. Since all of these characteristics affect the use of web portals by final users, they were adapted to the web portal context. Some sub-characteristics defined in the standard, which will be specified later, were adapted to the contexts of web portals. However, other sub-characteristics were not included because they could be considered as not being sufficiently relevant for web portal usage. The obtained model was modified as a result of the analysis of the related existing works shown in the previous section. What follows is an analysis of the characteristics and sub-characteristics found in the quality in use model of the ISO/IEC 25010 standard and in the proposals found in literature.

The definition of the characteristic of **usability** was taken from the standard [9] and adapted to the domain of web portals. Usability has the sub-characteristics of “effectiveness”, “efficiency” and “satisfaction” as reported in the standard [9]. The effectiveness sub-characteristic is defined in terms of accuracy and completeness of the

achievement of the users' goals. The efficiency sub-characteristic is defined in terms of the resources used by portal users as they carry out their tasks by using the portal. With regard to the satisfaction sub-characteristic, the works found that study important aspects for web portal users' satisfaction were [1, 2, 6, 12, 15]. These works have been used as the basis to define the sub-characteristics of satisfaction. As a result of this analysis of the related literature, the sub-characteristics of satisfaction which were added to the model were: the ease with which users can perform their tasks through the portal, the users' visual and personal experiences, the perceived interaction quality, the perceived transaction quality, and finally the sense of being in community while using a portal. These sub-characteristics will be analyzed in detail as follows.

The easier it is to use a portal, the more satisfied the users will be. This signifies that the portal should be easy to navigate, easy to learn how to operate, and it should be easy to find information/resources that users' need. "Experience" is a sub-characteristic that includes the user's visual and personal experience as a result of using a web portal. This sub-characteristic includes the following aspects: the effect of the design of the web portal on the users' satisfaction, along with the perception of acceptable response times, the sense of being in control while doing their tasks, and the attractiveness and enjoyability for users. The "quality of the interactions perceived" includes aspects related to portal users' interactions with the organization through its web portal. Some important aspects of the interaction quality are the ease of communication with the organization through its portal, and the ease of managing and integrating the roles and relationships that users have with the organization. The "perceived transaction quality" sub-characteristic has been supported by some previous studies to include aspects that assess the users' satisfaction with a web portal as a result of completing all the transactions or operations they expect and need to achieve their goal. Another sub-characteristic of satisfaction is "sense of community", which can be defined in terms of user satisfaction with regard to the feeling of being part of a group of users with similar interests and needs.

The definition of **safety** characteristic was borrowed from the standard [9] but re-named as security. From the perspective of final users of web portals, the sub-characteristics defined for security are, therefore, "personal security risk" and "economic damage risk". These new sub-characteristics are being proposed for the first time in this paper, and they are part of the contribution of this work. They were identified as a result of analyzing the web portal context, and the definition of the safety characteristic and its sub-characteristics in the ISO/IEC 25010 [9] standard. The safety sub-characteristics proposed by the standard that were the basis of the new sub-characteristics defined in our proposal were: "Commercial damage" and, "Public health and safety". Commercial damage is defined in the standard as the degree of expected impact of harm to commercial property, operations or reputation in the intended contexts of use. This could include costs of correcting erroneous output, inability to provide an acceptable service that could affect users directly, or loss of current or future sales. Public health and safety are defined according to the ISO/IEC 25010 standard [9] as being the degree of expected impact of harm to the public in the intended contexts of use. The complete definition of the security sub-characteristics will be specified in the proposal for the model below.

Flexibility, when referring to web portals, this indicates whether the portal can be used by a large number of users with different preferences or cultures, or even different

levels of disabilities. Flexibility must therefore be taken into account in the context of portals, and we therefore decided to adapt it and to add it to our proposal. “Accessibility” and “personalization” have been identified as being the sub-characteristics of flexibility. Accessibility was proposed and defined by the standard [9] as the degree of usability for users with specified disabilities. This definition was used in reference to the ISO 9241-171 definition: “the usability of a product, service, environment or facility by people with the widest range of capabilities” [17]. The personalization sub-characteristic was not defined as a sub-characteristic in the ISO/IEC 25010 standard, but is clearly mentioned in the context of use of final users, and is also supported by previous studies.

One important aspect that deserves to be mentioned is the importance of data and information quality in a web portal environment. As we have seen in the related literature, data and information quality are important prerequisites for overall quality in use [18]. People who use data originating from portals in, for example, their work need to be sure that the data has an adequate level of quality. Although the importance of data and information quality has been proven, they will not be included in this proposal owing to the fact that SQuARE clearly differentiates between the quality of a product and the quality of the data manipulated by the software product. Moreover, the authors of [19] have defined a data quality model for web portals, denominated as SPDQM (SQuARE Portal Data Quality Model), which is the reference that can be used to assess the level of data and information quality in a web portal.

Table 1 shows the characteristics and sub-characteristics defined in this proposal as a result of the aforementioned analysis. This table gathers together the related studies that support the identification of each of these characteristics and sub-characteristics that are part of the quality in use model for web portals.

Table 1. Characteristics and sub-characteristics supported by related literature

Quality in use for web portals characteristics and subcharacteristics									
QiUWeP		[9]	[6]	[13]	[14]	[15]	[1]	[16]	
Usability	Effectiveness	x							
	Efficiency	x							
	Satisfaction	Ease of Use			x		x		x
		Experience							x
		Perceived Interaction Quality			x	x	x		x
		Perceived Transaction Quality						x	
Sense of Community			x					x	
Security	Personal Security Risk	x			x				
	Economic damage risk	x			x				
Flexibility	Accessibility	x				x			
	Personalization		x						

The quality in use model for web portals proposed in this study are specified below, where all the definitions of the characteristics and sub-characteristics of the model have been defined. The quality in use model for web portals is shown in Fig. 2.

1. Usability: The extent to which a web portal can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
 - 1.1. Effectiveness: The degree to which the web portal can be used by the users to achieve their specific goals with accuracy and completeness in a specified context of use.
 - 1.2. Efficiency: The degree of resources consumed by web portal users in relation to the effectiveness reached in a specific context of use.
 - 1.3. Satisfaction: the web portal users' degree of satisfaction with regard to achieving their pragmatic and hedonic goals in a given context of use. These goals are related to likability, trust and pleasure.
 - 1.3.1. Ease of Use: The web portal users' degree of satisfaction in relation to being able to use the portal easily to achieve their goals. Important aspects include simple, intuitive and consistent navigation.
 - 1.3.2. Experience: The web portal users' degree of satisfaction in relation to the visual and personal experience of visiting the portal. Issues include design and style, enjoyability or entertainment, along with acceptable response times for users/clients.
 - 1.3.3. Perceived Interaction Quality: The web portal users' degree of satisfaction with regard to the experience of interacting with the organization via its web portal.
 - 1.3.4. Perceived Transaction Quality: The web portal users' degree of satisfaction with regard to being able to carry out and complete all the operations that they wish to through the web portal.
 - 1.3.5. Sense of Community: The web portal users' degree of satisfaction in relation to meeting, collaborating and communicating with other users with similar interests and needs.
2. Security: The degree, to which the web portal does not, under specified conditions, lead to a state in which the personal security of its users is endangered and economic damage is caused.
 - 2.1. Personal Security Risk: the degree of expected impact of harm to the personal security of the portal's users or clients in the intended contexts of use.
 - 2.2. Economic damage risk: the degree of expected impact of causing economic damage to the web portal users owing to insecure operations in the intended contexts of use.
3. Flexibility: the degree to which the quality in use requirements for web portals can be achieved in different contexts of use and for as many users as possible. Flexibility can be achieved by adapting the web portal to additional user groups or cultures. Flexibility enables portals to consider circumstances, opportunities and individual preferences.
 - 3.1. Accessibility: the degree of effectiveness, efficiency, safety and satisfaction, when people with the widest range of capabilities use the web portal.
 - 3.2. Personalization: the degree to which the users can modify certain aspects of the portal to suit their specific preferences and needs.

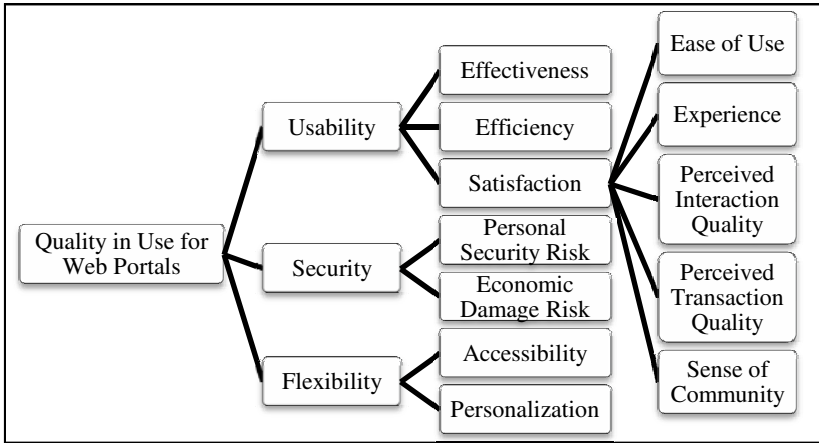


Fig. 2. Quality in use model for Web Portals

4 Conclusions and Future Work

As the contents of this paper show, the quality of web portals is worth observing and should be evaluated taking into account the different perspectives of quality. In this respect, the quality in use is the perception of the quality provided by final users, and the results of using the portal. Although some characteristics of quality in use have been studied for the context of web portals in the literature analyzed, no quality in use model is aligned with the ISO/IEC 25010 standard. In this proposal we have defined a quality in use model for web portals, based on the ISO/IEC 25010, the new standard for software product quality and quality in use. This approach takes into account the portal users' perspective and therefore aims to assess the results of use of the web portal, both from the perspective of user satisfaction, and the effectiveness and efficiency of the users while they use the web portal. Our proposal can be used to assess the quality perceived when using a web portal and achieving user's goals. This proposal takes as basis the ISO/IEC 25010 standard [9], since this is an evolution of ISO/IEC 9126, along with certain related previous studies which were taken into account in order to define the characteristics and sub-characteristics of the model that are needed to assess quality in use in the context of web portals. As part of this work we are currently defining measures of effectiveness, efficiency, security and flexibility. For these measurements, we are working on a questionnaire to assess the satisfaction of the users of a web portal. As part of our future work, the appropriateness of this model will be validated by applying it to assess the quality in use of various web portals in different context of use. This will be made using techniques of empirical validations as surveys and the analysis of the results of these evaluations will make possible refine the model if it were necessary.

Acknowledgments

This research is part of the projects PEGASO-MAGO (TIN2009-13718-C02-01), and DQNet (TIN2008-04951-E/TIN), both supported by the Spanish Ministerio de Educación y Ciencia, and TALES (HITO-2009-14), both supported by the Consejería de Educación y Ciencia of Junta de Comunidades de Castilla-La Mancha.

References

1. Tate, M., et al.: Stakeholder Expectations of Service Quality in a University Web Portal, p. 1–21 (2009)
2. Tate, M., Evermann, J., Hope, B., Barnes, S.: Perceived Service Quality in a University Web Portal: Revising the E-Qual Instrument (2007)
3. Aneja, A., Brooksby, B., Rowan, C.: Corporate portal framework for transforming content chaos on intranets. *Intel Technology Journal* 11, 21–28 (2001)
4. Ángeles Moraga, M., Calero, C., Piattini, M.: A First Proposal of a Portal Quality Model. In: IADIS International Conference (2004)
5. Collins, H.: Corporate Portal Definitions and Features. Amacom Books, New York (2001)
6. Manouselis, N., Sampson, D.G.: Multiple Dimensions of User Satisfaction as Quality Criteria for Web Portals. In: ICWI 2004, pp. 535–542. IADIS (2004)
7. Liu, C.-T., Du, T.C., Tsai, H.-H.: A study of the service quality of general portals. *Information & Management* 46(1), 52–56 (2009)
8. Bevan, N.: Quality in use: Meeting user needs for quality. *Journal of Systems and Software* 49, 89–96 (1999)
9. ISO/IEC CD 25010.3: Systems and software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Software product quality and system quality in use models. ISO (2009)
10. ISO/IEC 9126-1: Software engineering – Product quality - Part 1: Quality model. ISO (2001)
11. Bevan, N.: Extending Quality in Use to Provide a Framework for Usability Measurement, pp. 13–22 (2009)
12. Zhang, P., Small, R.V., Dran, G.: Websites that Satisfy Users: A Theoretical Framework for Web User Interface Design and Evaluation. In: Proceedings of 32nd IEEE International Conference on System Sciences, Hawaii, USA (1999)
13. WebQual home page, <http://www.webqual.co.uk/>
14. Gounaris, S., Dimitriadis, S.: Assessing service quality on the web: Evidence from business to consumer portals. *The Journal of Services Marketing* 17, 529 (2003)
15. Kuo, T., Lu, I.-Y., Huang, C., et al.: Measuring user's perceived portal service quality: An empirical study. *Total Quality Management and Business Excellence* 16, 309 (2005)
16. Barnes, S., Vidgen, R.: WebQual: An exploration of Web-site Quality. In: Proceedings of the Eighth European Conference on Information Systems, Vienna (2000)
17. ISO 9241-171: Ergonomics of human-system interaction – Part 171: Guidance on software accessibility. ISO (2008)
18. Wynn, M., Zhang, S.: Web Portals in SMEs - Two Case Studies. In: Proc. of the 2008 Third International Conference on Internet and Web Applications and Service (ICIW). IEEE Computer Society, Los Alamitos (2008)
19. Carmen Moraga, M.M., Calero, C., Caro, A.: SQuaRE-Aligned Data Quality Model for Web Portals. In: Ninth International Conference on Quality Software, pp. 117–122 (2009)

Towards Support Processes for Web Projects

Pablo Becker and Luis Olsina

GIDIS_Web, Engineering School, UNLPam
Calle 9 y 110, (6360) Gral. Pico, La Pampa, Argentina
{beckerp, olsinal}@ing.unlpam.edu.ar

Abstract. Measurement, evaluation and analysis are support processes to main engineering processes. In this work we present an integrated strategy whose rationale is supported by a well-defined measurement and evaluation process, a conceptual framework that relies on an ontological base, and quality evaluation methods. Particularly, we discuss some process views for specifying project context, nonfunctional requirements, measurement, evaluation and analysis. Finally, the benefits of having well-established measurement and evaluation processes as quality drivers for web projects are highlighted.

Keywords: Process, Measurement, Evaluation, Quality, C-INCAMI, WebQEM.

1 Introduction

Quality, scope, time and cost are the main interdependent variables for managing software/web development projects. For each engineered project, and independently of the development lifecycle adopted, levels of quality for its entities and attributes should be agreed, specified, measured and evaluated for analyzing and improving them. To assure repeatability and consistency of results, it is necessary to have well-defined measurement and evaluation (M&E) support processes with clearly established activities, sequences, inputs and outputs, roles, etc.

Given the inherent complexity of the process domain, a process can be modeled taking into account different views [4], namely: i) *Functional*, which includes the activities' structure, inputs and outputs, etc.; ii) *Informational* that includes the structure and interrelationships among artifacts produced or consumed by the activities; iii) *Behavioral*, which models the dynamic view of processes; and, iv) *Organizational* that deals with agents, roles and responsibilities. For specifying all the views, different modeling languages can be used. Each language has its own strengths and weaknesses that can make it more suitable for modeling certain views than others. In this work we focus on functional and behavioral perspectives of the M&E process, using UML activity diagrams and the SPEM Profile [17]. Activity diagram is a well-known standard widespread both in academic and industry, easy to understand, and also fits to our modeling needs for both views. We are planning to cover the integration of views using the SPEM approach.

Besides establishing a set of activities and procedures for specifying, collecting, storing, and using metrics (for measurement) and indicators (for evaluation), we argue that more robust analysis and decision-making process can be achieved if the following

three aspects of an evaluation strategy are considered at once [15]: i) a *M&E process*; ii) a *conceptual framework* that relies on an ontological base; and, iii) *evaluation methods and tools* instantiated from both the framework and process.

Firstly, to assure repeatability and reproducibility for M&E activities and then consistent results, it is necessary to provide a well-defined process (model), which prescribes or informs a set of activities, their inputs and outputs, roles, interdependencies, and so forth. Secondly, a well-established conceptual framework should be built upon a robust base, which explicitly and formally specifies the main agreed concepts, properties, relationships, and constraints for the M&E domain. Lastly, web quality evaluation methods and tools appropriately instantiated from both the process descriptions and framework components may be derived.

In previous works, we have built, based on a metrics and indicators ontology [13], a framework called C-INCAMI (*Contextual-Information Need, Concept model, Attribute, Metric and Indicator*) [15], and the WebQEM method (*Web Quality Evaluation Methodology*) [14] and its associated tool. This methodology was used in different case studies [14, 16]. Nevertheless, the M&E process was not explicitly and formally specified as we currently do it in the present work.

The modeling of these processes contributes to: i) facilitate the understanding and communication among stakeholders; ii) integrate and formalize different activities that are interrelated as nonfunctional requirements specification, context specification, M&E design, analysis and recommendation; iii) reuse the ontological base stemming from the C-INCAMI framework easing understandability and interoperability; iv) allow the instantiation of methods and tools; and v) foster the integration with other processes such as testing and quality assurance.

The rest of this paper is organized as follows. Section 2 presents related work on M&E processes. Section 3 gives an overview of the C-INCAMI framework. Section 4 discusses the functional view and to lesser extent the behavioral view of these processes, illustrating them by examples. Details of the C-INCAMI tool are given in Section 5 and, finally, the main remarks and future directions are outlined.

2 Related Work and Discussion

There exists an ongoing SQuaRE (*Software product Quality Requirements and Evaluation*) project that proposes harmonizing many ISO standards related to quality models, M&E processes, etc. The ISO 25000 [10] document provides guidelines for the use of the new series of standards; however, the documents aimed at specifying M&E processes are not issued yet. So the standards for software measurement process (ISO 15939 [9]), and the process for evaluators (ISO 14598-5 [8]) are still in force. In [9] two activities out of four are considered to be the core measurement process, namely: *Plan the measurement process*, and *Perform the measurement process*. In [8] five activities are specified, namely: *Establishment of evaluation requirements*, *Specification of the evaluation*, *Design of the evaluation*, *Execution of the evaluation plan*, and *Conclusion of the evaluation*. In fact, we can observe that there is so far no unique ISO standard that specifies in an integrated way the M&E process and approach as a whole. Moreover, in [13] we have shown that very often there is some lack of consensus about the used terminology among the quoted ISO standards as well as some missing terms, mainly for the evaluation domain.

The CMMI (*Capability Maturity Model Integration*) [3] *de facto* standard is also worthy to mention in that it provides support process areas such as *Measurement and Analysis*, among others. It aligns information needs and measurement objectives with a focus on goal-oriented measurement –following to some extent the GQM (*Goal Question Metric*) [1] approach and the [9] measurement process. It specifies (specific/generic) practices to achieve the given process area goals. But, a process model itself is not defined; rather represents practices (i.e. actions/activities) without explicitly establishing sequences, parallelisms, control points, etc. Some specific practices for *Measurement and Analysis* are for example: *Establish measurement objectives*, *Specify measures*, *Collect measurement data*, and *Analyze measurement data*. However, a clear distinction between measurement and evaluation processes is missing in addition to lacking a robust conceptual base.

On the other hand, for the metrics and indicators domain there are several proposals regarding conceptual frameworks [6, 7, 11]. Unlike [7, 11], our conceptual framework (C-INCAMI) has an ontological root [13]. This ontology considered sources as ISO standards, articles and books, following the WebQEM terminology as well. Our ontology has similarities to the one presented in [5] and then slightly refined by some coauthors in [6], but in [15] we have modeled some terms (e.g., elementary indicator, global indicator, etc.) and relationships (e.g., measurement and measure, metric and indicator, among others) that differ semantically with those proposed in [6]. Also we have added new concepts, such as context, context properties, etc., not considered in the quoted works.

Ultimately, the main contribution of our research with regard to related work is a M&E strategy whose rationale is supported by integrated, well-defined M&E processes, which in turn rely on a M&E conceptual framework backed up by an ontological base. Next, an overview of our conceptual framework is given.

3 C-INCAMI Components: An Overview

The C-INCAMI framework provides a domain (ontological) model defining all the concepts and relationships needed to design and implement M&E processes. It is an approach in which the requirements specification, M&E design, and analysis of results are designed to satisfy a specific information need in a given context. In C-INCAMI, concepts and relationships are meant to be used along the activities of M&E processes. This way, a common understanding of data and metadata is shared among the organization's projects leading to more consistent analysis and results across projects.

Taking into account the main activities of the process (which are discussed in Section 4), the framework is structured in six components, namely: i) *M&E project definition*, ii) *Nonfunctional requirements specification*, iii) *Context specification*, iv) *Measurement design and implementation*, v) *Evaluation design and implementation*, and vi) *Analysis and recommendation specification*. For space reasons, just the components shown in Fig. 1 are presented below –for more details, see [15].

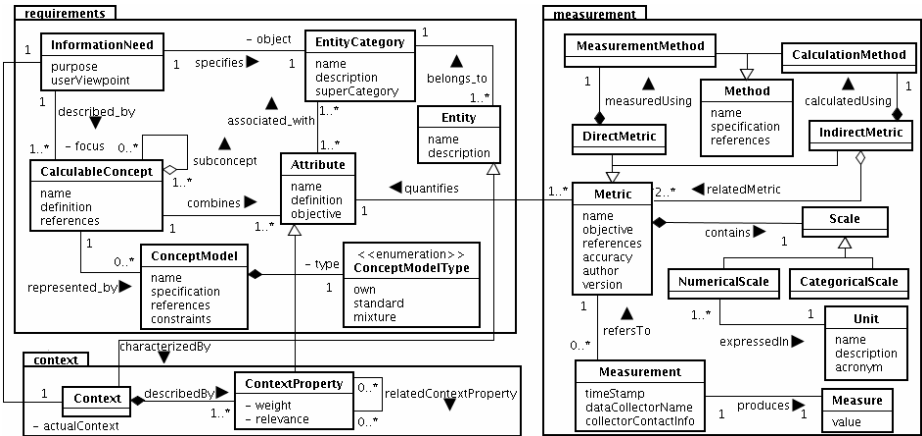


Fig. 1. Some concepts and relationships for the Nonfunctional requirements specification, Context specification, and Measurement design and implementation components

Nonfunctional requirements specification component (*requirements* package in Fig. 1) allows specifying the *Information Need* of a M&E project. The information need identifies the *purpose* and the *user viewpoint*; in turn, it focuses on a *Calculable Concept* and specifies the *Entity Category* to evaluate –e.g. a resource, process, product, etc. A calculable concept can be defined as an abstract relationship between attributes of an entity and a given information need. For instance, external quality, quality in use, etc., are instances of a calculable concept. This can be represented by a *Concept Model* such as ISO software quality models. The leaves of an instantiated model (e.g. a requirement tree) are *Attributes* associated with an *Entity*.

In the *Context specification* component one concept is *Context*, which represents the relevant state of the situation of the entity to be assessed with regard to the information need. We consider *Context* as a special kind of *Entity* in which related relevant entities are involved. Consequently, the context can be quantified through its related entities that may be resources –as a network infrastructure, a working team, life-cycle processes-, the organization or the project itself, among others. To describe the context, attributes of the relevant entities are used –which are also *Attributes* called *Context Properties*. All context properties' metadata may be stored in an organizational repository so for each M&E project the particular metadata and its values are stored as well.

The *Measurement design and implementation* component allows specifying the metrics that quantify attributes. To design a *Metric* the *Measurement* and *Calculation Method* and the *Scale* should be defined. Whereas a measurement method is applied to a *Direct Metric*, a calculation method is applied to an *Indirect Metric*. A *Measurement* produces a *Measure* as shown in Fig. 1.

4 Modeling the Process for M&E

Process modeling is not a simple job. At the beginning of process modeling usually engineers think about what a process must do rather than how activities descriptions

should be performed. So we can say that a process prescribes (or informs) a set of phases and activities, inputs and outputs, among other concerns, in order to foster repeatability and reproducibility. Taking into account the C-INCAMI framework and to some extent the ISO standards for M&E, our integrated process proposal embraces the following main (core) activities: i) *Define Non-Functional Requirements*; ii) *Design the Measurement*; iii) *Design the Evaluation*; iv) *Implement the Measurement*; v) *Implement the Evaluation*; and vi) *Analyze and Recommend*.

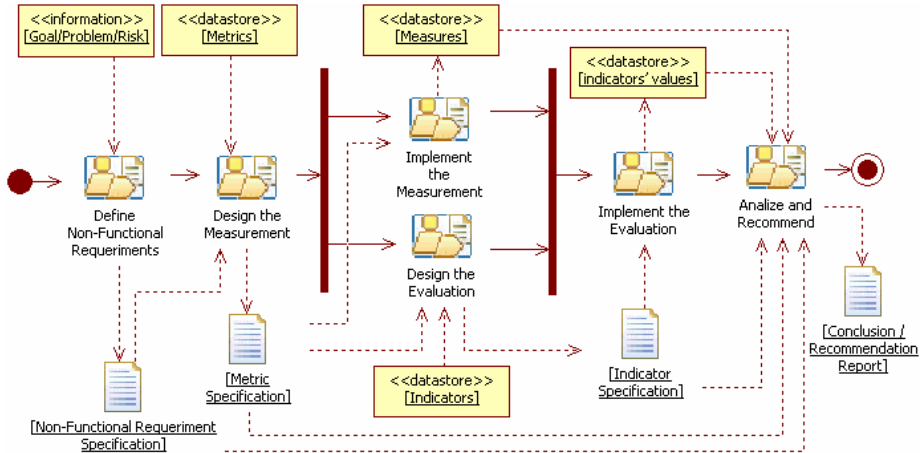


Fig. 2. Overview of the Measurement and Evaluation Process

These activities as well as sequences, parallelisms, inputs and outputs are shown in Fig. 2 using a terminological correspondence between the conceptual framework and process. The `<<datastore>>` stereotype represents repositories, e.g., the *Metrics* repository stores the metadata for the designed metrics.

The proposed *M&E* process (initially introduced in [2]) follows a goal-oriented approach. First, the *Define Non-Functional Requirements* activity has a specific goal or problem as input and a nonfunctional specification document as output. Then, the *Design the Measurement* activity allows identifying the metrics from the *Metrics* repository to quantify attributes: the output is a metric specification document. Once the measurement was designed, the evaluation design and the measurement implementation activities can be performed. The *Design the Evaluation* activity allows identifying indicators to be used. The *Implement the Measurement* activity uses the specified metrics to obtain the measures, which are stored in the *Measures* repository. Next, the *Implement the Evaluation* activity can be carried out. Finally, *Analyze and Recommend* activity has as inputs the values (i.e., data) of measures and indicators, the requirements specification document, and the associated metrics and indicators specifications (i.e., metadata).

In the next subsections, we illustrate the above six activities using excerpts of the case study developed in [16]. The case study dealt with the *Cuspide.com* shopping cart with the purpose of improving the external quality applying evaluation and WMR (*Web Model Refactoring*) in a systematic way.

4.1 Define Non-functional Requirements

Once the requirement project has been created, *Define Non-Functional Requirements* is the first core activity to be performed as shown in Fig. 2. It consists of: *Establish the Information Need*, *Specify the Context*, and *Select a Concept Model* sub-activities as shown in Fig. 3 and described below.

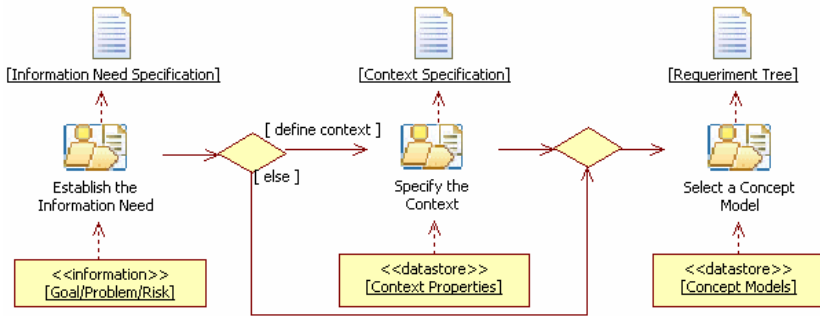


Fig. 3. Sub-activities for the *Define Non-Functional Requirements* activity

The *Establish the Information Need* activity is aimed at indicating the purpose, the user viewpoint, the focus, etc. of the M&E project. Thus, we first *Define the purpose* (e.g. “improve” to our example); then *Define the user viewpoint* (e.g. “final user”), and *Establish the object* (e.g. “web application” as entity category). The next sub-activity, *Establish Entity*, is aimed at identifying one or more concrete entities (e.g. “Cuspide.com shopping cart”), which belong to the selected entity category; however, the definition of the concrete entity can be deferred until measurement implementation. Note that it is possible to identify more than one entity for comparison purposes. *Identify the focus* is the final sub-activity in order to *Establish the Information Need* activity; e.g. the higher-level focus (*CalculableConcept*) is “external quality”, while “usability” and “content quality” are the sub-concepts.

Once the information need specification document is yielded (Fig. 3), we optionally can *Specify the Context* as shown in Fig. 4. It involves to *Select relevant Context Properties* –from the organizational repository of context properties [12] - and, for each selected property the *Quantify Context Property* sub-activity must be performed based on the associated metric. In the end, we get as output a context specification document for a specific project. As example, the following relevant context properties are selected for the “refactoring project”: “lifecycle type” whose value (based on the associated metric) is “Agile Methodology”; and “technique type” whose value is “WMR”. Note that these context properties, i.e. the name, definition, value, etc. should be recorded with the M&E project for further comparisons.

According to Fig. 3 *Select a Concept Model* is the last sub-activity to define non-functional requirements. It involves both *Select a Model* and *Edit the Model* sub-activities. Concept models (e.g. the ISO 9126 internal and external quality models) are chosen from an organizational repository regarding the quality focus; in our example the external quality focus was considered. If the selected model is not totally suitable, e.g. some sub-characteristics or attributes are missing, it is necessary to *Edit*

the Model, adding or removing sub-concepts and/or attributes accordingly. Table 1 shows, in the first column, an edited quality model (a requirement tree) for our study. Ultimately, the resulting artifact from the *Define Non-Functional Requirements* activity is the nonfunctional requirement specification document, which contains all the above mentioned information.

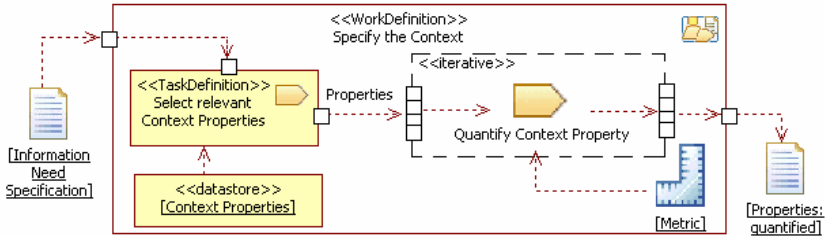


Fig. 4. Activities to Specify the Context

4.2 Design the Measurement and Evaluation

The *Design the Measurement* activity (recall Fig. 2) should be performed to identify suitable metrics regarding the nonfunctional requirement specification document. It consists of *Establish Entity* (still optional) and *Assign one Metric to each Attribute* activities (Fig. 5). It is important to remark that this activity does not mean designing each metric itself but selecting it from an organizational repository; however, if a needed metric is not there, then it should be designed, agreed by experts, and stored.

To *Assign one Metric to each Attribute*, first, we have to *Identify a Metric* that quantifies the attribute. The selected metric can be either direct or indirect. If the metric is indirect it is necessary to *Identify related Metrics* and *Identify Attributes quantified by related Metrics*. These two sub-activities allow identifying the extra attributes and metrics so that data collector may later gather data accordingly. Lately, if the measurement or calculation method of the metric can be automated the *Select a Tool* activity should be performed. Note in Fig. 5 that all these activities should be repeated for each attribute of the requirement tree as indicated by the `<<iterative>>` stereotype.

Regarding the case study, the attributes presented in table 1 are inputs to perform the *Assign one Metric to each Attribute* activity. For example, by performing the *Identify a Metric* sub-activity for the “Line item information completeness” attribute (defined as “the extent to which the line item information coverage is the sufficient amount of information to an intended user goal and task”), the direct metric named “Degree of completeness to the line item information” was assigned.

The selected metric scale specifies three categories considering an ordinal scale type, namely: (0) *Incomplete*, it has less information than category 1 or no information at all; (1) *Partially complete*, it only has title, price, quantity, and sometimes availability fields; and (2) *Complete*, it has title, author, price, quantity, added on date, and availability. The measurement method is objective and the data collection was made observationally. Because the selected metric is a direct metric, the *Identify related Metrics* and *Identify Attributes quantified by related Metrics* sub-activities are not necessary to be performed. As final output of these activities (Fig. 5) we get the metrics specification document.

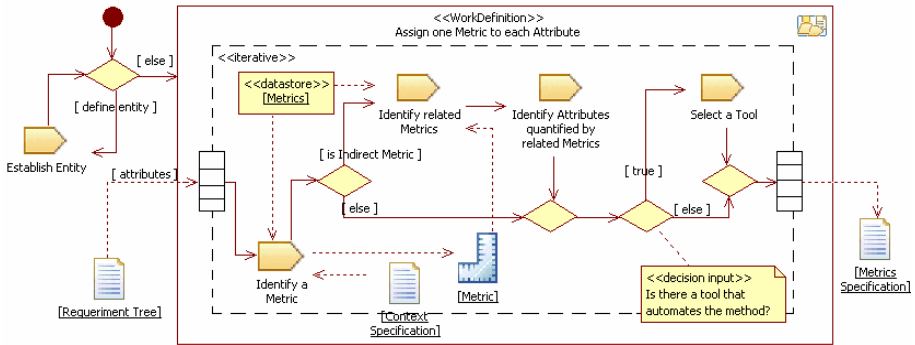


Fig. 5. The *Design Measurement* activity

On the other hand, the value of a particular metric does not necessarily represent the satisfaction level achieved by its associated attribute (that is an elementary non-functional requirement). Thus, we need to define a new mapping that will produce an elementary indicator value representing the achieved satisfaction level. Also we need to define partial and global indicators to evaluate the higher-level concepts of the requirement tree. Therefore we have to perform the *Design the Evaluation* activity as shown in Fig. 2. Particularly, *Design the Evaluation* entails two activities, namely: *Identify Elementary Indicators* and *Identify Partial and Global Indicators*.

The first activity allows specifying an indicator per each attribute (from the requirements tree) using the associated metric specification as input. For this, the involved sub-activities are: *Establish the Elementary Model*, *Establish the Calculation Method* (optional), and *Identify the Scale*. For example, for the “*Line item information completeness*” attribute, the elementary indicator named “*Performance Level of the line item information completeness*” was identified. For the elementary model, the following mapping from the metric specification is used: the measure value 0 maps to 0%, 1 to 50%, and 2 maps to 100%. The decision criteria –in terms of acceptability ranges- were: unsatisfactory (red) for values within 0-40; marginal (yellow) for values within 40-70; and satisfactory (green) for values within 70-100.

Then, we perform the *Identify Partial and Global Indicators* activity for each concept in a similar way as shown above. The final output is an indicators specification document.

4.3 Implement the Measurement and Evaluation

The *Implement the Measurement* activity implies collecting all the measures for all attributes per each entity. In order to do this, the metrics specification document and the tools that automate the measurement and/or calculation methods are inputs. The involved activities are: *Establish Entity* and *Measure Attributes*. The activity *Measure Attributes* should be performed for each entity, recording the values, timestamps, etc. In our study, using the *Degree of completeness to the line item information* metric, the measurement produced the value 1 (see attribute 2.1.1.1 in table 1). All the measures are stored in the *Measures* datastore for further use.

In a nutshell, for implementing the evaluation two activities are considered, viz. *Calculate Elementary Indicators* and *Calculate Partial and Global Indicators*. These activities must be performed for each entity. Looking at the design of the “*Performance Level of the line item information completeness*” indicator and the value 1 for the measure, which are inputs to the *Calculate Elementary Indicators* activity, an indicator value of 50% is yielded (see the third column of the table) as a consequence of applying the above elementary model. Once all the elementary indicators’ values were obtained they serve as input to the *Calculate Partial and Global Indicators* activity. Then, the aggregation model is executed producing partial and global indicators’ values. Some values are shown in the last column of table 1.

Table 1. Excerpt of the external quality model for a shopping cart [16]. Legend: EI stands for Elementary Indicator; P/GI stands for Partial/Global Indicator.

Characteristics and Attributes (<i>in italic</i>)	Measure	EI value	P/GI value
External Quality			61.97%
1 Usability			60.88%
.....			
2 Content Quality			63.05%
2.1 Content Suitability			63.05%
2.1.1 Basic Information Coverage			50%
2.1.1.1 <i>Line item information completeness</i>	1	50%	
2.1.1.2 <i>Product description appropriateness</i>	1	50%	
2.1.2 Coverage of other related Information			76.89%
.....			
<i>Return policy information completeness</i>	...	33%	

4.4 Analyze and Recommend

It is recognized that data, information, and knowledge are valuable assets for better organizational decision making. As such, data and metadata obtained from M&E processes serve as inputs for analysis, conclusion and recommendation processes.

First, taking into account the business commitment, established information need, context, metrics and indicators specifications as well as the data properties with regard to the scale, the *Design the Analysis* activity (figure not shown) is performed. This activity involves deciding on the allowable mathematical and statistical methods and techniques for analysis regarding the scale type, datasets properties, etc., the suitable tools for the kinds of analysis at hand, the presentation and visualization mechanisms, and so forth.

Once the above design decisions have been made, the *Implement the Analysis* activity follows. Basically, it consists of implementing the designed procedures, running the planned tools and storing the results according to the established formats in order to produce the analysis report. Next by performing the *Elaborate the Conclusion Report* activity, this ends up drawing conclusions in the form of a conclusion report. However, decision makers very often need not only a conclusion report but also a recommendation report –which contains for instance strengths and weaknesses, course of actions to tackle the problems identified in the analysis, etc.

For our case study, after looking at the conclusion report we were ready to start refactoring the navigation and presentation models to improve the Cuspide's shopping cart component. For example, one recommendation -stemming from the indicators' acceptability levels- was: changes should be planned in the "*Basic Information Coverage*" sub-characteristic (ranked 50%), mainly in the "*Line item information completeness*" (2.1.1.1), and "*Product description appropriateness*" (2.1.1.2) attributes. A further course of action was performed by applying the *Add Node Attribute*, and *Turn Information into Link* refactoring models, which had impact in both attributes. As result of the improvement action, i.e. by executing the refactoring process, the enhanced shopping cart satisfied totally (100%) these nonfunctional requirements -see details in [16].

5 Technological Support for M&E Projects

With the aim of providing support to the third foundation mentioned in the Section 1, i.e., *evaluation methods and tools*, the above discussed M&E process and framework were instantiated in a web application called C-INCAMI Tool, which in turn gives technological support to the WebQEM methodology. The current tool version¹ fully takes into account the specified M&E process.

C-INCAMI Tool guides evaluators throughout the project definition, non-functional requirements specification, M&E design and implementation, and report generation activities showing results in a user-friendly way. The application presents to evaluators the activities that can be executed at any moment, enforcing the dependencies between them as defined by the process. Also it records the artifacts produced by the activities and inputs them to the required dependent activities. A screenshot of the application is shown in Fig. 6.

The architectural components of the tool are based on the conceptual components defined in the C-INCAMI framework (as shown in Section 3). These are responsible for implementing the business logic regarding the creation and manipulation of the *Information Need*, *Concept Model*, *Metric*, *Indicator*, among other concepts. Also, the tool allows creating and managing M&E projects (called *MEProject* in C-INCAMI framework [15]). Each *MEProject* contains a concrete *Requirement Project*; from this, one or more *Measurement Projects* can be created; and, in turn, for each measurement project one or more *Evaluation Projects* could be defined and created. So, for each M&E project, subprojects can be managed accordingly. Each subproject is responsible for handling metadata and data to the corresponding conceptual component. This separation of concerns for each *MEProject* facilitates ultimately the traceability and consistency for ulterior intra- and inter-project analysis.

The web application has a multi-tier client-server architecture. For the persistence layer all repositories modeled in the process (recall <<*datastore*>> stereotype) were implemented using semantic web technologies, particularly, the Sesame RDF repository (www.openrdf.org/). In the presentation layer we have also used RIA (*Rich Internet Application*) technologies. Other employed technologies for the framework implementation were JavaTM. Ultimately, all documents are interoperable and usable among different analysis tools because they are specified in XML.

¹ Available at <http://gidisw.ing.unlpam.edu.ar:8080/INCAMI-WS/>

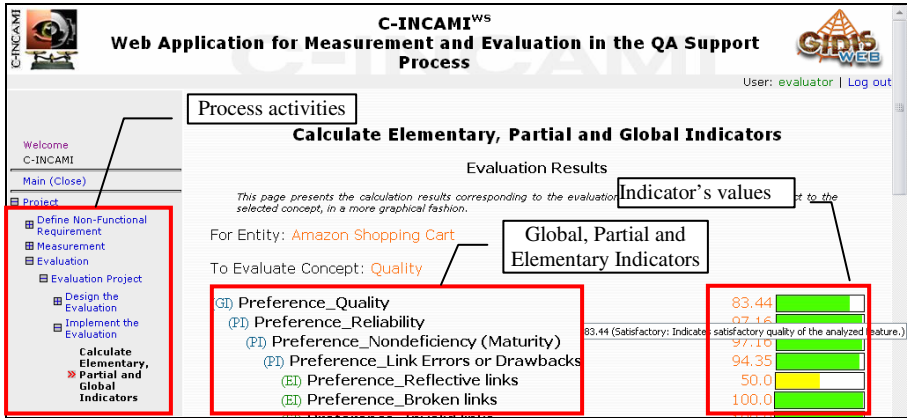


Fig. 6. C-INCAMI Tool screenshot showing results in the Evaluation activity

6 Conclusion and Future Work

Organizations relying only on engineering process areas and not paying enough attention to supporting process areas –as measurement, evaluation and analysis- are more prone to making less justifiable and repeatable projects and, therefore, more prone to fail. So having well-established strategies to cope with M&E software/web projects and programs may be considered key quality drivers as well. This way, we have introduced our strategy based on three foundations viz. the M&E process, the conceptual framework, and evaluation methods/tools. Particularly, we have addressed in this paper the specification of the M&E process stressing in the Introduction Section the main contributions.

The underlying hypothesis of our proposal is that without well-established M&E processes and without appropriate recorded metadata of concrete projects’ information needs, context properties, attributes, metrics and indicators it is difficult to ensure data and datasets are repeatable and comparable among the organization’s projects. As a consequence, analysis, comparisons and recommendations can be made in a less robust or even wrong way.

Finally, a humble contribution of this research with regard to ISO-related work is a M&E approach whose rationale is supported by interrelated and well-defined requirement, measurement, evaluation, analysis and recommendation processes. In addition, the discussed processes rely on a M&E conceptual framework backed up by an ontological base. Also specific methods and tools can be instantiated from both the defined M&E process and the C-INCAMI framework.

A future work is devoted to deal with modeling the remainder views –not just the functional and behavioral views as we have discussed here- and their integration to the C-INCAMI Tool.

Acknowledgements. This work and line of research is supported by the PAE-PICT 2188 project at UNLPam, from the Science and Technology Agency, Argentina.

References

1. Basili, V., Rombach, H.D.: The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* 14(6), 758–773 (1989)
2. Becker, P., Molina, H., Olsina, L.: Integrating Process and Measurement and Evaluation Framework. In: *CIBSE 2009*, Medellin, Colombia, pp. 261–266 (2009) (in Spanish)
3. Capability Maturity Model Integration, Version 1.2 CMMI for Development, CMU/SEI-2006-TR-008 ESC-TR-2006-008, SEI Carnegie-Mellon University (2006)
4. Curtis, B., Kellner, M., Over, J.: Process Modelling. *ACM Comm.* 35(9), 75–90 (1992)
5. García, F., Ruiz, F., Bertoa, M.F., Calero, C., Genero, M., Olsina, L., Martín, M., Quer, C., Condori, N., Abrahao, S., Vallecillo, A., Piattini, M.: An ontology for software measurement. University of Castilla-La Mancha, Spain, TR. UCLM DIAB-04-02-2 (2004)
6. García, F., Bertoa, F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M.: Towards a consistent terminology for software measurement. *Information & Software Technology* 48(8), 631–644 (2006)
7. Goethert, W., Fisher, M.: Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques. *Software Engineering Measurement and Analysis Initiative*, CMU/SEI-2003-TN-024 (2003)
8. ISO/IEC 14598-5. IT - Software product evaluation - Part 5: Process for evaluators (1998)
9. ISO/IEC 15939. Software Engineering - Software Measurement Process (2002)
10. ISO/IEC 25000. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE (2005)
11. Kitchenham, B.A., Hughes, R.T., Linkman, S.G.: Modeling Software Measurement Data. *IEEE Transactions on Software Engineering* 27(9), 788–804 (2001)
12. Molina, H., Olsina, L.: Assessing Web Applications Consistently: A Context Information Approach. In: *IEEE CS, 8th Int'l Congress on Web Engineering*, NY, US, pp. 224–230 (2008)
13. Olsina, L., Martin, M.: Ontology for Software Metrics and Indicators. *Journal of Web Engineering* 2(4), 262–281 (2004)
14. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. *IEEE Multimedia* 9(4), 20–29 (2002)
15. Olsina, L., Papa, F., Molina, H.: How to Measure and Evaluate Web Applications in a Consistent Way. In: Rossi, Pastor, Schwabe, Olsina (eds.) *Springer HCIS book Web Engineering: Modeling and Implementing Web Applications*, pp. 385–420 (2008)
16. Olsina, L., Rossi, G., Garrido, A., Distanto, D., Canfora, G.: Web Applications Refactoring and Evaluation: A Quality-Oriented Improvement Approach. *Journal of Web Engineering* 7(4), 258–280 (2008)
17. SPEM. Software Process Engineering Metamodel Specification. Doc./02-11-14., Ver.1.0 (2002)

Reliability Verification of Search Engines' Hit Counts: How to Select a Reliable Hit Count for a Query

Takuya Funahashi and Hayato Yamana

Computer Science and Engineering Div., Waseda University,
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
{takuya,yamana}@yama.info.waseda.ac.jp

Abstract. In this paper, we investigate the trustworthiness of search engines' hit counts, numbers returned as search result counts. Since many studies adopt search engines' hit counts to estimate the popularity of input queries, the reliability of hit counts is indispensable for archiving trustworthy studies. However, hit counts are unreliable because they change, when a user clicks the "Search" button more than once or clicks the "Next" button on the search results page, or when a user queries the same term on separate days. In this paper, we analyze the characteristics of hit count transition by gathering various types of hit counts over two months by using 10,000 queries. The results of our study show that the hit counts with the largest search offset just before search engines adjust their hit counts are the most reliable. Moreover, hit counts are the most reliable when they are consistent over approximately a week.

Keywords: search engine, information retrieval, hit count, reliability, trustworthiness.

1 Introduction

Nowadays, many studies adopt search engines' hit counts to estimate the popularity of queried terms. Here, search engine hit count is defined as the number returned by a search engine as the search result count. Hit counts are widely used in fields such as machine translation research [1], word similarity measurements [2], and word clustering research [3]. For example, "Honto? Search" system [4] was proposed to help the user to determine trustworthiness of a statement that he or she was unconfident about, such as whether "the Japanese Prime Minister was Junichiro Koizumi" is true or false. The system compares the hit count of a statement with the hit count of its alternative statement to determine it. Thus, unreliable hit counts result in unreliable knowledge extraction. The use of hit counts in such a diverse set of fields claims that they be accurate and reliable.

Although hit counts are used for many studies, hit counts are unreliable because they "dance," i.e., change, in some situations. To the best of our knowledge, hit counts dance when a user clicks the "Search" button multiple times or clicks the "Next" button on the search results page, or when a user queries the same term on separate days. Previous researches have attempted to demonstrate the characteristics of hit count transition [5][6][7]; however, these researches do not clarify the reliability

of the hit counts, i.e., how to select the most reliable hit count for a query from among its variation.

The purpose of this paper is to investigate the trustworthiness of search engines' hit counts to adopt them with various types of studies, while we do not know the real hit counts. First, we demonstrate how hit counts dance by classifying the "hit count dance" into three cases: 1) when we click the "Search" button multiple times in a short time interval, 2) when we click the "Next" button multiple times in a short time interval, and 3) when we click the "Search" button on separate days. Fig. 1 shows these three cases. Second, we propose a new scheme to provide a basis to adopt hit counts on the basis of the observed hit count transition. In the experiment, we verify hit counts from three major search engines, Google, Yahoo!, and Bing, by using their "search APIs." Though we investigate three major search engines, the difference of hit counts among search engines is out of scope of this paper.

The rest of this paper is organized as follows: In Section 2, we review related work. Section 3 describes a process to verify hit counts from the viewpoint of providing a basis to adopt hit counts. Finally, the paper is concluded in Section 4.

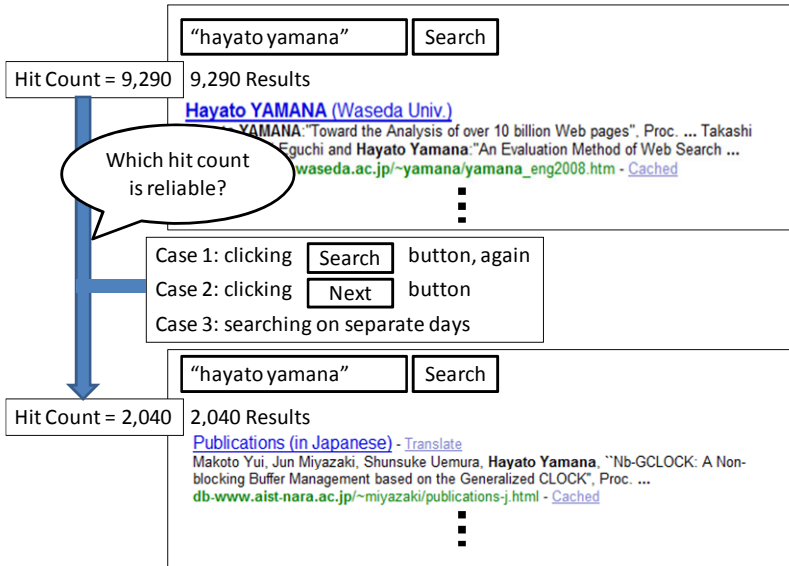


Fig. 1. Hit Count Dance

2 Related Work

Kilgarriff [5] reported that queries repeated the following day gave hit counts that varied by more than 10% as compared to the counts obtained the day before with 9 times in 30 days. He assumed that the reason was because queries were sent to different computers, at different points in the update cycle of their index, and with different data in their cache. He mentioned that the arbitrariness of search engine counts should be taken into account.

Thelwall [6] compared three major search engines with respect to four viewpoints—hit counts, the number of search result URLs, the number of search result sites, and the number of search-result top-level domains obtained by using 2,000 single word queries. Here, the three search engines considered were Google, Yahoo!, and Live Search. He compared these three search engines to clarify the difference among them on the basis of the above-mentioned four viewpoints; however, he was not concerned about the transition of the hit counts day by day, nor concerned about the transition when clicking “Next” button.

Uyar [7] proposed “Percentage of Error,” how differ between the hit count and the number of returned documents, using queries whose hit counts were less than 1,000. Uyar assumed that the actual number of returned documents was reliable. Here, we are able to retrieve 1,000 documents at maximum for each query from search engines. “Percentage of Error” is defined as the difference between a hit count of the first result page and the number of returned documents, normalized by the number of returned documents. Uyar used 1,000 queries that were generated as random numbers with seven and eight digits. Uyar compared Google, Yahoo! and Live Search to conclude that Google provided the most accurate hit counts for both single and multiple term queries. Google provided less than 10% error for 78% of single queries. His main goal was to compare three search engines to find out the search engine returning accurate hit counts; however, he did not concerned about the hit count transition within the search engine. In addition, Uyar investigated the average daily hit count change ratio. However, he was not concerned about the transition of the hit counts day by day. Moreover, these results cannot be directly generalized to queries that return a higher number of documents because his research was based on queries that returned less than 1,000 documents.

As shown above, previous researches showed the hit count dance characteristics to find out the search engine to provide the most accurate hit counts among three major search engines. However, they did not clarify the basis to adopt the accurate hit counts within a search engine, i.e., how to select the most reliable hit count for a query from among the variation of hit counts when clicking the “Search” button multiple times in a short time interval or when clicking the “Next” button multiple times in a short time interval, or when clicking the “Search” button on separate days.

In this paper, we show the characteristics of the hit count dance in more detail in Section 3. Then, we propose a new scheme for selecting accurate hit counts to enhance their reliability.

3 Experiment and Trustworthiness of Hit Count

Our research goal is to find out the most reliable hit count for a query from among the variation of hit counts within a search engine, when clicking the “Search” button multiple times in a short time interval or when clicking the “Next” button multiple times in a short time interval, or when clicking the “Search” button on separate days. Here, the difference of hit counts among search engines is out of scope of this paper.

3.1 Experiment

Three types of experiments were performed on the basis of the three “hit count dance” cases described in Section 1:

- 1) Clicking the “Search” button many times
- 2) Clicking the “Next” button step by step to reach the last search result page
- 3) Searching with the same query on different days

We used 10,000 queries provided by Yahoo! JAPAN as the top 10,000 frequent queries in December 2007. The dataset is provided for the Japanese national “info-plosion” project [8]. Yahoo! JAPAN provides the top 10,000 frequent queries as they are. This implies that the written language of the queries is mainly Japanese but not limited to Japanese. The distribution of the queries' length, number of space-separated words, is shown in Table 1.

Table 1. Distribution of queries' length

Length of query (word)	Frequency
1	9,522
2	424
3	42
4	1
5	1
Total	10,000

In the experiment, the queries were submitted to Google, Yahoo!, and Bing via their search APIs, with these APIs' default setting; non-phrase search, normal-safe filter and the number of search engine returned is 10. Although we submit 10,000 queries to each of these search engines, sometimes, they return errors such as “no results” or “connection timed out.” We omitted such erroneous results from our experimental result. These experiments were performed from October 2009 to December 2009.

Case 1: Clicking the “Search” button many times. To demonstrate the characteristics of hit count transition in case 1, we use the coefficient of variation, called CV. CV is a normalized measure of dispersion as shown below:

$$CV = \frac{\text{standard deviation}}{\text{average}} = \frac{\sqrt{\text{variance}}}{\text{average}} \quad (1)$$

Each query out of the 10,000 queries was submitted to all the search engines, 100 times in 5 min, i.e., each search engine performed the same search 100 times in 5 min. Here, we submitted the same query many times in a short time span to avoid the other effects such as search engines' index update cycle. After gathering 100 variations of hit counts with the same query, CV is calculated both for each query and for each search engine.

Table 2 shows the result of case 1, where the numbers in total are not equal to 10,000 because search engines sometimes return errors as described. As shown in Table 2, Google returns almost the same hit counts, i.e., consistent hit counts. For only 9 queries out of 10,000, the hit count dances with a CV less than 0.1%. Yahoo! also returns consistent hit counts because 99.4 % queries have a CV of less than 0.1%. Additionally, the maximum CV is less than 5%. Bing’s hit counts dances a little as compared to those of the other search engines; however, 97.5% queries have a CV of less than 0.5%. Only one query has a CV of more than 20%; this deviation is related to the “porn words” that resulted in some effects of Bing’s search filter. As a result, we are able to conclude that hit counts rarely dance in case 1. Even when hit counts dance, the CV is less than 5% except Bing’s rare case.

Table 2. Distribution of hit counts when the “Search” button is clicked many times

Range	Frequency		
	Google	Bing	Yahoo!
$CV = 0.0\%$	9,977	699	9,096
$0.0\% < CV \leq 0.1\%$	9	2,555	730
$0.1\% < CV \leq 0.5\%$	0	6,191	46
$0.5\% < CV \leq 1\%$	0	171	4
$1\% < CV \leq 5\%$	0	56	1
$5\% < CV \leq 10\%$	0	12	0
$10\% < CV \leq 20\%$	0	4	0
$20\% < CV \leq 100\%$	0	1	0
$100\% < CV$	0	0	0
Total	9,986	9,689	9,877

Case 2: Clicking the “Next” button continuously to reach the last page of the search results. In order to find out the characteristics of the hit count transition for a given query when clicking the “Next” button continuously, we gathered HitCount(1,10), HitCount(11, 20), ..., and HitCount(991, 1000) for each query. Here, HitCount(N, N+9) is defined as the hit count when we set the number of results per page to 10 and the “search offset” to N, i.e., retrieving from the N-th ranked result. For example, HitCount(1,10) shows the hit count when the top 10 search results are retrieved. HitCount(11,20) shows the hit count when the search results from 11th to 20th are retrieved, i.e., after clicking “Next” button once. HitCount(21,30) shows the hit count after clicking “Next” button twice.

In order to find out the transition patterns of the hit count with clicking the “Next” button, we define “Deep hit count Vector,” in short DV, for each query. DV is defined as below based on HitCount(1, 10), HitCount(11, 20), ..., and HitCount(991, 1000). Each element in DV indicates the difference ratio of the hit count compared to the HitCount(1, 10).

$$DV = \left\{ \frac{HitCount(1,10)}{HitCount(1,10)}, \frac{HitCount(11,20)}{HitCount(1,10)}, \dots, \frac{HitCount(991,1000)}{HitCount(1,10)} \right\} \quad (2)$$

In this experiment, we submitted the queries only to Bing and Yahoo! because their APIs are able to return the top 1,000 documents at maximum as search results, while Google's API returns only the top 64 documents at maximum.

After calculating the set of DVs for 10,000 queries, we apply k-means clustering technique with cosine similarity to all pairs of DVs to extract transition patterns of the hit counts. Here, we omit the DVs that lack some elements because of errors such as "no results" or "connection timed out." We vary its clustering size k from 1 to 6; then, selected the best size by manual. Here, we choose the best size based on the following two points;

1. Start offsets when hit count dances, i.e., changes, begin are clearly different among clusters.
2. Curves of change ratio are clearly different among clusters.

Finally, we chose the best clustering size—1 on Bing and 2 on Yahoo!. Fig. 2 and Fig. 3 show the clustered hit count transition of DVs, each of whose curves represents the mean of the DVs clustered into the same cluster.

On Bing, the best clustering size is 1 where 100% of queries are clustered in. All DVs show the same characteristics such that the hit count is almost consistent till the search offset becomes more than 900. Nevertheless, after a search offset exceeds 900, the hit count falls down to 1.3% compared to $HitCount(1, 10)$. The final hit count, $HitCount(991, 1000)$, is the same number that a user is able to actually retrieve from Bing. This means that Bing adjusts the hit count to be the same as the number a user is able to actually retrieve such as 1,000. Hence, we have to omit the last hit count number on Bing.

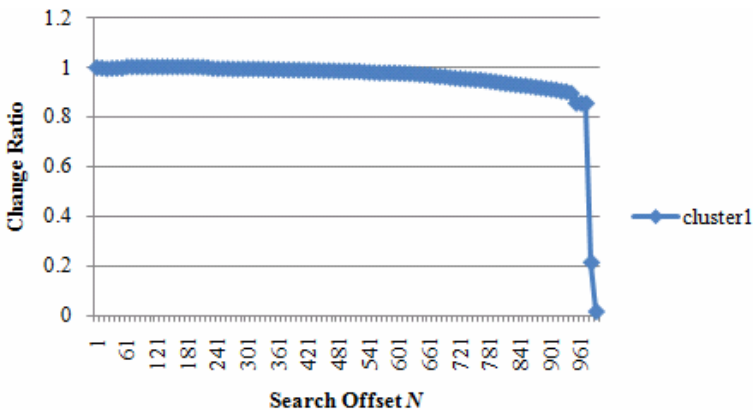


Fig. 2. Clustering result of hit count transition when clicking the "Next" button continuously to reach the last page of the search results (Bing)

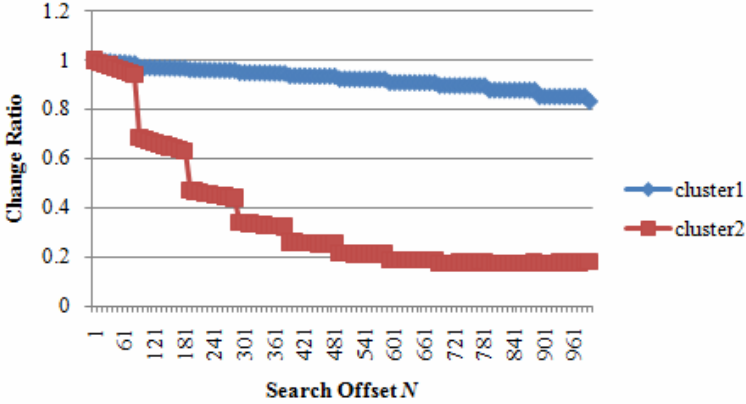


Fig. 3. Clustering result of hit count transition when clicking the “Next” button continuous to reach the last page of the search results (Yahoo!)

On Yahoo!, we have two clusters. Cluster 1 includes 86% of queries whose hit counts decrease slowly. Cluster 2 includes 14% of queries whose hit counts decrease faster than those of cluster 1 and saturate at around 20% of HitCount(1, 10).

From the above observation, we can assume that search engines return roughly estimated hit counts HitCount(1, 10) at first. With an increase in the start offset, hit counts decrease to re-calculate the hit counts by using more matched results as shown in Fig.4.

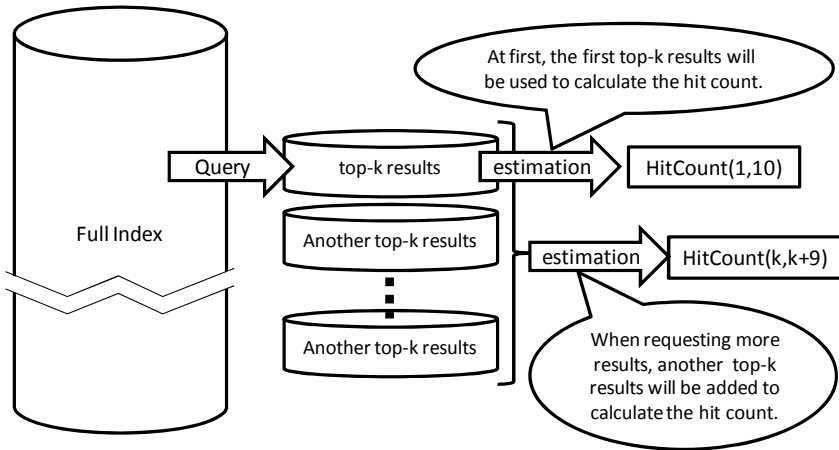


Fig. 4. How to estimate hit counts (assumption)

Here, we should consider many speed-up techniques adopted by search engines such as distributed processing [9], early termination [10], and index pruning [11]. These techniques enable fast searching; however, they might have some side-effects while calculating hit counts. We assume that these side-effects result in a rough

estimation of HitCount(1, 10). From this viewpoint, when we search with a larger search offset, we will be able to get a more reliable hit count except in the case when search engines adjust the hit count to the number that a user actually retrieves. Since hit counts are estimated by using sampled Web pages that are matched with the query through the searching process, hit counts become more reliable when a larger search offset is used. The larger the search offset is, the larger is the number of sampled Web pages that will be used for estimating the hit count. Therefore, we can conclude that the most reliable hit count is HitCount(k, k+9) where k is the largest number, usually 991 for a search having more than 1,000 matched pages. If a search engine adjusts the last hit count, we should use the hit count just before the adjusted hit count.

Case 3: Searching on different days. In order to demonstrate the characteristics of the hit count transition in case 3 and to find out what the reliable hit count during the period is, we gathered hit counts during two months, from October 11th, 2009, to December 12th, 2009, by using the same 10,000 queries. In order to find out the transition patterns of the hit counts when searching on different days, we define “Variational ratio Vector,” in short VV, for each query as shown below. Each element in VV indicates the change ratio of the hit count compared to the hit count on October 11.

$$VV = \left\{ \frac{HitCount(Oct, 11)}{HitCount(Oct, 11)}, \frac{HitCount(Oct, 12)}{HitCount(Oct, 11)}, \dots, \frac{HitCount(Dec, 12)}{HitCount(Oct, 11)} \right\} \tag{3}$$

where HitCount(*date*) is the hit count searched on *date*

where HitCount(*date*) is the hit count searched on *date*

In order to clarify the hit count transition, we apply k-means clustering technique with a cosine similarity to all pairs of VVs. Here, we omit the VVs that lack some elements because of errors such as “no results” or “connection timed out.” We varied its clustering size from 1 to 6 and selected the best size by manual on the same criterion used in case 2. Finally, we chose the best clustering size—4 on Google, 5 on Bing, and 3 on Yahoo!.

Fig. 5–Fig. 7 show the clustering result of VVs, each of whose curves represents the means of VVs clustered into it. Table 3 shows the size of each cluster, i.e., the percentage of queries clustered into the same cluster.

Table 3. Percentage of queries clustered into each cluster

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Google	80.3%	11.0%	7.2%	1.5%	-
Bing	59.1%	19.4%	15.1%	5.6%	0.8%
Yahoo!	67.2%	29.6%	3.2%	-	-

On Google, the largest cluster is cluster 1 whose hit count is almost consistent during the period. The second largest cluster is cluster 2 whose hit count is also consistent but with pulse noise between Oct. 20th and Oct. 22nd. The next largest cluster is cluster 3 whose hit count increases slowly. Finally, the smallest cluster is cluster 4 whose hit count increases eight-fold times around Dec. 2nd.

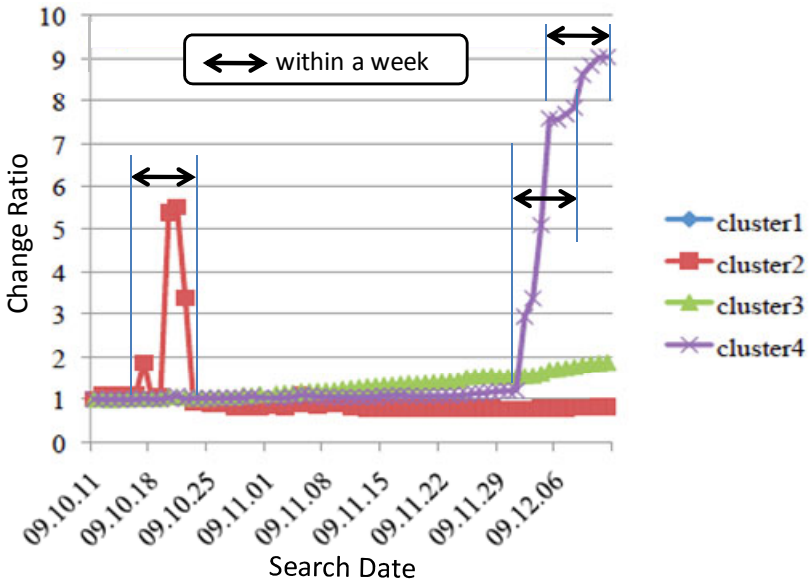


Fig. 5. Clustering result of daily hit count transition (Google)

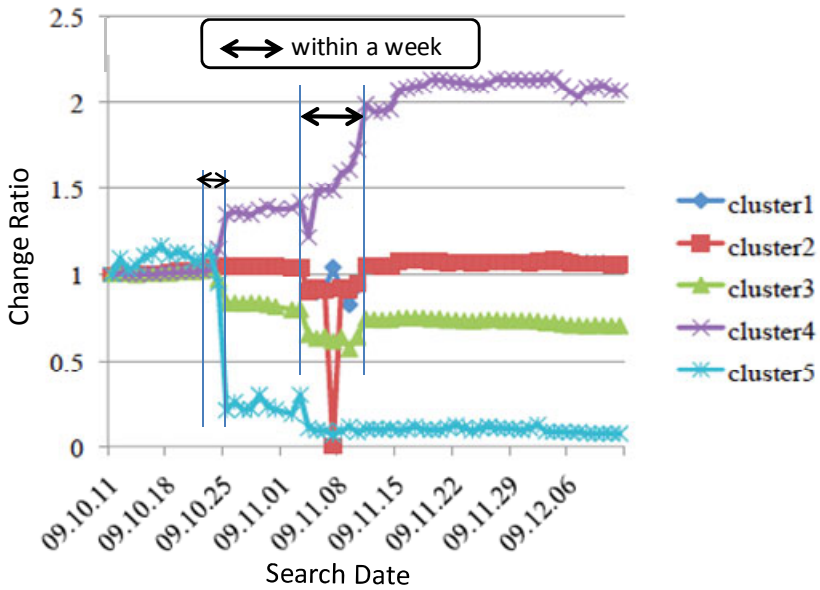


Fig. 6. Clustering result of daily hit count transition (Bing)

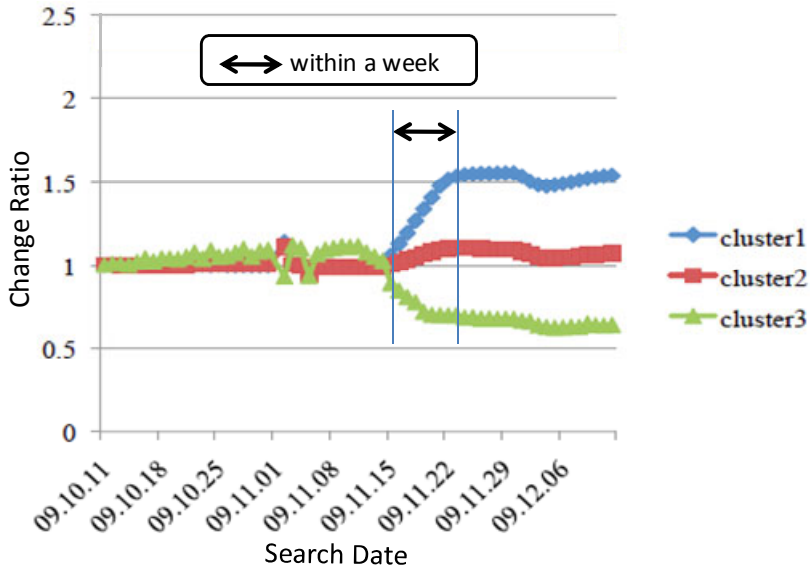


Fig. 7. Clustering result of daily hit count transition (Yahoo!)

On Bing, hit counts vary on Oct. 25th. The phenomenon may be resulted from Bing's search index update on that day. From Nov. 3rd to Nov. 11th, the hit counts dance very hard. Finally, hit counts calm down again from Nov. 11th. These transitions seem to repeat "stay" and "dance."

On Yahoo!, hit counts are almost consistent except between Nov. 13 and Nov. 20. Yahoo! seems to update its index during this week. After Nov. 20th, we can observe increased, decreased, and stable clusters.

As a result, we summarize that hit counts repeat "stay" and "dance" as their temporal transition whose increase or decrease speed depends on the clustered queries. The "dance" phase continues up to one week as shown in Fig.5 to Fig.7. During the "dance" phase, hit counts usually change over 30% in comparison with those at the beginning of the phase. Hence, we are able to conclude that hit counts are more reliable when they keep stable numbers during a week in comparison with when they change, because dancing hard implies an index updating phase.

3.2 Basis to Select a Reliable Hit Count for a Query

Based on 3.1, we are able to provide the most reliable hit count for a query within a search engine, i.e., the most reliable hit count among its variation when clicking the "Search" button multiple times in a short time interval or when clicking the "Next" button multiple times in a short time interval, or when clicking the "Search" button on separate days.

From the experiment in Case 1, we do not have to consider the hit count dance caused by clicking the "Search Next" button because it does not have a large effect. Instead, we should consider the hit count dances in Case 2 and Case 3. From the

experiment in Case 2, we are able to obtain the most reliable hit count, $\text{HitCount}(k, k+9)$, where k is defined as the largest number, usually 991 for the search having more than 1,000 matched pages. If a search engine adjusts the last hit count, we should use the hit count just before the adjusted hit count. Moreover, based on the experiment in Case 3, hit counts seem to repeat “stay” and “dance” phenomena. We should use the hit counts that keep almost the same number during a week in order to avoid the search engines’ index updating phase or some other changing phase. Here, the “stay” phase is defined as the phase where the changing ratio of hit counts is less than 30%.

4 Conclusion

In this paper, we provide a basis to adopt hit counts with various types of studies. Previous researches investigated the hit count dance characteristics to find out the search engine to provide the most accurate hit counts among three major search engines. However, they did not clarify the basis to adopt the accurate hit count for a query within a search engine, i.e., how to select the most reliable hit count for a query from among its variation when clicking the “Search” button multiple times or when clicking the “Next” button, or when clicking the “Search” button on separate days. In this paper, we analyzed the characteristics of hit count transition by gathering various types of hit counts over two months by using 10,000 queries. We have concluded that the hit counts with the largest search offset just before search engines adjust their hit counts are the most reliable. Moreover, hit counts are the most reliable when they are consistent, i.e., less than 30% change, over approximately a week.

There remain two aspects after this work, which have not yet been explored fully. The first aspect is the consideration of the queries’ famousness. In this paper, we selected well-used queries. Considering the architecture of search engines such as result cache, the query selection may have some effect on “hit count dance.” Hence, we should also use non-famous queries. The second aspect is the verification time period. We performed our experiment from October 2009 to December 2009. Two months might be insufficient for the conclusion of the hit count dance. Therefore, we should continue our experiment and make hit count dance more obvious.

Acknowledgement

The authors are grateful for the financial support by the Grant-in-Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture (No. 21300038). We would like to thank our anonymous reviewers who have provided helpful comments on the refinement.

References

1. Kilgarriff, A., Gefenstette, G.: Introduction to the Special Issue on the Web as Corpus. *J. of Computational Linguistics* 29(3), 333–347 (2003)
2. Cilibrasi, R.L., Vitanyi, P.M.B.: The Google Similarity Distance. *IEEE Trans. on Knowledge and Data Engineering* 19(3), 370–383 (2007)

3. Matsuo, Y., Sakai, T., Uchiyama, K., Ishizuka, M.: Graph-based Word Clustering using Web Search Engine. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing, pp. 542–550 (2006)
4. Yamamoto, Y., Tezuka, T., Jatowt, A., Tanaka, K.: Honto? Search: Estimating Trustworthiness of Web Information by Search Results Aggregation and Temporal Analysis. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J.X. (eds.) APWeb/WAIM 2007. LNCS, vol. 4505, pp. 253–264. Springer, Heidelberg (2007)
5. Kilgarriff, A.: Googleology is Bad Science. *J. of Computational Linguistics* 33(1), 147–151 (2007)
6. Thelwall, M.: Quantitative Comparisons of Search Engine Results. *J. of the American Society for Information Science and Technology* 59(11), 1702–1710 (2008)
7. Uyar, A.: Investigation of the Accuracy of Search Engine Hit Counts. *J. of Information Science* 35(4), 469–480 (2009)
8. info-plosion,
<http://www.infoplosion.nii.ac.jp/info-plosion/ctr.php/m/IndexEng/a/Index/> (accessed 28/4/2010)
9. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, New York (2008)
10. Anh, V.N., de Kretser, O., Moffat, A.: Vector-Space Ranking with Effective Early Termination. In: Proc. of the 24th Ann. Int'l ACM SIGIR Conf., pp. 35–42 (2001)
11. Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y.S., Soffer, A.: Static Index Pruning for Information Retrieval Systems. In: Proc. of the 24th Ann. Int'l ACM SIGIR Conf., pp. 43–50 (2001)

Selecting Materialized Views for RDF Data

Roger Castillo and Ulf Leser

Humboldt University of Berlin

{castillo,leser}@informatik.hu-berlin.de

<http://www.hu-berlin.de/>

Abstract. In the design of a relational database, the administrator has to decide, given a fixed or estimated workload, which indexes should be created. This so called index selection problem is a non-trivial optimization problem in relational databases. In this paper we describe a novel approach for index selection on RDF data sets. We propose an algorithm to automatically suggest a set of indexes as materialized views based on a workload of SPARQL queries. The selected set of indexes aims to decrease the cost of the workload. We provide a cost model to estimate the potential impact of candidate indexes on query performance and an algorithm to select an optimal set of indexes. This algorithm may be integrated into an existing SPARQL query engine. We experimentally evaluate our approach on a standard query processor. We claim that our approach is the first comprehensive suggestion for the index selection problem in RDF.

Keywords: SPARQL, Indexing, Index Selection, RDF, Materialized Views.

1 Introduction

The index selection problem has been studied since the early 70's and its importance has been well recognized [1]. The basic principle of index selection is to find the best set of indexes for a given workload that fit into a given amount of space. Although well studied into relational databases, as far as we know, there are not approaches, which propose an automated index selection system for RDF data given a workload of SPARQL queries. An RDF data set is a collection of statements called *triples* [2], of the form (s,p,o) where s is a subject, p is a predicate, and o is an object. Each triple states the relation between subject and object. A set of triples can be represented as a directed graph where subjects and objects represent nodes and predicates represent edges connecting these nodes. The SPARQL query language is the official standard for querying RDF repositories [3]. In the relational representation, RDF triples are stored in one or more tables. A SPARQL query consists of patterns. These patterns are translated into one or more SQL queries, depending on the query and the system used. If the SPARQL query consists of more than one pattern, it is translated

into an SQL query that typically contains as many joins as the query has patterns. Optimizing these joins and/or reducing their number is one of the critical issues to obtain scalable SPARQL systems.

In [4] we propose to materialize SPARQL queries and use these materialized views to speed-up queries. We target large data sets and SPARQL queries consisting of many basic graph patterns. Examples of huge data sets are, the UniProt database containing more than 600 million triples [5] or the W3C SWEO Linking Open Data Community with more than 4 billion triples [6]. In such datasets, executing a query with many graph patterns becomes a problem.

```

SELECT ?generecord ?process WHERE {
  ?pubmedrecord ?p mesh:D017966.
  ?article sc:identified_by_pmid ?pubmedrecord.
  ?generecord sc:describes_gene_or_gene_product_mentioned_by ?article.
  ?protein rdfs:subClassOf ?res.
  ?res owl:onProperty ro:has_function.
  ?res owl:someValuesFrom ?res2.
  ?res2 owl:onProperty ro:realized_as.
  ?res2 owl:someValuesFrom ?process.
  ?process <http://purl.org/obo/owl/obo#part_of> go:G0_0007166.
  ?process rdfs:subClassOf go:G0_0007166.
  ?protein rdfs:subClassOf ?parent.
  ?parent owl:equivalentClass ?res3.
  ?res3 owl:hasValue ?generecord.
}

```

Listing 1. Complex SPARQL query to retrieve all genes that are associated with both CA1 Pyramidal Neurons and the signal transduction processes [7]

Consider the query [4] in Listing 1. Executing this query on a conventional SPARQL processor results in the computation of twelve self-joins of a large triple table. However, one can safely assume properties such as, (identified_by_pmid, describes_gene_or_gene_product_mentioned_by), (onProperty, someValuesFrom), and (subClassOf, equivalentClass, hasValue) of an object are used together very often. Therefore, by materializing this information inside the system, the query could be computed with seven joins, as the materialized view would help to retrieve the information on *generecord* and *process*.

In this paper, we provide a novel approach to automate the task of selecting indexes for RDF data given a workload of SPARQL queries. We currently work with a constraint of number of indexes, but extensions to other constraint resources such as storage space is straightforward. For simplicity, in the rest of this work we refer to materialized views as indexes.

This paper is structured as follows. In Section 2 we review related work. Section 3 gives an overview of our approach and introduces algorithms to select a set of indexes. There, we also propose a cost model to evaluate candidate indexes. Section 4 provides an evaluation of our approach. Finally, we conclude in Section 5.

¹ Namespaces omitted for simplicity.

2 Related Work

In relational database systems the problem of index selection has been continuously addressed since the early 70's [1]. In [8], Caprara et al. propose a practical solution that builds on a branch-and-bound algorithm to suggest a reduced set of candidate indexes. Chaudhuri et al. propose an index selection tool in [9]. They implement a cost-driven approach, dividing the problem into three basic stages: First, generation of a set of candidate indexes and selection of those which are more promising based on the query syntax and estimated cost. Second, optimization algorithms to evaluate sets of candidate indexes, and finally, iterative generation of multi-column indexes from the simpler “good” alternatives. An enhancement of this approach is proposed in [10]. Contrary to the previous works, this approach considers the combination of classical index structures, such as B+ trees, and materialized views to define an optimal physical design for a database system.

Similar to [10], we propose an automated selection of a set of indexes in the form of materialized views. We have in common with these previous works that we disregard the cost of updating a view, i.e., we also only consider SELECT queries in the workload. However, instead of focusing on indexing the relational representation of an RDF storage scheme, we fully exploit the RDF graph-structure for indexing. We are not indexing single attributes or triples, but fractions of queries that occur frequently in a given workload. Therefore, our approach suggests a set of native RDF/SPARQL indexes whose concepts are viable for all possible implementations of RDF stores.

In addition to the previous approaches, several efforts have been dedicated to provide data structures or algorithms to index RDF data in relational databases [11][12][13][14][15][16][17][18][19]. These approaches target the clever construction of indexes that are workload-unaware, i.e., they aim at achieving good scalability for any query.

In [4] we described an approach to use a predefined set of materialized SPARQL queries as indexes for RDF data. We propose the evaluation of SPARQL queries using (offline) materialized results of other queries. We refer to those materialized SPARQL queries as RDFMatView indexes. Here, we target the orthogonal problem of selecting a set of RDFMatView indexes for a workload of SPARQL queries. As far as we know, there exists no system for RDF data, which provides functionality similar to our approach. Note that in this paper we assume that the workload consists of distinct queries. An extension of the method to allow queries to appear more than once is straightforward.

3 RDFMatView Index Selection Approach

We propose the evaluation of a given workload of SPARQL queries to suggest a suitable set of RDFMatView indexes. The goal is to globally minimize the estimated response time for all queries contained in the given workload. In this section, we formally introduce all necessary concepts for this idea.

```
SELECT * WHERE {
  ?university rdf:type ub:University;
  ub:name ?uni_name .
}
```

Listing 2. SPARQL query that returns universities and their names

```
SELECT * WHERE {
  ?uni rdf:type ub:University;
  ub:name ?uni_name .
  ?ub_AssistantProfessor ub:worksFor ?uni;
}
```

Listing 3. SPARQL query that returns universities and their professors

```
SELECT * WHERE {
  ?the_uni rdf:type ub:University;
  ub:name ?uni_name .
  ?student ub:studyAt ?the_uni .
}
```

Listing 4. SPARQL query that returns universities and their students

3.1 Workload of SPARQL Queries

Let W be a workload of n different SPARQL queries. Even when the queries in the workload are assumed to be different, there may exist some similarity between their query patterns. As example, consider a simple workload consisting of the three SPARQL queries q_1 , q_2 , and q_3 given in Listing 2-4, respectively. Clearly, we could materialize q_1 and reuse the result for the execution of all three queries, as q_1 is a sub-query of q_2 and q_3 (and, of course, also of q_1). On the other hand, q_1 also is very simple, and pre-computing it might not save much time for the entire workload. Suppose we were allowed to create only one index. In this case, we need to decide whether it is more advantageous to materialize q_1 , gaining limited savings for all queries, or, for instance, materializing only q_3 . This would only help to speed-up the query itself, but nevertheless could offer the highest total savings. As one can see, all queries are different. However, they share triple patterns. Our idea is to discover those shared triple patterns such that they can be used as indexes to improve the processing time of the workload. A candidate index set provides an initial search space for the index selection problem. Similar to index selection approaches for RDBMS (regarding a workload of SQL queries) where the attributes to index are taken from SQL queries, we extract our *index information* from the SPARQL query patterns and generate a set of indexes. Afterwards, we analyze which of them are most frequently used in the workload. Such strategy (*query containment* in the database field) requires to determine which pattern is contained in another query pattern by creating mappings between their elements, and which of them brings more savings in time for the workload. The following section describes our method in detail.

Table 1. An initial set of candidate indexes. One index is generated from each query of the workload.

	$index_1$	$index_2$	$index_3$...	$index_n$
$query_1$	x	0	0	...	0
$query_2$	0	w	0	...	t
$query_3$	y	0	r	...	0
...
$query_n$	z	0	0	...	s

3.2 Candidate Indexes

In [4] we showed that it is possible to use materialized SPARQL queries to speed up the execution of other queries. We refer to a materialized SPARQL query as RDFMatView index. During query processing, the system may decide which of the existing indexes to use. First, it has to decide which indexes it may use for a given query. This task is performed by finding mappings between index and query patterns using a query containment algorithm [20] adapted for SPARQL queries. Essentially, we find all mappings between any index pattern and the query patterns by enumerating all possible cases. If a mapping exists, the index is eligible for that query. Note that, for a given index, there potentially are many different eligible mappings.

In the problem treated here, we assume a workload of SPARQL queries. From this workload, we derive a set of candidate indexes building our search space. There are different methods that may be used to provide a candidate index set. Currently, we generate one candidate index for each query, which is the query itself such that each query can be processed by using at least one index. This approach guarantees that very expensive queries are considered as potential indexes even if they are not useful for processing any other query than themselves. Therefore, the *candidate index set*, denoted as I_c , has as many indexes as the workload has distinct queries. However, for future work, we plan to consider alternative options to create candidate index sets, such as sub-queries with a minimal size constraint and query patterns with overlapping triple patterns.

An index created for a query i sometimes is also eligible for a query j . We represent this information by computing an asymmetric quadratic matrix, where the queries from the workload are the rows and the indexes are the columns. An example is shown in Table 1. A value m_{ij} in the matrix indicates which savings can be achieved by using index v_i for query q_j ; if this value is 0, it means that the index is not eligible for the query. The computation of these values will be described in the following sections. The total savings of an index can be computed by summing up all values in its column.

3.3 Cost Model

To decide which indexes from the candidate index set are the most effective for the given workload, the system needs to evaluate all indexes and their influences

on query processing. This decision should be made based on the expected time reduction that the usage of an index from the candidate index set brings to the execution of the queries from the workload. We refer to these savings as *gain of an index*; such gains need to be compared to the resources (such as number of indexes, storage space, or time to keep it up-to-date) allowed. Currently, our approach suggests an optimal set of indexes regarding costs and a given number of indexes (ν) since no other index information is available (indexes are generated at processing time, and therefore no offline estimations of their exact number of occurrences or processing time are possible)². However, we want to emphasize that the implementation of any other resource constraint in the system is straightforward when additional index information such as number of occurrences or processing time is provided.

We estimate the gain an index offers to a query by comparing the costs it takes to execute the query with or without the index. Our cost model is purely abstract; absolute values have no meaning. We only look at the differences between different costs, which should roughly correlate with the savings in time (as shown in Section 4).

As our knowledge about the indexes is very limited, our cost model is defined on the potential number of occurrences an index could have in a given data graph. To estimate the costs, we need the size of the index pattern $|P|$, which is the number of triples in the query pattern and the size of the data graph $|G|$ (total number of triples). Assume a large triple table containing the RDF data set and a SPARQL query. Since each triple pattern from the query may potentially match all triples from the data graph, we define the cost of an query as follows.

Definition 1 (Cost of a query). *Let q be a query over a data graph G and P_q be the pattern of q . The cost $c(q)$ of q is defined as:*

$$c(q) = |G|^{|P_q|}$$

Since we build candidate indexes from a workload of SPARQL queries, we use this model to estimate costs for both queries and candidate indexes. We refer to the cost of an index as the estimated time we could save when it is precomputed; thus, high costs are better. Having a high cost means that the index pattern covers a larger number of query patterns.

Definition 2 (Saving of an index). *Let i be an index over a data graph G . The saving $s(i)$ of i is defined as:*

$$s(i) = c(i)$$

We generate a set of candidate indexes using each query pattern as an index pattern. Thus, for each query Q of the workload there exist at least one index (generated from the query pattern $P(Q)$), which can improve its processing time.

² An estimation of which size of ν is optimal for the workload is out of the scope of this paper.

However, this is the simplest case, and we look for indexes, which could be used to improve the processing time of more than one query from the workload. This particular case may imply that a query might not be completely covered by an index, i.e., there is a residual part of the query pattern, which is not covered by the index pattern. This fact requires to estimate the processing time of using the index plus evaluating the residual part of the query.

We estimate a cost for processing the residual part by looking at the number of patterns it contains and applying the cost model provided in Definition 3. Using this model to estimate costs we can now define the cost of a query when using indexes. Even though there exists a cost for retrieving the precomputed data, currently we assume real cost of an index $c(i) = 0$; if i is precomputed.

Definition 3 (Cost of a query when using an index). *Let q be a SPARQL query, i an index and let r be the residual part of q when using i , or q otherwise. The cost $c(q, i)$ of executing q using i is defined as follows:*

$$c(q, i) = \begin{cases} c(q), & \text{if } i \text{ not eligible for } q \\ 0, & \text{if } |r| = 0 \\ c(r), & \text{otherwise} \end{cases}$$

Based on the cost for executing a query with or without an index we can now define the gain an index provides when used in a query.

Definition 4 (Gain of an index). *Let q be a query and i an index. The gain of i when used in q is defined as follows:*

$$\text{gain}_q(i) = c(q) - c(q, i)$$

Note that $\text{gain}_q(i) = 0$ if i is not eligible for q .

We can now define the optimal set of indexes given a workload: It is the one set, for which the sum of the gains of the indexes in the set is the highest.

Definition 5 (Gain for a SPARQL workload). *Let W be a workload of SPARQL queries and i be an index. The gain of and index $i \in I$ in W is defined as follows:*

$$\text{gain}_W(i) = \sum_{q \in W} \text{gain}_q(i) | i \in I \}$$

Together, we have:

Definition 6 (Optimal set of indexes). *Let I be a set of candidate indexes for a workload W . Let ν be the maximum number of indexes, which can be suggested for the workload. A subset $S \subseteq I$ for speeding up W is called optimal if the following holds:*

- $\sum_{i \in S} \text{gain}_W(i)$ is maximal and
- $|S| \leq \nu$.

Even when our model accurately tries to capture the influence of an index in the workload, a closer look into it reveals areas of improvement. Especially, the case when two indexes are eligible for the same query, their gains are taken into account for both indexes. This behavior should be avoided since at query processing time, only one index is considered (except in the cases when indexes overlap); resulting in an overestimation of the gains. Therefore, discovering overlapping indexes is a logical next factor to consider in future research.

3.4 Enumeration of Search Space

Finding an optimal subset from all candidate indexes is not trivial. If there are n candidate indexes each offering a specific gain, choosing the optimal set under a space constraint is NP-Complete, as shown in [1] by reduction on the Knapsack problem. This also holds for a uniform space model as in our case. However, there are fast approximation algorithms that have provable quality. Specifically, we propose the use of a greedy heuristic [21], which sorts the items (indexes) in decreasing order of estimated value. We then select indexes in decreasing order until the ν parameter is reached. This algorithm guarantees that our solution is bounded with a value of at least $gain_W(J) \leq \frac{gain_W(I)}{2}$, where $J \subseteq I$ and $gain_W(I)$ is the maximal gain that can be achieved from the candidate index set using ν indexes.

4 Evaluation

This section describes the evaluation of our approach using the Berlin SPARQL Benchmark [22]. We use the ARQ/Jena RDF Storage System (version 2.5.7) [23] and the RDFMatView approach [4] on Postgres 8.2 as framework.

4.1 Dataset and Queries

To evaluate the performance of our approach, we generate a set of nine indexes and compare the workload processing time (using indexes) against standard query processing (without indexes) over four datasets with 250k, 500k, 1M, and 10M triples respectively. Additionally, we evaluate our approach by creating three sets of indexes suggested by our approach containing 3, 4 and 5 indexes respectively. As in the previous test, these sets are used to process the entire workload and their processing time is compared to the workload processing time when using five randomly generated index configurations (each configuration consists of 3 sets containing 3, 4 and 5 indexes). We evaluate these six configurations over an RDF dataset containing 500K triples. Our datasets are based on e-commerce use case information and were created by using the data generator provided in [22]. Based on the set of queries provided in [22], we derive 18 queries as workload. We transformed the query patterns into simple graph patterns (the only form of patterns the current RDFMatView approach implementation can cope with) and removed bindings to variables. Bound variables incur extremely

high selectivity resulting in the retrieval of only a handful of triples. Such queries are well supported by existing index structures and do not require the type of join-optimization that is achieved with the RDFMatView approach; therefore, performance gains would be only marginal.

4.2 Results

Using our approach, we generate a set of indexes for each dataset and evaluate the workload using the RDFMatView approach and plain ARQ (without indexes). To compute the workload processing time, all queries were executed 5 times and average execution times are reported. Figure 1 illustrates the processing time for each workload over the four datasets. For each dataset, the set of suggested indexes contains 9 indexes. These indexes could be used to process 10 queries from the workload respectively. In all scenarios, the workload processing time dramatically improves in comparison to standard query processing.

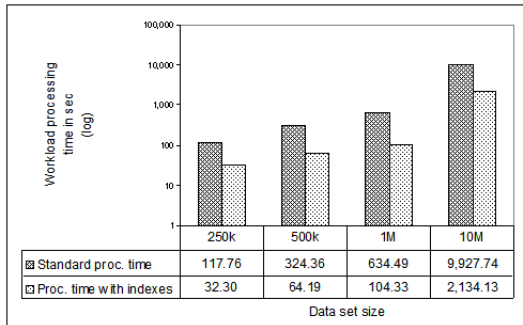


Fig. 1. Workload processing time without using indexes compared to workload processing time using nine suggested indexes. The graphic illustrates that for each dataset, there is a large gain of processing time for the given workload when using the set of indexes suggested by our approach.

To verify the goodness of our approach, we generate three optimal sets of indexes (based on our cost model) consisting of 3, 4, and 5 indexes. We also create five random index configurations with the same number of indexes and evaluate the workload using the RDFMatView system and plain ARQ (without indexes). With this configuration we want to stress that our suggested sets of indexes are a suitable indexing solution for the given workload. Figure 2 shows that all sets suggested by our system improve the processing time of standard query processing (without indexes). In fact, the processing time resulting from using our selected indexes is better than the 66% of the processing time when using random index sets and ARQ. However, it also shows that some random index configurations are quite better than our index selection (see `random_1` and `random_3` with 3 and 5 indexes respectively). A closer look into the workload reveals that some queries

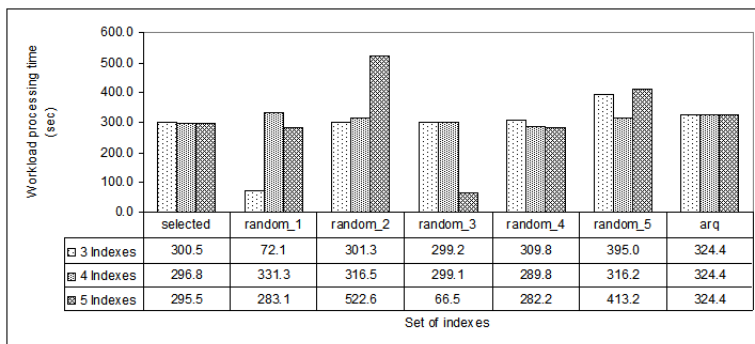


Fig. 2. Workload processing time using six different index configurations over a 500K triples dataset. Each configuration contains three sets with 3,4 and 5 indexes respectively. The workload was processed using indexes selected by our approach, randomly selected indexes and without indexes (standard ARQ).

generate large numbers of matches when they are queried against the dataset independent from their number of triple patterns, incurring in costly processing time. Clearly, our cost model does not consider these cases since indexes are evaluated regarding their number of triple patterns and no assumption about real number of occurrences is foreseen. In [4] has been proved that covering a large number of patterns using as few indexes as possible significantly improves the processing time. These savings in time are due to the number of joins required to answer the SPARQL query, which are drastically reduced using this approach. To perform this process, RDFMatView requires overlapping indexes³, which allow an optimal covering of the query patterns. Since our index selection approach suggests indexes regarding the complete query pattern for each query, it may occur that those indexes do not overlap at all. In queries where the number of covered patterns is smaller than the number of residual patterns, the final processing time may increase depending on the number of partial results resulting from the covered and uncovered patterns using RDFMatView query processing. Thus, the selection of potential overlapping indexes which could be used to process queries from the workload and a more accurate estimation of the number of query occurrences are natural next factors to consider in future work.

5 Conclusions and Future Work

We introduced and developed a system to analyze SPARQL queries, which returns a set of RDFMatView indexes such that their application in the query processing improves the entire workload processing time. Candidate indexes are evaluated by a cost model regarding their potential number of occurrences in a

³ Indexes with triple patterns in common.

given dataset. Results show that indexes selected using our approach can dramatically decrease the workload processing time. Furthermore, we analyzed and compared the processing time of the workload using six different index configurations. Each configuration consists of 3 index sets with 3, 4 and 5 indexes respectively. One of these configurations is suggested by our system and five are randomly generated. Upon analysis, indexes selected by our approach outperform randomly selected indexes in most cases by assuring a suitable indexing solution for the given workload. Up to now, our approach is restricted to queries containing only a basic graph pattern and without filters, modifiers and blank nodes. We currently work with a constraint on the number of indexes, but extensions to other constraints like storage space are straightforward. Currently, we are working on novel ideas for optimization of the index selection process analyzing not only the given queries but also generating potential indexes based on sub patterns of the query patterns (assuming a minimum pattern size). Another interesting improvement is to generate statistics using fixed predicates, for instance $count(?x, p, ?y)$, which can be used as a fine-grained cost model for each triple pattern. We believe that such an approach should improve the quality of the suggested indexes.

References

1. Comer, D.: The difficulty of optimum index selection. *ACM Trans. Database Syst.* 3(4), 440–445 (1978)
2. Manola, F., Miller, E.: *RDF Primer*, W3C Recommendation (February 2004)
3. Prud’hommeaux, E., Seaborne, A.: *SPARQL Query Language for RDF*, W3C Recommendation (April 2008)
4. Castillo, R., Leser, U., Rothe, C.: *RDFMatView: Indexing RDF Data for SPARQL Queries*. Technical report, Humboldt University of Berlin (2010)
5. UniProt: *RDF Dataset*, <http://dev.isb-sib.ch/projects/uniprot-rdf/>
6. W3C SWEO Community Project, *Linking open data on the semantic web*, <http://esw.w3.org/topic/sweoig/taskforces/communityprojects/linkingopendata/>
7. Stenzhorn, H., Srinivas, K., Samwald, M., Ruttenberg, A.: Simplifying access to large-scale health care and life sciences datasets. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 864–868. Springer, Heidelberg (2008)
8. Caprara, A., Fischetti, M., Maio, D.: Exact and approximate algorithms for the index selection problem in physical database design. *IEEE Transactions on Knowledge and Data Engineering* 7(6), 955–967 (1995)
9. Chaudhuri, S., Narasayya, V.R.: An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In: *VLDB 1997: Proceedings of the 23rd International Conference on Very Large Data Bases*, pp. 146–155. Morgan Kaufmann Publishers Inc., San Francisco (1997)
10. Agrawal, S., Chaudhuri, S., Narasayya, V.R.: Automated Selection of Materialized Views and Indexes in SQL Databases. In: *VLDB 2000: Proceedings of the 26th International Conference on Very Large Data Bases*, pp. 496–505. Morgan Kaufmann Publishers Inc., San Francisco (2000)

11. Groppe, J., Groppe, S., Ebers, S., Linnemann, V.: Efficient Processing of SPARQL Joins in Memory by Dynamically Restricting Triple Patterns. In: Proceedings of the 24th ACM Symposium on Applied Computing (ACM SAC 2009), Honolulu, Hawaii, March 8-12, pp. 1231–1238. ACM, New York (2009)
12. Groppe, S., Groppe, J., Linnemann, V.: Using an Index of Precomputed Joins in order to speed up SPARQL Processing. In: Cardoso, J., Cordeiro, J., Filipe, J. (eds.) Proceedings 9th International Conference on Enterprise Information Systems, ICEIS 2007 (1), Funchal, Madeira, Portugal, June 12-16, DISI, pp. 13–20. INSTICC (2007)
13. Jinghua Groppe, S.G., Kolbaum, J.: Optimization of SPARQL by Using coreSPARQL. In: Cordeiro, J., Filipe, J. (eds.) Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009), Milano, Italien, Mai 6 - 10, DISI, pp. 107–112. INSTICC (2009)
14. Olaf Hartig, R.H.: The SPARQL Query Graph Model for Query Optimization. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 564–578. Springer, Heidelberg (2007)
15. Matono, A., Amagasa, T., Yoshikawa, M., Uemura, S.: An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays. In: Proceedings of SWDB 2003, pp. 151–168 (2003)
16. Udea, O., Pugliese, A., Subrahmanian, V.S.: GRIN: A Graph Based RDF Index. In: AAAI, pp. 1465–1470 (2007)
17. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. Proc. VLDB Endow. 1(1), 1008–1019 (2008)
18. Yan, X., Yu, P.S., Han, J.: Graph indexing based on discriminative frequent structure analysis. ACM Trans. Database Syst. 30(4), 960–993 (2005)
19. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. Proc. VLDB Endow. 1(1), 647–659 (2008)
20. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10(4), 270–294 (2001)
21. Dantzig, G.: Linear Programming and Extensions. Princeton University Press, Princeton (August 1998)
22. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems - Special Issue on Scalability and Performance of Semantic Web Systems (2009)
23. ARQJena: ARQ - A SPARQL Processor for Jena (2010), <http://jena.sourceforge.net/ARQ/>

Semantic Wonder Cloud: Exploratory Search in DBpedia

Roberto Mirizzi¹, Azzurra Ragone^{1,2}, Tommaso Di Noia¹,
and Eugenio Di Sciascio¹

¹ Politecnico di Bari – Via Orabona, 4, 70125 Bari, Italy
mirizzi@deemail.poliba.it, {ragone,dinoia,disciacio}@poliba.it

² University of Trento – Via Sommarive, 14, 38100 Povo (Trento), Italy
ragone@disi.unitn.it

Abstract. Inspired by the Google Wonder Wheel^[1], in this paper we present Semantic Wonder Cloud (SWOC): a tool that helps users in knowledge exploration within the DBpedia dataset by adopting a hybrid approach. We describe both the architecture and the user interface. The system exploits not only pure semantic connections in the underlying RDF graph but it mixes the meaning of such information with external non-semantic knowledge sources, such as web search engines and tagging systems. Semantic Wonder Cloud allows the user to explore the relations between resources of knowledge domain via a simple and intuitive graphical interface.

1 Introduction

“The Web, they say, is leaving the era of search and entering one of discovery. What’s the difference? Search is what you do when you’re looking for something. Discovery is when something wonderful that you didn’t know existed, or didn’t know how to ask for, finds you.”^[11] When Jeffrey O’Brien wrote this sentence in 2006, he was referring to recommender systems on the web^[1]. Nevertheless, it can be also interpreted in a broader sense if we think that the recommended items are pieces of knowledge extracted from a knowledge base. In other words, that sentence summarizes in an excellent way the problem of information discovery in a knowledge repository. The main issue there is assisting a user to discover new knowledge in a repository and what might be the role played by the interconnected nature of the Semantic Web.

Thanks to its hyperlinked structure, the Web defined a new way of browsing documents and knowledge: selection, navigation and trial-and-error tactics^[12] were and still are exploited by users to search for relevant information satisfying some initial requirements. In^[12], the author distinguishes among three distinct categories of search strategies: lookup, learn and investigate. He also groups the last two under the umbrella of *exploratory search*. Lookup has database systems as background technology and it is used for finding exact information, i.e.,

¹ <http://www.googlewonderwheel.com/>

records in a database or documents containing a specific keyword. The above mentioned search strategy is by far the most used in the current “syntactic” web where the information sources are mainly based on textual documents. Nevertheless, it is applied in many of the current applications exploiting structured metadata available in the Semantic Web: records are RDF triples and keywords are URIs with a specific associated meaning (semantics). In the other two search strategies – learn and investigate – the final aim of the end user changes [13]. She is no more interested in a fact retrieval or query answering but her focus is on:

Learn: Knowledge acquisition, comprehension of new concepts and ideas, comparison of information;

Investigate: Analysis, synthesis, evaluation, knowledge discovery.

In this paper, inspired by *Google Wonder Wheel*, we present *Semantic Wonder Cloud (SWOC)*: a tool for exploratory knowledge search for DBpedia [5]. Differently from other RDF explorers, SWOC allows the user to explore DBpedia not just via directed links in the RDF dataset but via newly computed associations between DBpedia nodes. Such new associations are computed exploiting extra-knowledge extracted from Web search engines and social tagging systems. Main contributions on this paper are:

- novel approach to *exploratory search* in the Semantic Web: in this work we focus on DBpedia and present a new way to explore a RDF dataset discovering new knowledge associations;
- visual tool to guide the user during the knowledge discovery process;
- hybrid approach to rank pairs of resources on DBpedia: our system mixes a *semantic-based* approach, which relies on the structure of the RDF graph, and a *text-based* IR approach, which computes the popularity of the resources by querying external information sources

The remainder of this paper is structured as follows. In Section 2 we present **Linked Data** and in particular **DBpedia** as the knowledge base we use in this work. Section 3 provides details on the implementation of SWOC. Section 4 overviews related work. An outline of the future work concludes the paper.

2 Linked Data and DBpedia

The idea behind **Linked Data** [4] is using the Web to allow linking data and simplifying the publication of new interconnected data on the Web. It proposes a new method of exposing, connecting and sharing data through dereferenceable URIs on the Web. The goal is to extend the Web by publishing various open datasets as RDF triples on the Web and by setting RDF links between data items from different data sources. According to this aim, URIs are fundamental to identify everything. Using HTTP URIs, things can be referred to and looked up both by people and by agents. **Linked Data** uses RDF to describe things in the world. In this paper we focus on **DBpedia** [5], that is one of the main clouds of the

Linked Data graph. It is the machine-understandable equivalent of Wikipedia project. It is possible to ask queries against DBpedia (through its SPARQL endpoint <http://dbpedia.org/sparql>), and link other data sets on the web to DBpedia data. Currently the DBpedia dataset (version 3.5.1²) contains almost three million and half resources, including more than three hundred thousand persons, over four hundred thousand places, thousands of films, companies, music albums, etc.. All this information is stored in RDF triples. The whole knowledge base consists of over one billion triples. DBpedia labels and abstracts of resources are stored in 92 different languages. The graph is highly connected to other RDF dataset of the Linked Data cloud. There are more than half million categories, inherited from Wikipedia, and almost one hundred thousand YAGO ²¹ categories. DBpedia has many strong points over existing knowledge bases: it is spread over many domains; being based on Wikipedia, it represents a real community agreement; it follows the changing in Wikipedia, so it is continuously updated; it is multilingual. Moreover DBpedia has a central role in the Linked Data community effort: it is one of the central interlinking-hubs of the emerging Web of Data, inducing data providers to link their RDF datasets to DBpedia. The DBpedia RDF dataset is hosted and published using *OpenLink Virtuoso*³. This infrastructure gives access to DBpedia's RDF data through a SPARQL endpoint, with HTTP support for any Web client's standard GETs for HTML or RDF representations of DBpedia resources. In DBpedia there is a special kind of resources called **categories**⁴. Since in Wikipedia they are used to classify and cluster sets of documents, in DBpedia they classify sets of resources. They might be seen as abstract concepts describing and clustering sets of resources. As a matter of fact, to stress this relation, every DBpedia category is also a `skos:Concept`. Moreover, since DBpedia categories have their own labels we may think of these labels as names for clusters of resources. Most categories can have a number of other categories listed as subcategories. Exploring from more general categories to more specific ones it is possible to cluster resources according to more fine-grained domains. Similarly, navigating towards more general categories the obtained clusters will be broader and less domain-specific. In order to avoid an explosion of articles belonging to a given Category, articles are not usually placed in every category which they logically belong to. In many cases they are placed just in the more specific subcategories. For this reason, in Wikipedia it may be necessary to dig up to find all the categories an article belongs to. DBpedia maintains these hierarchical relations using the SKOS vocabulary. In particular the property `skos:subject` of the SKOS vocabulary relates a Wikipedia article to its Category. Actually the `skos:subject` property has been deprecated from the SKOS vocabulary, although it has not yet been replaced by any other property⁵.

² <http://wiki.dbpedia.org/Downloads351>

³ <http://virtuoso.openlinksw.com/>

⁴ <http://en.wikipedia.org/wiki/Help:Category>

⁵ A natural candidate for the replacement could be the `dc:subject` property, belonging to *Dublin Core* specification, as proposed in <http://www.w3.org/2006/07/SWD/wiki/SkosDesign/Indexing>

Similarly in DBpedia a Category is linked to another Category (specifically a subcategory to its category) through the `skos:broader` property.

3 SWOC: Semantic Wonder Cloud

The graph-based structure of datasets in the Semantic Web allows a natural visualization and browsing of the formalized information simplifying the implementation of learning and investigating strategies for knowledge acquisition and discovery. Using an *Orienteering* [22] behavior, a user starts from an initial vague idea of what she is looking for and navigates through the information space. It has been shown that *Orienteering* navigation decreases the cognitive load, maintains a sense of location and gives a better feeling for context [10].

Effective user interfaces play a crucial role in order to provide a satisfactory user experience during an exploratory search. There are two main trends in visualizing and navigating RDF datasets [6]: via browsing a labeled oriented graph or displaying RDF properties as browsable facets of a node. The main issue of both approaches is to filter information which is not relevant for the explorative task. If we visualize all the triples together with their connections we have a huge unreadable representation of the underlying knowledge [18]. In fact, RDF triples are conceived to represent information for machine-to-machine interaction. A lot of information triplified in datasets is very useful in many automated computational tasks (e.g., web service interaction) but it is completely useless from a human perspective during an exploratory search [7].

In this section we introduce *Semantic Wonder Cloud* (SWOC), a tool that helps users in exploratory knowledge search in DBpedia. The front-end of the system, inspired by *Google Wonder Wheel*, is available at <http://sisinflab.poliba.it/semantic-wonder-cloud/index/>. A sketch of the functional architecture is represented in Figure 1. SWOC consists of two main subsystems. The first subsystem, the SWOC explorer, is a Graphical User Interface (GUI) designed to allow exploration of DBpedia. The second subsystem – *DBpediaRanker* – is the back-end of the whole system whose main task is to compute similarity between pairs of DBpedia nodes. In order to compute a *similarity value* between two resources res_1 and res_2 , *DBpediaRanker* performs three main tasks:

- analysis of the underlying **RDF graph structure**;
- **textual analysis** on the abstract of the resources;
- exploitation of **external information sources**, such as search engines and social tagging systems.

More details are provided in Section 3.1

We stress that, differently from common RDF visualization tools⁶, SWOC does not visualize the DBpedia RDF graph as it is. There are some key differences w.r.t. other RDF explorers.

⁶ By way of example refer to

<http://timerollson.wordpress.com/2007/09/30/links-for-2007-09-30/> and <http://www.mkbergman.com/414/large-scale-rdf-graph-visualization-tools/>

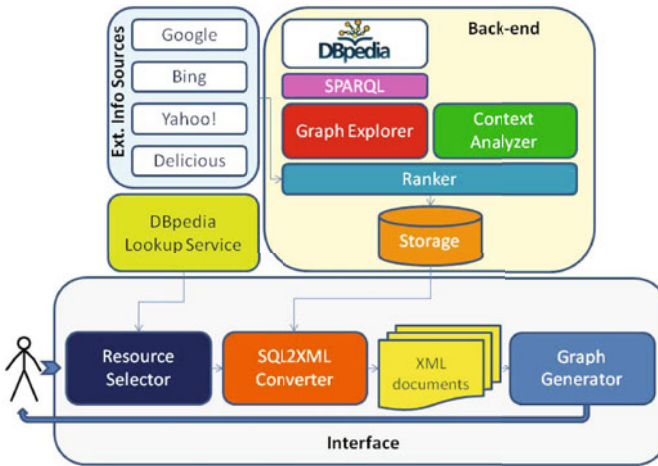


Fig. 1. *Semantic Wonder Cloud* functional architecture

- The graph displayed by SWOC represents possible new associations between resources computed by mixing the *semantic* knowledge formalized in the DBpedia dataset and the *non-semantic* statistical knowledge available from web search engines and social tagging systems. Hence, the navigation is driven by knowledge associations between resources rather than guided by the structure of the underlying dataset. Displayed associations may even not exist in the original DBpedia graph.
- Each node in the displayed graph has a different size representing how relevant it is with respect to the resource represented in the center of the graph itself.
- There are two main classes of nodes in SWOC: one representing DBpedia categories, the other representing instances. This reflects in some way the different nature of the knowledge they represent. Categories are used to group and classify resources. Hence, if the user clicks on a category node then she sees the most relevant (popular) instances of that category together with its most relevant categories.

Referring to the previous characteristics, our tool has been designed to be a user centered explorer of RDF knowledge bases rather than a common RDF graphic visualizer. Actually, SWOC can not be considered an RDF visualizer in the classical sense. It is a tool useful to help end-users in exploratory knowledge search. Usually end-users do not care about obtaining a sketch of the whole result set (think to results returned by a search engine: rarely you would go beyond the second page), but are more interested in finding something *relevant* among the first results. Thanks to SWOC, end-users can navigate from a resource to another one exploring and discovering new knowledge. Following the exploratory search paradigm, SWOC differentiates also from traditional search engines. The latter

allow to find what you are looking for and already know, SWOC allows to explore what you probably did not know to exist.

3.1 Backend

In this section we describe the back-end of the system, based on our hybrid ranking algorithm *DBpediaRanker* [14], used to rank resources in DBpedia. This algorithm computes the *relevance* of DBpedia resources w.r.t. a starting node. In a nutshell, *DBpediaRanker* explores the DBpedia graph and queries external information sources in order to compute a *similarity value* for each pair of resources reached during the exploration.

The graph browsing, and the consequent ranking of resources, is performed *offline* and, at the end, the result is a weighted graph where nodes are DBpedia resources and weights represent the similarity value between any pair of nodes. The graph so obtained will then be used at *runtime* in the *exploratory search*, to look for resources that are semantically related to the initial one. The similarity value between any pair of resources in the DBpedia graph is computed querying search engines and social bookmarking systems and exploits *textual* and *link analysis* in DBpedia. For each pair of resources in the explored graph, we perform a query to each external information source: we search for the number of returned web pages containing the labels of each node individually and then for the two labels together. Moreover, we look at, respectively, **abstracts** in Wikipedia and **wikilinks**, i.e., links between Wikipedia pages. Specifically, given two resource nodes res_1 and res_2 , we check if the label of node res_1 is contained in the abstract of node res_2 , and vice versa. The rationale behind this check is that if a DBpedia resource name appears in the abstract of another DBpedia resource it is reasonable to think that these resources are related with each other. For the same reason, we also check if the Wikipedia page of resource res_1 has a link to the Wikipedia page of resource res_2 , and vice versa.

The back-end of SWOC is composed by the following components:

- **Graph Explorer:** it explores the DBpedia graph, through SPARQL queries, starting from a set of initial seeds. Each node is explored within a predefined number of hops and following a predefined set of properties. If you want to explore only a domain-specific subset of the DBpedia graph (see below), the initial seeds have to belong to that domain (i.e. to the same context). During our experiments we set the maximum explored depth equal 2, and we explored the graph following two properties belonging to the SKOS vocabulary, i.e., `skos:subject` and `skos:broader` (see Section 2). A deep explanation of these choices is provided in [14].
- **Context Analyzer:** it allows to limit the exploration of the graph to a specific context. In this paper the exploration has been limited to the *IT* domain (specifically *databases* and *programming languages*). It uses the *Ranker* (see below) to determine if a resource belongs to the context. The context is represented by the most popular DBpedia categories (see Section 2), i.e., the categories that are reached more often during the exploration of the graph.

For example, in the selected domain, one of the most popular categories is *Programming languages*⁷.

- **Ranker:** this is the core component of the system. It determines a similarity value between any pairs of nodes (res_1 and res_2) discovered by the *Graph Explorer*; this similarity value is the weight associated to the edge between the two resources. The score is computed by querying the external information sources with the following formula:

$$sim(res_1, res_2, info_source) = \frac{p_{res_1, res_2}}{p_{res_1}} + \frac{p_{res_1, res_2}}{p_{res_2}} \quad (1)$$

where p_{res_1, res_2} is the number of documents that contain (or have been tagged with) both the label associated to res_1 and the one associated to res_2 , and p_{res_1} and p_{res_2} are the number of documents that contain (or have been tagged by) the label associated to respectively res_1 and res_2 . Furthermore *Ranker* exploits additional information from *DBpedia*. In fact it checks if there is a *dbpprop:wikilink* between res_1 and res_2 and vice versa, and in positive case it assumes a stronger relation between the two resources. Finally, the component checks if the *label* of res_1 is contained in the *abstract* of res_2 and vice versa.

- **Storage:** all the similarity values between the pairs of resources, together with the “popularity” of each resource (calculated as the number of times that each node is reached in the exploration) are stored in a DBMS for an efficient retrieval at runtime.

3.2 Interface

The interface of *SWOC* is a Flash-based web application that presents the knowledge stored in the back-end (see Section 3.1). A screenshot of the web interface is presented in Figure 2.

The first step for the user is the choice of a node to start the exploration. This is done by keying in some characters, concerning a subject to explore, in the text field marked as (a) in Figure 2. The system responds with an auto-complete list containing a set of labels which refer to *DBpedia* resources where each of them is described by a unique *URI* which identifies it with no ambiguity. This list is obtained by querying the *DBpedia URI* lookup web service⁸. Note that, thanks to the uniqueness of *DBpedia URI* identifying suggested resources, the system does not suffer from problems such as synonymy or polysemy as for simple keywords-based queries to search engines.

After the user selects a resource res_1 from the drop down list, the system generates a graph, centered in res_1 (that has the maximum circle radius), and whose constellation is constituted by the ten most similar resources res_i , ($i = 1, \dots, 10$), w.r.t. res_1 . If a *DBpedia* category is highly relevant w.r.t. res_n , it is drawn in the constellation with a different color (the radius of the circle follows the same rules as seen before).

⁷ http://dbpedia.org/resource/Category:Programming_languages

⁸ <http://lookup.dbpedia.org/api/search.asmx>

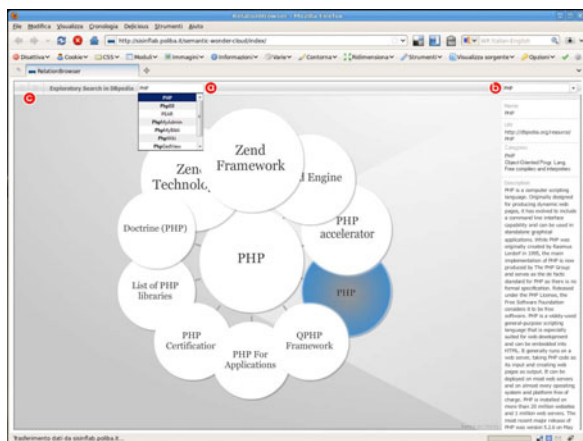


Fig. 2. Semantic Wonder Cloud web-interface

As seen in Section 3.1, the similarity value between res_1 and its constellation has been calculated off-line by the *Ranker*. In the interface the similarity value is normalized on a scale in the range [1, 10] and rounded to the nearest integer, where 10 means that res_i is highly relevant w.r.t. res_1 (according to our *Ranker*), and 1 means that res_1 and res_i are not very similar. These values influence the size of the circle radii: the bigger the constellation’s circles are, the more important the referring resources are w.r.t. res_1 .

In order to generate the graph, SWOC uses a XML file containing information on: relation types (instance, category), the types of node, the nodes to be plotted and the relations among them. An example of such file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<RelationViewerData>
  <Settings appTitle="Semantic Wonder Cloud" startID="7" defaultRadius="150" maxRadius="180">
    <RelationTypes>
      <UndirectedRelation color="0xAAAAAA" lineSize="4"/>
    </RelationTypes>
    <NodeTypes>
      <Node1/>
      <Node2/>
      ...
      <Node10/>
    </NodeTypes>
  </Settings>

  <Nodes>
    <Node10 id="7" name="PHP" URI="http://dbpedia.org/resource/PHP" dataURL="moreNodes.php?id=7">
      <![CDATA[
        PHP: Hypertext Preprocessor is a widely used, general-purpose scripting language that was originally
        designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded ...
      ]]>
    </Node10>
    ...
  </Nodes>

  <Relations>
    <UndirectedRelation fromID="7" toID="69"/>
    ...
  </Relations>
</RelationViewerData>
```

Each node corresponds to a DBpedia resource, and it can be uniquely identified by its *URI*. It has also an abstract which goes in the *CDATA* section of the Node content.

Being the whole DBpedia dataset constituted by more than three millions resources, it was not feasible to have a single XML file containing all the nodes. Even in cases where the *Context Analyzer* (see Section 3.1) had limited the graph exploration, the number of nodes would still be large, and the relations among nodes would be a really huge set⁹. For this reason we opted a dynamic creation of the required XML file. For each selected resource (i.e., a node that appears in the center of the graph), the corresponding XML file is dynamically created at run-time by querying the DB populated by the *Storage* module in the off-line computation. The *SQL2XML Converter* component of the system (see Figure 1) takes as input the *ID* of the resource res_n to explore, then it queries the underlying database and generates the XML file containing the top-10 relevant nodes w.r.t. res_n . Finally the *Graph Generator* accepts as input the XML file previously created and plots the corresponding graph on the screen.

During the exploration, a local list (marked as *(b)* in Figure 2) is populated with discovered nodes up to that moment: it is possible to browse a node from that list in order to directly explore it. Moreover the system keeps track of the exploration path (marked as *(c)* in Figure 2): the user can move backward and forward through resources that have previously been explored.

On the right side of the interface, an area is displayed containing the DBpedia abstract of the selected resource res_n , together with its *URI* and the set of the most relevant DBpedia categories res_n belongs to.

4 Related Work

The rapid growth of semantic metadata and related ontologies in the Semantic Web of Data asks for new tools and approaches to data exploration and visualization. Nowadays, RDF datasets with their formalized knowledge are not only developed for machine-to-machine interaction but also for knowledge discovery and navigation. One of the main issues to be faced is how to manipulate these huge repositories of information in a “*overview first, zoom and filter, then details-on-demand*” fashion as pointed out in [19]. Here we discuss only some approaches representative of the graph-based approach and of the faceted one even though alternative solutions have been adopted for specific tasks [17]. We do not focus on issues typical of clustering and faceted categories for information exploration (see [9]) but we concentrate on specific problems related to the semantic nature of the underlying data. Many approaches to semantic data visualization adopt a graph-based approach exploiting concept relations, entities aggregation and number of nodes instantiating a given class. In [15] the authors focus on discovering disconnections in the ontology graph for a visualization of smaller graphs. Although the smaller dimension of the visualized graphs the approach does not scale well for huge dataset as DBpedia. Moreover, it exploits only topological information to suggest the navigation of the graph. In [20], a cluster-based visualization is adopted. Also in this case only relations explicitly stated in the

⁹ Each node in the DBpedia graph has on the average more than half a thousand nearby nodes within two hops.

dataset are displayed. A slightly different approach is adopted in [10] where the author focuses on a visual rendering of information stored in Freebase¹⁰ and Wikipedia. In the latter case they use SemanticProxy¹¹ to extract structured information from textual pages. In this case the graph displays only some aggregated and filtered information related to a given resource. Faceted browsing [23] has been widely adopted for many RDF dataset spanning from DBpedia to DBLP. A lot of work has been produced in this area. Faceted browsing improves usability over current interfaces and RDF visualizers as it provides a better information lookup w.r.t. to keyword searches. Nevertheless faceted interfaces are domain-dependent, do not fully support graph-based navigation, and they become difficult to use for the end user as the number of presented choices grows. An implementation for faceted navigation of arbitrary RDF data is presented in [16]. They identify important facets by ranking the predicates that best represent and most efficiently navigate the dataset. Differently from our approach, their ranking metrics take into account only the structure of the dataset while we also exploit external information sources to improve the results. Hahn et al. [8] present a faceted browser for Wikipedia. The system is based on the DBpedia data extraction framework and Neofonie¹² search technology. Faceted views are also available in Virtuoso¹³, but here ranking is obtained only from text and entity frequency while semantics associated with links is not explored. This aspect leads to a poor “semantic” ranking in most cases. “Context-aware semantic association ranking” [2] is useful because it allows to find and present to the end-user the top relevant information. A common problem with faceted browsing is the impossibility to navigate through relations different from the ones *explicitly* represented in the dataset. On the contrary SWOC allows to discover new knowledge by exploring also *implicit* relations that do not existed explicitly in the dataset.

5 Conclusions and Future Work

Ten years after the seminal paper by Tim Berners-Lee et al. [3], the Semantic Web is finally becoming a well established reality. New technologies have been developed able to manipulate large sets of semantic metadata available in on-line RDF datasets. The Semantic Web, as a web of data, can be considered nearly done. Together with the availability of new tools to manipulate semantic metadata, we see the flourishing of mash-ups able to aggregate metadata coming from heterogeneous repositories and new semantic search engines able to provide more precise results to the user, as well as new opportunities and new means of manipulating semantic information appeared. Thanks to the highly interconnected nature of semantic metadata the user may browse knowledge repositories

¹⁰ <http://www.freebase.com>

¹¹ <http://www.semanticproxy.com>

¹² <http://www.neofonie.de/index.jsp>

¹³ <http://www.openlinksw.com/dataspace/dav/wiki/Main/VirtuosoFacetsViewsLinkedData>

discovering new relations and information. In this paper we presented **SWOC**: a tool for exploratory knowledge search in **DBpedia**. The tool exploits both knowledge encoded in **DBpedia** graph and statistical knowledge disseminated in web search engines and social tagging systems in order to find and display new associations between pair of nodes based on their *semantic similarity*. Thanks to the graphical interface of **SWOC**, the end-user is guided through the exploration of the knowledge space represented by the **DBpedia** dataset. The associations generated by the system allow the user to discover new knowledge via a simple and intuitive point-and-click interaction with the graphical interface. We plan to use **SWOC** has a new way to navigate through tags in social web sites like Delicious in order to mix exploratory search and lookup search for a better user experience. The idea is that once the user finds a new knowledge item represented by the corresponding **DBpedia** *URI* then she is redirected to its corresponding web resources/documents via a lookup in tagging systems or even in web search engines.

Acknowledgment

We are very grateful to Diego Guarino and Moritz Stefaner for the implementation of **SWOC** and to Enrico Motta for fruitful discussion. This research has been supported by Apulia Strategic projects PS_092, PS_121, PS_025.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17(6), 734–749 (2005)
2. Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I.B., Sheth, A.P.: Context-Aware Semantic Association Ranking. In: *Proceedings of SWDB 2003, The first International Workshop on Semantic Web and Databases, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7-8*, pp. 33–50 (2003)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* 284(5), 34–43 (2001)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22 (2009)
5. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: *Dbpedia - a crystallization point for the web of data*. In: *Web Semantics: Science, Services and Agents on the World Wide Web (July 2009)*
6. Deligiannidis, L., Kochut, K.J., Sheth, A.P.: Rdf data exploration and visualization. In: *CIMS 2007: Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience*, pp. 39–46. ACM, New York (2007)
7. Geroimenko, V., Chen, C.: *Visualizing the Semantic Web: XML-Based Internet and Information Visualization*, 2nd edn. Springer, London (2006)
8. Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Brgle, M., Dwiger, H., Scheel, U.: Faceted wikipedia search. In: *13th International Conference on Business Information Systems, BIS 2010* (2010)

9. Hearst, M.A.: Clustering versus faceted categories for information exploration. *ACM Commun.* 49(4), 59–61 (2006)
10. Hirsch, C., Hosking, J., Grundy, J.: Interactive visualization tools for exploring the semantic graph of large knowledge spaces. In: *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW 2009)*, IUI 2009 (2009)
11. O'Brien, J.M.: The race to create a 'smart' Google (2006), http://money.cnn.com/magazines/fortune/fortune_archive/2006/11/27/8394347/index.htm?section=money_latest
12. Marchionini, G.: Exploratory search: from finding to understanding. *ACM Commun.* 49(4), 41–46 (2006)
13. Marchionini, G., Shneiderman, B.: Finding facts vs. browsing knowledge in hypertext systems. *Computer* 21(1), 70–80 (1988)
14. Mirizzi, R., Ragone, A., Di Noia, T., Di Sciascio, E.: Ranking the linked data: the case of dbpedia. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) *ICWE 2010*. LNCS, vol. 6189, pp. 337–354. Springer, Heidelberg (2010)
15. Mutton, P., Golbeck, J.: Visualization of semantic metadata and ontologies. In: *IV 2003: Proceedings of the Seventh International Conference on Information Visualization*, Washington, DC, USA, p. 300. IEEE Computer Society, Los Alamitos (2003)
16. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for rdf data. 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. 559–572. Springer, Heidelberg (2006)
17. Petrelli, D., Mazumdar, S., Dadzie, A.-S., Ciravegna, F.: Multi visualization and dynamic query for effective exploration of semantic data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) *ISWC 2009*. LNCS, vol. 5823, pp. 505–520. Springer, Heidelberg (2009)
18. Schraefel, M., Karger, D.: The pathetic fallacy of rdf. In: *International Workshop on the Semantic Web and User Interaction, SWUI 2006* (2006)
19. Shneiderman, B.: The eyes have it: a task by data type taxonomy of information visualizations. In: *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann Publishers Inc., San Francisco (2003)
20. Stuckenschmidt, H., Van Harmelen, F., De Waard, A., Scerri, T., Bhogal, R., Van Buel, J., Crowlesmith, I., Fluit, C., Kampman, A., Broekstra, J., Van Mulligen, E.: Exploring large document repositories with rdf technology: The dope project
21. Suchanek, F., Kasneci, G., Weikum, G.: YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In: *16th International World Wide Web Conference (WWW 2007)*, pp. 697–706 (2007)
22. Teevan, J., Alvarado, C., Ackerman, M.S., Karger, D.R.: The perfect search engine is not enough: a study of orienteering behavior in directed search. In: *CHI 2004: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 415–422. ACM, New York (2004)
23. Yee, K.-P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: *CHI 2003: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 401–408. ACM, New York (2003)

Managing Adaptivity in Web Collaborative Processes Using Policies and User Profiles

Juri Luca De Coi¹, Marco Fisichella¹, and Maristella Matera²

¹ Forschungszentrum L3S, Hannover 30167, Germany
{decoi, fisichella}@L3S.de

² Politecnico di Milano, Milano 20133, Italy
matera@elet.polimi.it

Abstract. *Adaptive Web collaborative processes* are flexible processes executed on the Web, that can configure themselves according to information not available at definition time. In this paper we show how such processes can be supported by integrating an engine for flexible process definition and execution, COOPER, and a policy engine, PROTUNE. The resulting framework is *open* in its nature, since it can integrate any (external) source of data. In particular, we show how our framework can exploit user profiles available on the Web to support process adaptivity.

1 Introduction

In several contexts, the success and popularity of workflow applications is limited by the intrinsic complexity of Workflow Management Systems (WfMSs), and especially by the low flexibility of workflow enactment, which leaves no freedom to process actors to tailor processes and process activities to their specific needs [15]. This is especially true whenever different actors have to coordinate toward a common goal (e.g., the release of some artifact), since *collaborative processes* are difficult to predict completely in advance and, unlike traditional business processes, they escape the ability of being fully modeled [7]. For this reason, process engines are needed which are flexible enough to be adapted to the preferences of the individual actors and to the evolution of background knowledge and competences.

Flexibility can refer both to the whole process (i.e., to the flow of the different activities) as well as to single process activities, which actors might want to configure and personalize themselves. Flexibility can be achieved in several ways: in this paper we suggest to foster it by adapting processes according to information which is not available at definition time or can vary over time, such as data stored in a user profile. We propose a solution based on the integration of a Web platform for flexible process definition and execution, COOPER [6], and a policy engine, PROTUNE [3,4], which enables adaptivity rules to be enacted during process execution based on context information. The resulting framework is *open* in its nature, since it can integrate any (external) source of data. In particular, we show how our framework can exploit user profiles available on the Web (e.g., the ones provided by social applications) to support process adaptivity.

1.1 Running Scenario

To highlight the requirements that motivated our work and to exemplify the concepts introduced in this paper, we consider a scenario involving a process commonly enforced by employees of the fictitious Computer & Communication Research Center (C3R in the following). C3R employees are assigned to projects which are managed by project leaders. Each C3R employee is allowed to attend up to a given number of conferences per year. This number can be exceeded upon project leader's approval. Finally, for each conference the registration fee and the travelling expenses cannot exceed a given amount altogether.

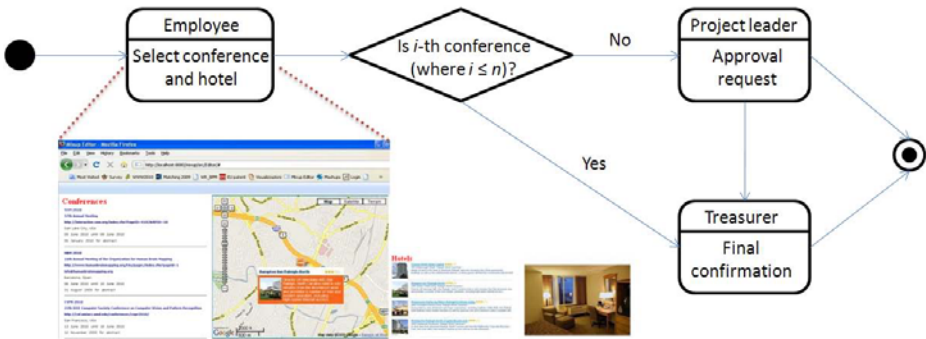


Fig. 1. Workflow of our running scenario

Fig. 1 shows a workflow corresponding to our scenario: whenever an employee wants to attend some conference, she starts the workflow and enters the first activity. She is thus presented with a subset of the conferences indexed by DB-World¹, which are relevant to the project she works in and her research preferences. As soon as the employee selects a conference, a Google map² shows its location together with hotels close to it. Only hotels are displayed such that the price of the overnight stay throughout the conference together with the conference registration fee does not exceed the allowed amount. Finally, as soon as a hotel is selected, its images available on Flickr³ are automatically displayed.

After the employee is done with her activity, the control flow comes either to the treasurer or to the project leader. The first case happens if and only if in the current year the employee did not attend the maximum allowed number of conferences n yet. In the second case the project leader must be asked, who can in turn either deny the request (thereby terminating the workflow) or accept it, in which case the workflow falls back to asking the treasurer for the final confirmation. Upon conclusion, an approval or denial message is sent to the employee.

¹ <http://www.cs.wisc.edu/dbworld/>

² <http://maps.google.com/>

³ <http://www.flickr.com/>

The scenario we just described shows the advantages adaptive processes could bring to daily life. On the one hand, *intra-activity personalization* would allow to configure single process activities according to the user needs and preferences (e.g., by showing a C₃R employee only relevant conferences and only hotels satisfying her budget constraints). On the other hand, *inter-activity personalization* would allow to configure the process flow according to the user needs (e.g., by – not – asking the project leader to approve a request according to employee-related information). User-related information can be derived from user’s profiles, which can be both those managed locally at the user’s company (e.g., to store the number of attended conference and the available budget), or those available at external community-based Web applications (e.g., the user’s LinkedIn profile storing her skills and research topics).

This paper is organized as follows: Sections 2 and 3 introduce the technologies and tools our solution makes use of, namely, the COOPER process engine and the PROTUNE policy engine respectively. Section 4 describes our reference architecture. We report about related work in Section 5 and conclude in Section 6.

2 The COOPER Process Engine

COOPER is the tool we chose for the definition and execution of processes on the Web. The choice of COOPER is mainly due to its extension mechanisms, which ease the integration of external resources. The most remarkable feature of COOPER is indeed the possibility to integrate libraries of predefined *activity types*, i.e., definitions of activities that are regularly performed by users collaborating in specific domains. COOPER can be easily extended by delegating external modules (from now on *handlers*) to process activities of a given *type*. Since COOPER is a Web platform supporting the execution of processes on the Web, the definition of a new activity type implies associating an activity with a Web front-end, playing the role of the user interface during the execution of the activity, and with a handler in charge of managing the execution of the activity. At run-time, as soon as COOPER starts processing an activity, it checks its type and, if some handler is available for activities of that type, it hands the activity over to that handler. Input parameters and environmental information (like the current user and run-time data) are provided by COOPER to the handler and return parameters are provided back from the handler to COOPER. To some extent, control on the activity flow can be also delegated to handlers, since they can provide COOPER with the activities to be started after they return.

Libraries of activity types are one of the mechanisms used by COOPER to compose sound, “well-structured” processes [11]. In [6] we showed how COOPER’s mechanisms for process definition guarantee the semantically correct execution and termination of process instances. This result is especially important since COOPER allows end-users to act as process designers: in particular, they can define new processes by extending process templates or by composing new models from scratch. End-users can also modify template-based process definitions at run-time, as long as the template constraints the process might hold are not

violated. Such a flexibility is achieved because each process definition bears all metadata (process structure, role and resource assignments) and run-time data (state of the process and of the activities, execution timestamps) needed for the correct execution. On the other hand, the automatic enforcement of constraints helps (often unexperienced) users during the re-definition and evolution of running processes.

More recently, COOPER has also been extended to enable the definition of *mashup*-like activity types which reuse external services [9]. The mashup paradigm is offered both to process designers who need to introduce new activity types, and to end users who want to customize their activities at run-time by integrating external services which help them to accomplish their tasks. The integration of external services has been made possible by the addition of an activity type (*container activity*), whose handler is in charge of composing and coordinating the execution of the services. This feature further increases the extensibility of COOPER and greatly facilitated its integration with the policy engine and context data.

3 The Protune Policy Engine

According to [16]’s well-known definition, policies are “rules governing the choices in the behavior of a system”, i.e., statements which describe which decision the system must take or which actions it must perform according to specific circumstances. *Policy languages* are special-purpose programming languages which allow to specify policies, whereas *policy engines* are software components able to enforce policies expressed in some policy language.

PROTUNE is a framework for specifying and cooperatively enforcing security and privacy policies on the Semantic Web. Protune is based on Datalog and, as such, it is an LP-based policy language (cf. [8]). A PROTUNE program is basically a set of normal logic program *rules* [12] $A \leftarrow L_1, \dots, L_n$ where $n \geq 0$, A is an *atom* (called the *head* of the rule) and L_1, \dots, L_n (the *body* of the rule) are *literals*, i.e., $\forall i : 0 \leq i \leq n$ L_i equals either A_i or $\sim A_i$ for some atom A_i . Rules whose body is empty are called *facts*.

Given an atom $p(t_1, \dots, t_a)$ where $a \geq 0$, p and a are called the *name* and the *arity* respectively of the *predicate* exploited in the atom, whereas t_1, \dots, t_a are *terms*, i.e., either constants or variables.

With respect to Datalog, PROTUNE presents the following main differences.

- Policy language.** PROTUNE is a policy language and not a language for data retrieval
- Actions.** The evaluation of a literal might require to perform actions
- Objects.** PROTUNE supports objects, i.e., sets of (*attribute, value*) pairs linked to an identifier

Being PROTUNE a policy language, its application scenarios are essentially different than the ones of Datalog, which is a language for data retrieval: whoever

issues a Datalog query is automatically allowed to retrieve the requested information, whereas in general not everyone issuing a PROTUNE query is allowed to access the requested resource or service. For this reason, whilst the process of evaluating a Datalog query is single-step, the process of evaluating a PROTUNE query involves up to two steps: checking whether the query can be evaluated and, if this is the case, actually evaluating it.

The evaluation of a Datalog (and, more generally, of a Prolog) literal is based on SLDNF-Resolution [12]. Only PROTUNE *logical* literals are evaluated in such a way, whereas the evaluation of PROTUNE *provisional* literals requires to perform an action: the evaluation of a positive (resp. negative) provisional literal is successful if the execution of such action is (resp. is not) successful.

The biggest difference between Datalog and PROTUNE with respect to the built-in data types is that PROTUNE supports *objects*, i.e., sets of (*attribute, value*) pairs linked to an identifier. Attributes and values can be objects in turn and attributes can be multi-valued. Objects are referred to by their identifiers, whereas a generic value of the attribute *attr* of an object *id* is denoted by *id.attr*. Finally, a value *val* can be defined for the attribute *attr* of an object *id* by asserting the fact *id.attr = val*.

4 Integrated Architecture for Adaptive Processes

This section describes our reference architecture and in particular illustrates how it enables access to context data (Section 4.1), inter-activity adaptivity (Section 4.2) and intra-activity adaptivity (Section 4.3).

4.1 User Profiles

In order to reduce dependencies among components, we designed an architecture as loosely-coupled as possible. As a consequence, COOPER does not communicate with user profiles directly but only through the PROTUNE engine, which is responsible for each adaptivity issue. Notice that our aim is not to manage the construction of such profiles or provide mechanisms for their integration⁴. Rather, we provide a way to access, according to specific codified rules, attributes that are distributed across different profiles.

Fig. 2 is meant to show the interplay between the PROTUNE engine and the user profiles in our running scenario. It shows the user profiles of two social networking sites as well as the C₃R one. The last one is supposed to contain information like the role of C₃R employees, the projects they are assigned to and the number of conferences they already attended during the current year, as well as the maximum value traveling expenses can amount to for an employee, and the maximum number of conferences an employee can attend in one year.

Beside the policy engine and the user profiles, Fig. 2 also shows *wrappers*, one for each user profile, and the so called *Meta-wrapper*: wrappers provide clients

⁴ User profile integration is still an open issue; the few proposed approaches (see for example [2]) are not widely accepted yet.

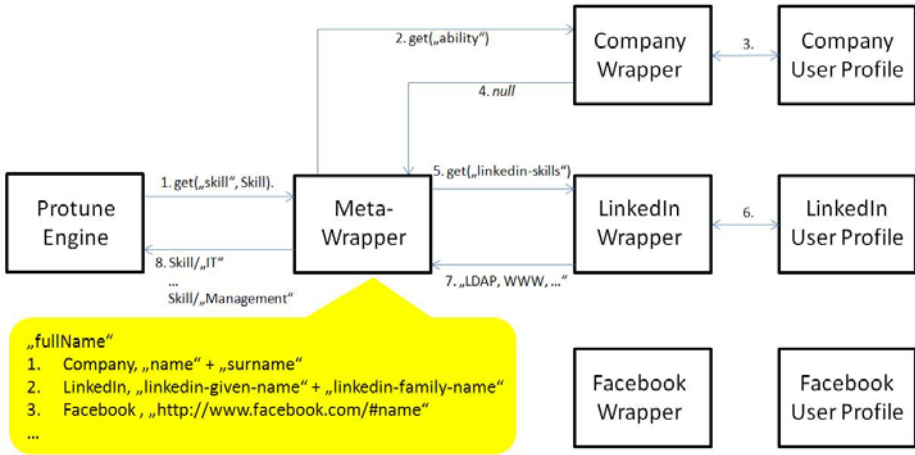


Fig. 2. Interplay between the PROTUNE engine and the user profiles

with an interface `get` which takes as input the (user profile-dependent) name of an attribute to be retrieved in a profile and returns the set of values of that attribute or `null` in case the value is undefined.

The *Meta-wrapper* is the key component responsible for performing the user-profile alignment which is needed in order to merge information coming from different sources. An example of user-profile alignment is provided by the cartoon in Fig. 2: whenever an incoming request asks for the value of the attribute (whose user profile-independent name is) `fullName`, *Meta-wrapper* first forwards the request to the C3R wrapper. If no result is returned (i.e., if the returned value is `null`), the LinkedIn wrapper is queried. If no result is returned, the Facebook wrapper is queried. The value returned by the *Meta-wrapper* is the concatenation of the (values of the) C3R user-profile attributes `name` and `surname` (assuming that they are available), otherwise it is the concatenation of the LinkedIn user-profile attributes `linkedin-given-name` and `linkedin-family-name` (assuming that they are available), otherwise it is the Facebook user-profile attributes <http://www.facebook.com/#name>.

As this example shows, the mapping implemented by the *Meta-wrapper* requires to specify for each user profile-independent attribute: (i) the ordering according to which the available user profiles have to be queried; (ii) the set of user profile-dependent attributes involved in the computation of the (value of the) user profile-independent one; (iii) how the user profile-dependent attributes have to be combined in order to compute the user profile-independent one.

Fig. 2 shows the steps involved in answering a given query issued by the PROTUNE engine to the *Meta-wrapper*. The interface provided by the latter is the PROTUNE-like predicate `get/2` whose first argument is the name of a user profile-independent attribute and whose second argument is (one of) its value(s). The query `get("skill", Skill)` asks for value(s) of the user profile-independent attribute `skill` (step 1). According to its built-in mapping, the *Meta-wrapper*

is aware of the fact that such user profile-independent attribute maps to: (i) the C3R attribute `ability`—each skill is described in a different `ability` attribute; and (ii) the LinkedIn attribute `linkedin-skills`—all skills are collected into a single comma-separated `linkedin-skills` attribute. Moreover, according to the built-in mapping, the C3R user profile must be queried before the LinkedIn one. For this reason the query `get("ability")` is issued to the C3R wrapper (step 2) and the C3R user profile is accessed (step 3). It happens that no skill information is available in it (step 4), therefore the query `get("linkedin-skills")` is issued to the LinkedIn wrapper (step 5) and the LinkedIn user profile is accessed (step 6). Since skill information is available in it, there is no need to query the Facebook wrapper. The Meta-wrapper works out the results (step 7) and eventually ends up by identifying a set of skills (among which IT and Management) which are returned one by one to the PROTUNE engine (step 8).

4.2 Inter-activity Adaptivity

As we mentioned in Section 4.1, in order to enforce inter-activity personalization, context data, and in particular user profiles, have to be taken into account by WFMSs whenever they are about to identify the next activity to be performed. To favor separation of concerns, it is advisable to uncouple user-profile management and control-flow management so that, whilst being the process engine still responsible for advancing the workflow to the next activity to be performed, the task of identifying the upcoming activity itself is delegated to an external component. This section describes an application of this approach to the COOPER.

As we mentioned in Section 2.2, COOPER's activities are typed and, as soon as COOPER identifies the type of the next activity to be performed, it delegates the handling of the activity to the component responsible for that activity type. New activity types can be defined and corresponding handlers can be registered at COOPER's, so that they will be invoked whenever needed.

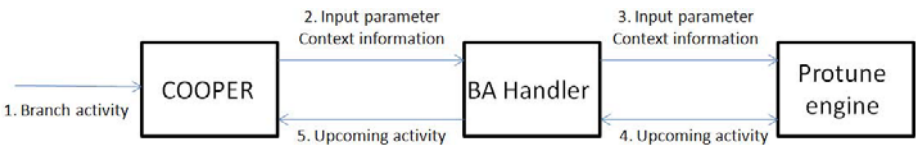


Fig. 3. Extension of COOPER by means of intra-activity policies

We exploited the COOPER's extension mechanism to achieve inter-activity adaptivity. We defined a new handler for branch activities⁵ (namely, *branch-activity handler* or *BA-handler*). Fig. 3 shows how a generic branch activity is processed by COOPER extended with our BA-handler. As soon as a branch activity comes in (step 1), it is redirected to the BA-handler, together with

⁵ Branching activities in COOPER are *system activities*, which can be used to specify the structure of the process, rather than the single work items. They distinguish from *user activities*, which are instead instantiations of activity types.

input parameters and context information (step 2). The BA-handler acts as a mediator and forwards such information to the PROTUNE engine (step 3). The PROTUNE engine evaluates its inter-activity policy in order to answer the BA-handler's question about the upcoming activity (step 4), which is eventually forwarded back to COOPER (step 5). Policy evaluation might require to use the information forwarded by the BA-handler as well as the one contained in the available user profiles, which are accessed as described in Section 4.

A generic inter-activity policy is a set of rules like the following.

```
allow(getNextActivity(Activity)) ←
...

```

Against query `getNextActivity(Activity)` issued by the BA-handler, the PROTUNE engine instantiates the variable `Activity` with the identifier of the activity to be performed next. The dots (...) represent conditions which have to be fulfilled in order for a given instantiation to take place and which might involve information forwarded by the BA-handler as well as user-profile information and, more generally, can be as expressive as described in Section 3.

For instance, the inter-activity policy mentioned in Section 4.1 might look like the following.

```
(1) allow(getNextActivity("a163")) ←
(2)   get("attendedConferences", AttendedConferences),
(3)   get("maximumNumber", MaximumNumber),
(4)   AttendedConferences ≤ MaximumNumber.

(5) allow(getNextActivity("a41")) ←
(6)   ~ allow(getNextActivity("a163")).
```

The rule at lines 1-4 states that the *Final confirmation* activity (whose identifier is supposed to be `a163`) will be performed next if the number of conferences the current employee already attended in the current year does not exceed the maximum allowed number (line 4). Both numbers are obtained by querying the Meta-wrapper according to the interface described in Section 4 (lines 2-3) and providing as input parameters the names of the metadata introduced in Section 4.1.

The rule at lines 5-6 states that the *Approval request* activity (whose identifier is supposed to be `a41`) will be performed next whenever the evaluation of the preceding rule is not successful, i.e., whenever the PROTUNE engine realizes that the *Final confirmation* activity should not be performed next. This is the case not only if the current employee already attended the maximum allowed number of conferences in the current year, but also if for some reason either of metadata `attendedConferences` and `maximumNumber` cannot be retrieved (i.e., if the evaluation of either of the literals at lines 2-3 is unsuccessful).

4.3 Intra-activity Adaptivity

For each user who has to carry out the activity, intra-activity adaptivity consists of configuring an activity according to the user profile on the basis of some specified policies. As mentioned in section 2, an activity type can also be instantiated through a mashup. If this is the case, the adaptivity policies could

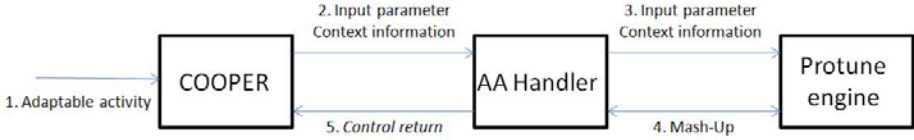


Fig. 4. Extension of COOPER by means of intra-activity policies

also be applied at a finer-grained level, addressing atomic services the mashup is composed of, to describe, for example, which services the mashup should consist of and how they should be configured.

As Fig. 4 shows, for intra-activity adaptivity we adopted a solution that is much similar to the approach we exploited for inter-activity adaptivity depicted in Fig. 3. We defined a new activity type (namely, *adaptable activity*) and a handler for it (namely, *adaptable-activity handler* or *AA-handler*). As soon as an adaptable activity comes in (step 1), it is redirected to the AA-handler, together with input parameters and context information (step 2). The AA-handler forwards such information to the PROTUNE engine (step 3), which uses (among else) such information in order to answer the AA-handler’s question about how the activity should be configured (step 4). Differently than the BA-handler described in Section 4.2, the AA-handler does not limit itself to forwarding such information back to COOPER, but is responsible for: (i) configuring the activity based on the information provided by the PROTUNE engine; (ii) coordinating the interaction with the user; and in particular (iii) identifying when the user completed the activity. At this point the control is returned to COOPER (step 5).

A generic intra-activity policy is a set of rules like the following.

```

(1) allow(getConten(Content)) ←
(2)   ...

(3) content.name: value ←
(4)   ...
  
```

Against query `getConten(Content)` issued by the AA-handler, the PROTUNE engine instantiates the variable `Content` with the information needed to configure the activity. Whilst in Section 4.2 the information to be returned (namely, the identifier of the activity to be performed next) could be modeled as an atomic entity (specifically, a PROTUNE string), the configuration of an adaptable activity can be modeled in the most natural way as a structured object (specifically, a PROTUNE object – cf. Section 3). However, as described in Section 3, before returning a PROTUNE object, the PROTUNE engine collects all $(name, value)$ pairs of its, and the process is iterated in case either of $name$ and $value$ is an object in turn. For this reason, before returning the PROTUNE object identified by the label `content`, the PROTUNE engine collects all the pairs it consists of, i.e., all pairs $(name, value)$ such that the PROTUNE atom `content.name: value` holds. Notice that this requires to evaluate conditions in line 4, which do not conceptually differ from the ones in line 2 or the ones described in Section 4.2.

For instance, (a fragment of) the intra-activity policy mentioned in Section 1.1 might look like the following.

```
(1)  allow(getContent(mashUp)).
(2)  mashUp.“application”:dbWorld ←
(3)    currentActivity(“a532”).
(4)  mashUp.“application”:googleMaps ←
(5)    currentActivity(“a532”).
(6)  mashUp.“application”:flickr ←
(7)    currentActivity(“a532”).

(8)  dbWorld.“conference”:Conference ←
(9)    get(“skill”, Skill),
(10)   getConferenceByTopic(Skill, Conference).
(11) dbWorld.“conference”:Conference ←
(12)   ~ get(“skill”, Skill),
(13)   getConference(Conference).
...

```

This policy makes use of the following provisional predicates (cf. Section 3).

currentActivity/1. Part of the interface to the context information available to the AA-handler: it checks the identifier of the current activity (lines 3, 5, 7)

get/2. The interface to the Meta-wrapper described in Section 4 (lines 9, 12)

getConferenceByTopic/2. Part of the interface to the DBWorld RSS feeds: it retrieves the conferences related to a given topic (line 10)

getConference/1. Part of the interface to the DBWorld APIs: it retrieves the conferences listed on DBWorld (line 13)

The rules at lines 2-7 state that, if the current activity is *Select conference and hotel* (whose identifier is supposed to be `a532`), it will show a mashup consisting of the following applications: DBWorld (represented by the PROTUNE object `dbWorld` – line 2), Google Maps (represented by `googleMaps` – line 4) and Flickr (represented by `flickr` – line 6). The rules at lines 8-13 then configure the DBWorld application by listing the conferences it should show (represented by the values of the `conference` attribute of `dbWorld` – lines 8, 11). In particular, the rule at lines 8-10 states that, if the skills of the current employee are available (line 9), only conferences relevant to them should be shown (line 10). On the other hand, the rule at lines 11-13 states that, if the skills are not available (line 12), all conferences should be shown (line 13).

Notice that the adaptation of component services (e.g., DBWorld) within a mashup-based instantiated activity is possible thanks to the availability of *mashup descriptors*, specifying both the characteristics of single components (e.g., the way they can be accessed), both the way they are integrated and executed in the mashup. For more details on such descriptors and the mashup execution logic they are able to support the reader is referred to 17.

5 Related Work

In order to support changing and non-repetitive processes, which are typical in collaboration scenarios, the main efforts have been done in the field of communication support systems or *Groupware* (see 13 for a survey). These applications

support the execution of individual tasks, especially based on asynchronous activities (e.g., sending emails), but offer very limited support to structuring tasks into process flows sustaining collaboration. WfMSs can in principle support the design of collaborative processes. However, they are too rigid to support the variable nature of such processes [7].

Positioned between the two extremes are *evolving workflows*. The authors of [15] identify *adaptability* as a dimension characterizing workflow evolution, and define it as the ability of a process to react to exceptional situations. In literature, this dimension is mainly addressed by mechanisms for exception handling that extend workflow definition languages with new and more complex control constructs [1]. The main disadvantage of such approaches is that the new languages are proprietary solutions lacking portability and whose execution engines are complex to implement. A general drawback of this paradigm is also the complexity of foreseeing at design time the whole set of exceptions that could occur at execution time [10]. Since the adaptation logic is built in the process design, dealing with non planned exceptions is not trivial: the process specification must be changed, and this in turn compromises the compliance of the already active workflow instances, enacted according to the original model, with the new process specification [5,14].

The approach proposed in this paper tries to overcome the drawbacks mentioned above by targeting *adaptivity*, i.e., the capability of a process to react at run-time in response to context parameters that cannot be adequately fixed in the workflow definition, such as properties of (evolving) user profiles. To the best of our knowledge, this issue has been scarcely investigated in literature, being process adaptivity often confused with process flexibility and adaptability [10]. The solution described in this paper allows us to introduce adaptivity features without affecting the process models. Moreover, context data and adaptivity policies can be managed separately from the process deployment, thereby easing the evolution of adaptivity requirements as well as of the process itself.

6 Conclusions and Further Work

This paper has presented an integrated framework where flexibility of Web collaborative processes is enhanced through run-time adaptivity with respect to context data. Collaborative processes are highly characterized by the need of adapting the process and the execution of single tasks to the variability of the needs, preferences and background of the involved actors. This need is even more accentuated if we consider that several sources of context data, and in particular user profiles, are available on the Web and can be exploited as a solid basis to provide adaptivity. We have therefore experimented the integration of the COOPER process engine, initially conceived to support flexible and dynamically defined collaborative processes on the Web, with an external policy manager, PROTUNE, in charge of accessing heterogenous sources of context data and executing adaptivity policies. Although we adopted such two specific platforms, the integration approach described in this paper is general in its nature, and can be

easily replicated in the context of different methodologies for flexible processes and policy management. The integration logics, indeed, is mainly based on the adoption of mediators, the so-called *handlers*, in charge of managing the interaction between the process engine and the policy engine, as well as the access to heterogeneous sources of context data.

An initial prototype has allowed us to prove the feasibility of our approach. Our future work is devoted to fully implementing the integration, and to testing its effectiveness and efficiency. This activity will also imply the definition of pre-defined *adaptive activity types* and of related adaptivity policies, as well as of policy-enhanced process templates covering the most frequent adaptivity requirements. We will also investigate the possibility that the users personalize the *Meta-wrapper*, for example giving them the freedom to select their most preferred user profiles and priorities for each profile.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. *Distributed and Parallel Databases* 14(1), 5–51 (2003)
2. Abel, F., Heckmann, D., Herder, E., Hidders, J., Houben, G.J., Krause, D., Leonardi, E., van der Sluis, K.: A framework for flexible user profile mashups. In: AP-WEB, pp. 1–10 (2009)
3. Bonatti, P.A., Olmedilla, D.: Driving and monitoring provisional trust negotiation with metapolicies. In: POLICY, pp. 14–23 (2005)
4. Bonatti, P.A., Olmedilla, D., Peer, J.: Advanced policy explanations on the web. In: ECAI, pp. 200–204 (2006)
5. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. *Data Knowl. Eng.* 24(3), 211–238 (1998)
6. Ceri, S., Daniel, F., Matera, M., Raffio, A.: Providing flexible process support to project-centered learning. *IEEE Trans. Knowl. Data Eng.* 21(6), 894–909 (2009)
7. Charoy, F., Guabtni, A., Faura, M.V.: A dynamic workflow management system for coordination of cooperative activities. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 205–216. Springer, Heidelberg (2006)
8. De Coi, J.L., Olmedilla, D.: A review of trust management, security and privacy policy languages. In: *CRYPTO*, pp. 483–490 (2008)
9. Fisichella, M., Matera, M.: Process Flexibility through Customizable Activity Types: a Mashup-based Approach. Submitted for publication (2010)
10. Kammer, P.J., Bolcer, G.A., Taylor, R.N., Hitomi, A.S., Bergman, M.: Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work* 9(3/4), 269–292 (2000)
11. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.: On structured workflow modelling. In: Wangler, B., Bergman, L.D. (eds.) *CAiSE 2000*. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000)
12. Lloyd, J.W.: *Foundations of Logic Programming*, 2nd edn. Springer, Heidelberg (1987)
13. Rama, J., Bishop, J.: A survey and comparison of CSCW groupware applications. In: *SAICSIT 2006*, pp. 198–205. Republic of South Africa (2006)

14. Sadiq, S.W.: Handling dynamic schema change in process models. In: Australasian Database Conference, pp. 120–126 (2000)
15. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. *Inf. Syst.* 30(5), 349–378 (2005)
16. Sloman, M.: Policy driven management for distributed systems. *J. Network Syst. Manage.* 2(4) (1994)
17. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) *WWW*, pp. 923–932. ACM, New York (2007)

Transformation of the Common Information Model to OWL

Andreas Textor^{1,2}, Jeanne Stynes², and Reinhold Kroeger¹

¹ RheinMain University of Applied Sciences
Distributed Systems Lab

Kurt-Schumacher-Ring 18, D-65197 Wiesbaden, Germany
{andreas.textor,reinhold.kroeger}@hs-rm.de

² Cork Institute of Technology
Department of Computing

Rossa Avenue, Bishopstown, Cork, Ireland
jeanne.stynes@cit.ie

Abstract. Managing an IT environment requires the exchange of structured data between different agents. The Common Information Model (CIM) is a comprehensive open standard that specifies how managed elements in an IT environment are modelled as a set of common objects and relationships between them. It has however limited support for knowledge interoperability and aggregation, as well as reasoning. By converting the existing CIM model into a format that can be processed by semantic web tools, these limitations can be overcome. This paper describes how CIM can be converted into a Web Ontology Language (OWL) ontology including constructs for which no obvious direct conversion exists, such as CIM qualifiers.

1 Introduction

With the constantly growing size and complexity of IT environments, comprehensive management systems that can effectively and efficiently manage these environments, are essential. A number of commercial and free systems exist that cover various parts of the required feature set for such management tasks. Usually, the management system is not comprised by a single tool, but by a set of different tools. To provide a unified view on the environment and allow interoperability between multiple management tools and managed elements, several integrated network management models were developed. Notable examples are the OSI network management model (also known as CMIP, the name of its protocol) and the still widely used simple network management model (SNMP). A more recent approach to specify a comprehensive IT management model is the Common Information Model (CIM) which is described in more detail in section 3. This widely recognized Distributed Management Task Force (DMTF) standard allows consistent management of managed elements in an IT environment. CIM is actively used; for example, the storage standard SMI-S (Storage Management Initiative Specification) of the SNIA (Storage Networking Industry Association, [1]) is based on CIM.

To establish interoperability mechanisms between the heterogenous integrated management models, mappings between different types of models can be defined. However, one question arises that can not be easily answered: “What happens when two different domains represent the same concept in a different way? A merely syntactic translation from the source model will not give the existing concept in the destination” [2]. This problem is known as *Semantic Matching* or *Ontology Matching* (see e.g. [3]). The problem can be approached by using ontologies to clearly define the semantics. The long term goal is to perform IT management based on a semantic foundation and a comprehensive domain model with the ability to model management rules in this model.

An IT management ontology can not only be used to clearly define the semantics of the managed elements in the managed system, but can also be used when describing associations of the managed system to adjacent domains. For example, if processes defined in an ontology for semantic business process management (see e.g. [4]) refer to physical or logical IT systems, such references can be directly and unambiguously expressed in the particular ontology. With the increasing development of semantic web technologies, including the specification of the Web Ontology Language (OWL) and numerous publications regarding merging and mapping of ontologies, querying, distributed reasoning etc., an ontology-based IT management approach can probably profit even more.

One aspect that is common to most recent approaches of applying semantic web technologies to the domain of IT management is the use of the Common Information Model as the domain model. To allow for semantic interoperability in IT management, a corresponding management ontology is required, and in [5], CIM has been proposed for this purpose. As [6] points out, CIM is usable for inferring properties about distributed systems, but is a semi-formal ontology with limited support for knowledge interoperability and aggregation, as well as reasoning. To create an IT management domain model that overcomes these shortcomings, a conversion of CIM to OWL is desirable. OWL can be used with reasoners, and can be augmented with rules formulated in the Semantic Web Rule Language (SWRL).

This paper presents how a conversion from CIM to OWL can be performed. Section 2 describes existing approaches to a conversion, section 3 gives an overview of the Common Information Model. Sections 5 and 6 describe the conversion of structural and additional CIM elements to OWL, respectively. Section 7 explains details of the implementation and examines the properties of the resulting ontology. The paper closes with a conclusion in section 8.

2 Related Work

The idea of applying semantic web technologies to the domain of IT management has been examined in several publications, e.g. [7,8]. Although CIM is often proposed for this purpose, the conversion of the native format in which CIM is specified into an ontology is not trivial. Several publications exist that attempt to create a conversion of CIM to OWL. In [6] the authors compare possible

conversions of CIM to RDFS (Resource Description Framework Schema) and to OWL. They find that RDFS is unsuitable to express CIM as it “does not provide constructs for expressing cardinality restrictions such as those used for describing association or aggregation relationships between CIM classes”. Also, some CIM qualifiers can not be adequately expressed in RDFS. They go on to construct an OWL-based ontology for CIM by using a previously defined mapping of UML to DAML+OIL (which is the predecessor to OWL) and create a mapping of CIM to UML. The conversion of structural information (classes and properties) can be expressed, but most CIM concepts (e.g. CIM qualifiers) have no mappings to either UML constructs or OWL constructs, so the resulting ontology lacks a large part of the information that is provided in the original model.

In [9] the author also addresses the conversion of CIM to OWL, but several decisions in the conversion prevent a complete or near-complete conversion: Properties of CIM classes are mapped to OWL datatype properties, which prevents any property that has the type of another CIM class (object properties must be employed for this to be possible). The author addresses the issue of CIM properties that are explicitly scoped to the corresponding class, but in OWL have global scope, by using the OWL *allValuesFrom* restriction. This makes it impossible to specify properties for multiple classes that have the same name but different ranges. Apart from that, a lot of CIM constructs and qualifiers are not considered at all in the conversion, such as Aggregation/Composition, Value/ValueMap, Keys, etc.

A more complete conversion approach from CIM to OWL is described in [10]. The authors introduce a meta-ontology that is used to model the CIM constructs that have no direct OWL correspondence, e.g. default values or qualifiers that set a value read-only or write-only. However, they do not describe how the meta-ontology is constructed. The authors handle more of the CIM constructs than the previously described approaches, such as several qualifiers, but do not describe how more complex elements, such as CIM methods, can be converted. Also, they leave out several of the restrictions the CIM defines, e.g. data types of properties are lost and in their approach it is generally not possible to support constraints for properties, such as `MaxLen` for strings.

3 DMTF CIM Overview

This section briefly describes the basic properties of the Common Information Model. CIM defines how managed elements in an IT environment are represented as a set of objects and relationships between them. The model is intended to allow a consistent management of these managed elements, independent of their manufacturer or provider. Unlike SMI (Structure of Management Information, the model underlying the popular SNMP protocol), CIM is an object-oriented model. CIM consists of

- A basic information model called the *meta schema*. The meta schema is defined using the Unified Modeling Language (UML, [11]).

- A syntax for the description of management objects called the *Managed Object Format (MOF)*.
- Two layers of generic management object classes called *Core Model* and *Common Model*.

Figure 1 shows the CIM meta schema definition in UML, as shown in the CIM specification [12]. The meta schema specifies most of the elements that are common in object-oriented modelling, namely

- *Classes, Properties* and *Methods*. The class hierarchy supports single inheritance (generalization) and overloading of methods. For methods, the CIM schema specifies only the prototypes of methods, not the implementation.
- *References* are a special kind of property that point to classes.
- *Qualifiers* are used to set additional characteristics of Named Elements, e.g. possible access rules for properties (**READ**, **WRITE**), marking a property as a key for instances (**Key**) or marking a class as one that can not be instantiated. Qualifiers can be compared to Java annotations; some qualifiers also have parameters.
- *Associations* are classes that are used to describe a relation between two classes. They usually contain two references.
- *Triggers* represent a state change (such as create, delete, update, or access) of a class instance, and update or access of a property.
- *Indications* are objects created as a result of a trigger. Instances of this class represent concrete events.
- *Schemas* group elements for administrative purposes (e.g. naming).

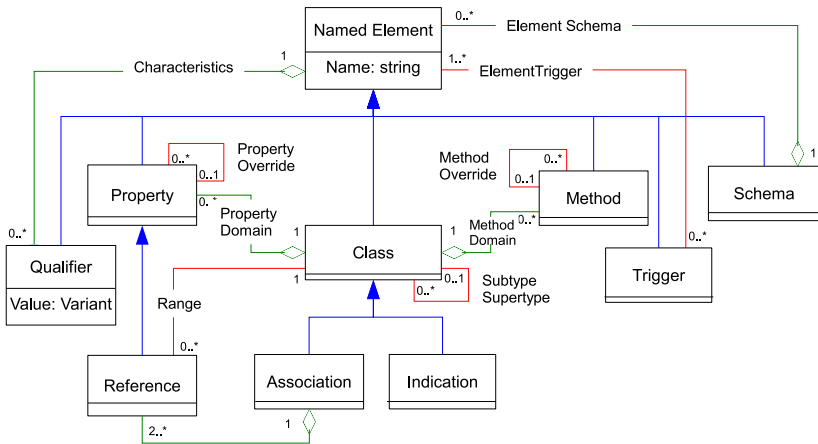


Fig. 1. CIM Meta Schema

Properties, references, parameters and methods (method return values) have a data type. Datatypes that are supported by CIM include {s,u}int{8,16,32,64} (e.g. uint8 or sint32), real{32,64}, string, boolean, datetime and strongly typed references (<classname> ref).

4 Translation Approach

For the construction of an OWL ontology, CIM schema elements have to be translated to OWL. In this case, the goal is an OWL 1 ontology. Some elements can be translated in a straightforward way, while other elements, especially qualifiers, can not be directly translated. To translate CIM constructs for which no direct translation exists, [10] proposes the use of a *CIM meta ontology*, although without describing it in detail. In our approach, we will create an ontology that consists of two parts: One part is the CIM meta ontology, which is statically modelled (i.e. manually) and which consists of super classes, properties and annotations that meta-model CIM constructs which can not be directly translated to OWL. The meta ontology has the namespace `meta`. The second part is the CIM schema ontology, which is modelled using OWL-, RDFS- and CIM meta constructs, and which represents the actual CIM model. It is automatically created by parsing and translating the MOF representation of the CIM schema, and has the namespace `cim`.

The translation of elements was performed incrementally. First of all, the basic CIM elements were translated to OWL constructs, such as classes. Possibilities for the translation of other elements were examined, and where possible, the corresponding OWL construct was used. In some cases a certain translation seems obvious, but isn't actually usable, e.g. using datatype properties to represent CIM properties where object properties are necessary. CIM concepts that have no corresponding OWL construct were modelled in the meta ontology, so that elements in the CIM schema part of the ontology can make use of these concepts. For the translation of the qualifiers, it was sometimes necessary to alter the structure of the model to be able to properly represent the qualifier; other qualifiers were modelled in the meta ontology.

When ontology elements such as classes and properties for corresponding CIM elements are created, the original name is retained. When "helper" elements are created (e.g. elements that do not exist in this form in the CIM schema), the identifiers of the original CIM class are suffixed with a double underscore "`_`", followed by the helper element name; further elements are concatenated with single underscores.

5 Translation of Structural Elements

The most basic element of the translation is a CIM class, which can be directly translated to `owl:Class`. Likewise, generalization (inheritance) can be expressed using the OWL subclass concept `rdfs:subClassOf`. Apart from generalization, CIM has another basic construct to express relationships between classes, the *Association*. An association is a special kind of a class describing a link between other classes. Associations are essentially normal classes with two typed reference properties, and are marked with the `Association` qualifier.

An *Aggregation* is a specialization of an association, where the two references are explicitly marked as parent and child (the class has both the `Association`

and **Aggregation** qualifiers, and the parent reference is also tagged with the **Associated** qualifier). The aggregation can be specialized even further using the **Composition** qualifier, which adds the semantics of a whole-part/compositional relationship to distinguish it from a collection or basic aggregation. None of the three constructs – association, aggregation and composition – can be directly translated into OWL. Using object properties for representing them is not possible, as association classes can inherit from other classes, but OWL object properties can not inherit from OWL classes. In order to model these constructs, corresponding classes and object properties are modelled in the CIM meta ontology.

The class **CIM_Association** represents an association and is the domain of the object property **CIM_Association_Role**, from which concrete CIM association object properties can inherit. Each concrete association has two association roles (i.e. two instances of the object property). **CIM_Aggregation** is modelled as a subclass of **CIM_Association** and has two object properties which represent the parent and the child part of the association, respectively, and which are subproperties of **CIM_Association_Role**. The **Composition** is modelled analogously as a subclass of the aggregation class and the parent and child subproperties of the aggregation parent and child object properties.

The modelling of CIM properties proves to be not straightforward either. While the structural element of a property could be modelled by a simple object property, this approach is not applicable when taking into account that properties can have qualifiers that make assertions about the property. A simple example is the **MaxLen** qualifier that can be used with properties of type **string** to assert a maximum length for its value. Qualifiers that make assertions can, in some cases, be represented by OWL property restrictions, which constrain the range of a property in specific contexts in a variety of ways. One important case where this is not possible is the combination of **Values** and **ValueMap** qualifiers: The **ValueMap** qualifier defines the set of permissible values for a property. When it is used in combination with the **Values** qualifier, the location of the value in the **ValueMap** array determines the location of the corresponding entry in the **Values** array. Listing 1 shows an example property from the CIM class **CIM_Job**.

```

1      [Write, Description ( "This property indicates whether the times "
2      "represented in the RunStartInterval and UntilTime properties "
3      "represent local times or UTC times. Time values are synchronized "
4      "worldwide by using the enumeration value 2, \"UTC Time\". ),
5      ValueMap { "1", "2" },
6      Values { "Local Time", "UTC Time" }]
7      uint16 LocalOrUtcTime;

```

Listing 1. ValueMap and Values qualifiers

To implement CIM properties in the OWL model, a structure as shown in figure 2 is used. In the figure, rectangular boxes represent OWL classes, rounded boxes with one connection represent datatype properties and rounded boxes with two connections represent object properties. Each CIM property is modelled as a combination of an OWL object property and an OWL class. This

class inherits from the meta ontology class `CIM_Value` that has two datatype properties, “value” and “valueMap”. To model the semantics from the original `ValueMap` and `Values` qualifiers, the class is restricted using `owl:oneOf` to allow only instances from a list of `CIM_Value` instances (one for each `ValueMap-Values` combination).

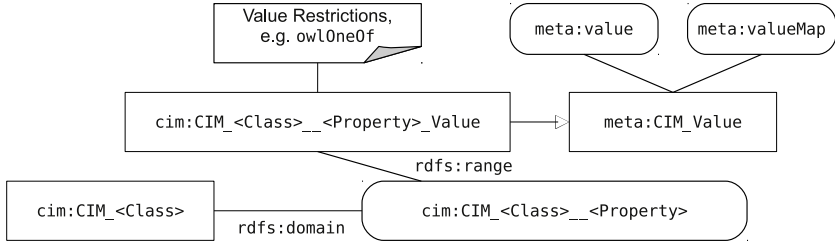


Fig. 2. Modelling of properties

For the example in listing [11](#), the following OWL elements are created: The class `CIM_Job_LocalOrUtcTime_Value` which is a subclass of `CIM_Value`, the object property `CIM_Job_LocalOrUtcTime` (with domain `CIM_Job` and range `CIM_Job_LocalOrUtcTime_Value`), the `CIM_Value` instance `CIM_Job_LocalOrUtcTime_Value_Local_Time` with data property `value` set to “Local Time” and data property `valueMap` set to “1”, likewise an instance for UTC Time, and finally, a subclass restriction on `CIM_Job` on the object property to limit all values to one of the two `CIM_Value` instances.

One important part that was left out so far, is the data type of the modelled property. To preserve the type information, the CIM primitive type is mapped to a corresponding XSD type, which is then added to the object property using the meta ontology annotation `type`. Another possibility would have been to create a subclass of `CIM_Value` for each primitive data type.

Each signed and unsigned number type is translated to its XSD equivalent, e.g. `uint8` becomes `xsd:unsignedByte`, `sint32` becomes `xsd:int`, `real32` becomes `xsd:float`, and so on. Translation is mostly straightforward, except for `char16`, which has to be translated into a string, and `datetime`, which has a corresponding XSD data type but which specifies a different lexical representation for the datetime string. While CIM uses the format “`yyymmddhhmmss.mmmmmmsutc`” (where “`mmmmmm`” is the number of microseconds, and “`utc`” is the offset from UTC in minutes), XSD uses the format “`yyyy-mm-ddThh:mm:ss`” (where T is the date/time separator; a decimal point and additional digits to increase the precision of fractional seconds can be added after the seconds part). Datetime values have to be converted accordingly. Default values that can be specified for CIM properties unfortunately can not be expressed in OWL directly. To translate the default values into the ontology, the annotation `defaultValue` is defined in the meta ontology and added to the corresponding object property in the same way as the type annotation is added. Whenever an

instance of a class that has as property with a default value is to be created, this annotation has to be taken into account.

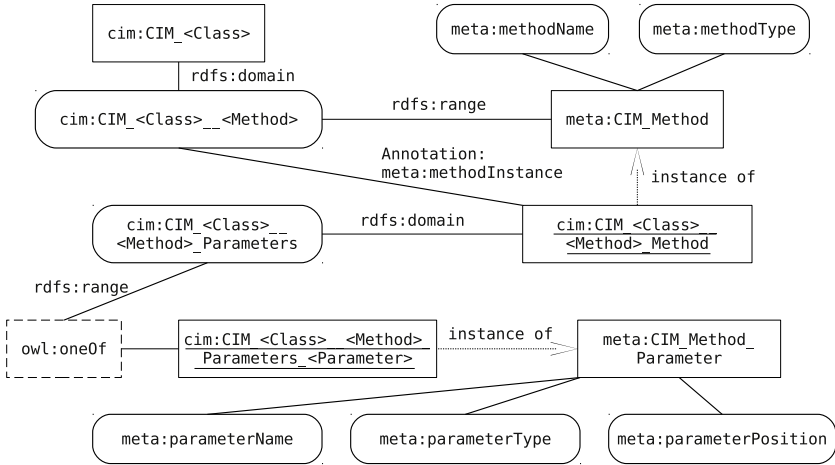


Fig. 3. Modelling of methods

The translation of CIM methods to OWL requires several OWL elements per method as well. Figure 3 shows how the translation is performed. A method basically has to provide the information about its name, type and its parameter list, where each parameter in turn has a name, a type and, in CIM, can have additional qualifiers. The meta ontology provides the class **CIM_Method**, that has two datatype properties, **methodName** and **methodType**. For each method of a CIM class, an OWL object property **CIM_<Class>_<Method>** is created, with the original CIM class as domain, and the **CIM_Method** class as range. An instance of the **CIM_Method** class is created and added as annotation to the object property. Generating both the object property and the instance is necessary, because in OWL, datatype properties can not be attached to object properties, but only to classes and instances. If the method overrides a method from a super class, a corresponding **rdfs:subPropertyOf** is added to the method object property.

Another object property is needed to associate the method instance with its parameters. Each method parameter is represented by an instance of the meta ontology class **CIM_Method_Parameter**, which in turn has datatype properties for its name, type and position. The position property is necessary, because OWL has no construct for ordered collections, and modelling a linked list would add unreasonable complexity. Note that although the OWL reference describes “Enumerated Datatypes” [13] that use **rdf:List** to define a list of values for a datatype property, a similar construct for OWL instances is not specified. The object property representing the parameter list has the method instance as domain and an **owl:oneOf** set of the method parameter instances as range.

6 Translation of Qualifiers

While section 5 described the translation of structural elements (classes, properties and methods), the second part to convert are the CIM qualifiers. Each qualifier can have a specific scope (e.g. can only be used in the context of classes, or in the context of properties and methods, etc.). A qualifier can also have parameters that have to be taken into account.

- *Values* and *ValueMap* are modelled in the meta ontology and were described in section 5.
- The *Override* qualifier can be used with properties and methods and indicates that the element in the derived class overrides the similar construct of the same name in the parent class in the inheritance tree. For the *Override* qualifier, an `rdfs:subPropertyOf` relationship is added to the object property that represents either the CIM property or the CIM method.
- The *Key* qualifier marks a property as the identifying property of the class. Keys are written once at object instantiation and are not modified thereafter. *Key* qualifiers on single properties can be translated by declaring the corresponding object property to be an instance of `owl:InverseFunctionalProperty`. If a property is declared to be inverse-functional, then the object of a property statement uniquely determines the subject (some individual). When more than one property in the source class is marked with the *Key* qualifier, they form a compound key. This poses the same problem as translating a SQL schema with composite primary keys to OWL (as proposed e.g. in [14]). One solution could be to create a synthetic class for the properties that comprise the key. This case is not handled by the converter.
- *Description* can be added to any element and provides a textual description of the element in human-readable format. It is converted into a `rdfs:comment`.
- The *Min* and *Max* qualifiers indicate the minimum and maximum cardinality of a reference property. They can be translated to `owl:minCardinality` and `owl:maxCardinality`.
- *MaxLen* is a qualifier that can only be attached to properties of type `string`, and specifies the maximum length of the string value. For this qualifier, an `owl:Restriction` is created for the class containing the property, as shown in the example in listing 2: using `owl:allValuesFrom` and `owl:withRestrictions`, the maximum length for the value can be specified as an XSD value restriction.
- *MaxValue* and *MinValue* specify the minimum and maximum values for `int` properties and can be translated analogously to the *MaxLen* qualifier, using `xsd:maxInclusive` and `xsd:minInclusive` datatype facets.
- The *Deprecated* qualifier marks an element as deprecated, and is converted using `owl:deprecatedClass` and `deprecatedProperty`, respectively.
- *Required* indicates that a non-NULL value is required for the property. It is translated by adding an `owl:minCardinality` of 1 to the object property.

- An *Alias* establishes an alternate name for a property or method and can be converted using the `owl:equivalentProperty` construct with the object property that represents the property or method.

```

1 <owl:Class rdf:about="#CIM_Account_CreationClassName_Value">
2   <rdfs:subClassOf rdf:resource="&meta;CIM_Value"/>
3   <rdfs:subClassOf>
4     <owl:Restriction>
5       <owl:onProperty rdf:resource="&meta;value"/>
6       <owl:allValuesFrom>
7         <rdf:Description>
8           <rdf:type rdf:resource="&rdfs;Datatype"/>
9           <owl:onDatatype rdf:resource="&xsd:string"/>
10          <owl:withRestrictions rdf:parseType="Collection">
11            <rdf:Description>
12              <xsd:maxLength
13                rdf:datatype="&xsd;integer">256</xsd:maxLength>
14            </rdf:Description>
15          </owl:withRestrictions>
16        </rdf:Description>
17      </owl:allValuesFrom>
18    </owl:Restriction>
19  </rdfs:subClassOf>
</owl:Class>

```

Listing 2. Conversion of MaxLen qualifier

- The *ModelCorrespondence* qualifier indicates a correspondence between two elements in the CIM schema. It is translated to the `rdfs:seeAlso` construct.
- *Read* and *Write* define that a property is readable or writable. As no OWL construct exists to represent this feature, they are modelled as annotations in the meta ontology, that are attached to the particular object property.
- *Version* provides the version number of a schema object; it is converted to `owl:versionInfo`.
- The *Abstract* qualifier indicates that a class is abstract and serves only as a base for new classes. A translation of this concept to OWL does not make much sense, as OWL classes can not be compared to classes from object-orientation in this respect. Therefore, a marker annotation `abstract` is modelled in the meta ontology and attached to the OWL class. This does not preserve the original semantics, but can be accounted for by tools that create instances of CIM classes.
- The *Units* and *PUnit* qualifiers provide information about the unit in which a property or method is expressed. While the (now deprecated) `Units` qualifier specifies a human-readable format (e.g. “Tenths of Decibels”), the `PUnit` qualifier uses a machine-readable format (e.g. “decibels*10⁻¹”). The qualifiers are converted using the meta ontology annotations `units` and `punit`, respectively.
- *UMLPackagePath* specifies the a position within a UML package hierarchy for a CIM class. A class hierarchy other than the inheritance hierarchy is not modelled in the ontology, so the package path is also modelled as an annotation in the meta ontology.
- The *ClassConstraint*, *PropertyConstraint* and *MethodConstraint* qualifiers can be used to specify OCL constraints for the particular elements. The

Object Constraint Language (OCL) is a declarative language for describing rules that apply to UML models. These qualifiers are currently not handled by the converter. One possibility to handle these qualifiers is to convert the OCL expressions to SWRL, as proposed in [15], which can then be included in the ontology.

Some more CIM qualifiers exist, which are not discussed in detail here. These qualifiers mostly serve as flags, for example the **Experimental** qualifier, and can be converted by creating appropriate annotations in the meta ontology.

7 Implementation and Resulting Ontology

The converter was prototypically implemented in Scala [16], a programming language that integrates object-oriented and functional features and that compiles to Java Byte Code. Java libraries can be used in Scala programs, and OWLAPI 2.2.0 was used for the creation of the ontology. The conversion is performed in two steps: The first step is parsing the MOF representation of the CIM schema. The parser was implemented as a combinatory parser using parser combinators that are part of Scala's standard library; no external parser generator was necessary. The second step consists of traversing the resulting abstract syntax tree and mapping elements to OWL axioms, as described in sections 5 and 6.

CIM schema version 2.23.0 consists of 1379 MOF files, containing roughly the same amount of OWL classes, and is converted to a total amount of 69295 OWL axioms, which are serialized into a 12 MB OWL file. The conversion takes about 35 seconds on an Intel Core 2 Duo with 2 GHz and 2 GB RAM; this can be interesting when a newer version of the CIM schema is to be translated. Loading the ontology using OWLAPI takes approximately 10 seconds and uses 130 MB on the same computer.

Only constructs that are valid in OWL DL are used; the resulting ontology has the expressiveness of OWL DL, which allows for decidable reasoning. As `owl:InverseFunctionalProperty` for the conversion of the **Key** qualifier is only used with object properties, the ontology does not require OWL Full expressiveness (the inverse functional characteristic for datatype properties can only be specified in OWL Full). Pellet 2.0.1 affirms the consistency of the ontology and the reasoning complexity of $\mathcal{ALUHOIN}^{(D)}$ (\mathcal{ALUE} is equivalent to \mathcal{S} , and $\mathcal{SHOIN}^{(D)}$ is the reasoning complexity of OWL DL).

8 Conclusion and Future Work

This paper presented a short introduction to the DMTF Common Information Model (CIM), and the motivation to convert CIM into the OWL format. It then explained in detail how CIM constructs such as classes, properties, methods and various qualifiers can be converted into OWL. The OWL ontology consists of two namespaces: `meta`, which contains manually modelled entities that represent CIM constructs for which no direct OWL translation exists, and `cim`, which

contains the actual OWL elements that model the CIM schema. The resulting ontology has the expressiveness of OWL DL. Future work includes the examination of reasoning performance in a real-world scenario using instance data and the possibility of a round-trip conversion (i.e. translating OWL back to CIM), as well as the application of the ontology in an IT management context using management rules formulated in SWRL.

References

1. Storage Networking Industry Association, <http://www.snia.org/>
2. De Vergara, J.E.L., Villagr a, V.A., Berrocal, J.: Semantic Management: advantages of using an ontology-based management information meta-model. In: Proceedings of the HP OpenView University Association Ninth Plenary Workshop (2002)
3. Shvaiko, P., Euzenat, J.: Ten Challenges for Ontology Matching. In: Proceedings of the 7th International Conference on Ontologies, DataBases, and Applications of Semantics, ODBASE (2008)
4. Hepp, M., Roman, D.: An Ontology Framework for Semantic Business Process Management. In: Proceedings of Wirtschaftsinformatik 2007, Karlsruhe (2007)
5. De Vergara, J.E.L., Villagr a, V.A., Asensio, J.I., Berrocal, J.: Ontologies: Giving Semantics to Network Management Models. IEEE Network 17 (May/June 2003)
6. Quirolgico, S., Assis, P., Westerinen, A., Baskey, M., Stokes, E.: Toward a Formal Common Information Model Ontology (2004)
7. De Vergara, J.E.L., Guerrero, A., Villagr a, V.A., Berrocal, J.: Ontology-Based Network Management: Study Cases and Lessons Learned. Journal of Network and Systems Management (2009)
8. Xu, H., Xiao, D.: A Common Ontology-based Intelligent Configuration Management Model for IP Network Devices. In: Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC). IEEE Computer Society, Los Alamitos (2006)
9. Heimbigner, D.: DMTF - CIM to OWL: A Case Study in Ontology Conversion. In: Ontology in Action Workshop in conjunction with the 2004 Conference on Software Engineering and Knowledge Engineering, SEKE 2004 (2004)
10. Majewska, M., Kryza, B., Kitowski, J.: Translation of Common Information Model to Web Ontology Language. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 414–417. Springer, Heidelberg (2007)
11. Object Management Group: Unified Modeling Language (UML), <http://uml.org/>
12. Distributed Management Task Force: Common Information Model (CIM), <http://www.dmtf.org/standards/cim/>
13. World Wide Web Consortium: OWL Web Ontology Language Reference, Enumerated Datatypes, <http://www.w3.org/TR/owl-ref/#EnumeratedDatatype>
14. Astrova, I.: Rules for Mapping SQL Relational Databases to OWL Ontologies. In: Metadata and Semantics, pp. 415–424. Springer, US (2009)
15. Milanovi c, M., Gaševi c, D., Giurca, A., Wagner, G.: On Interchanging Between OWL/SWRL and UML/OCL (2006)
16. The Scala Programming Language, <http://www.scala-lang.org/>

Improving Web Search Results for Homonyms by Suggesting Completions from an Ontology

Tian Tian¹, James Geller¹, and Soon Ae Chun²

¹ New Jersey Institute of Technology,
Newark, NJ, USA

{tt25,geller}@njit.edu

² CSI, City University of New York
Staten Island, NY, USA

{soon.chun}@csi.cuny.edu

Abstract. The terms that users pass to search engines are often ambiguous, by referring to homonyms. The search results in these cases are a mixture of links to documents that refer to different meanings of the search terms. Current search engines provide suggested query completion terms as a dropdown list. However, such lists are not well organized, mixing completions for different meanings. In addition, the suggested search phrases are not discriminating enough. We propose an approach to suggesting well organized and visually separated search completions based on ontologies. In addition, our approach supports the use of negative terms to disambiguate the suggested completions in the list. We present an algorithm to generate the suggested search completion terms. We have developed musician and basketball player ontologies and an Ontology-Supported Web Search (OSWS) System for “famous people” that generates the suggested search term completions, based on these ontologies.

Keywords: Ontology-supported Web search, semantic search methods, homonyms, negative search terms, suggested completions.

1 Introduction

Users’ information needs in the digital era can be fulfilled by keyword-based search engines. Such search engines have become the universal catalogs for world-wide resources. Unlike the old library catalogs that are mostly searchable by fixed fields (e.g., by authors, titles, and keywords predefined by authors), modern Web search engines provide a flexible, easy way to express search terms. However, the search results are typically long lists of hits that contain many irrelevant links [1]. Past research has concentrated either on refining the search keywords or on sifting and filtering the search results, to improve the precision of the returned hit lists [1].

Search engines face an additional complication when a search term is a homonym (a keyword with multiple meanings or multiple references) and the user is not aware that there are several concepts for this term. She might not be aware of this homonymy at all, or it might escape her attention at the moment of performing the Web search. For example, when looking for information about former President George W.

Bush she might momentarily forget about President George H. W. Bush, the father of President George W. Bush. She would then get results about both of them, which is not what she desired.

When using a search engine to satisfy an information need about a homonymous concept, a user is faced with two kinds of problems. She might get an overwhelming number of responses about one homonym, especially if this meaning is more popular, while the second homonym with a less popular meaning that she might be really interested in is hidden in a snippet on a much later page of hits, returned by the search engine. This is the case with lopsided preferences in meanings. For instance, the “Michael Jackson” who is a singer is much more popular than the basketball player of the same name. Hence many more search results contain references to the singer. In this situation, the user is at least aware that the results she is getting are not about the basketball player that she has been looking for. When formulating the initial query, it escaped her attention that there are two concepts for her search term and that more information might be available on the Web about the homonym that she is not interested in. At this point, she needs to wade through pages of reported hits for the wrong Michael Jackson or append terms to her query that will exclude the unwanted homonym and re-execute the search. This constitutes a kind of feedback loop between the user and the search engine.

The situation is even worse if the user is completely unaware of the fact that the search term is a homonym with two (or more) references, and all results that appear on the first few pages of hits are to the “wrong” reference. For example, a user located in the New York area, who types “Penn Station” into Google will see many references to Penn Station in New York City (NYC) and some references to Penn Station in Newark. These two Penn Stations are separated by a 20 minute train ride. Unbeknownst to her, there is also a Penn Station in Philadelphia, Pennsylvania. However, a reference to the latter does not appear on the first page of search results.

Google has a popular feature of displaying up to ten suggested search term completions for a query while the user is typing. Typing “George Bush” leads to suggested completions such as games, wiki, jokes, billboard, etc. The only small hint that there are two Presidents George Bush is one suggested completion “sr” (meaning senior). Google’s suggestions are presumably based on the most common search terms entered by its millions of users.

Our goal in this research is to improve the mechanism of suggested search completions in two ways. First, the display of suggested search term completions should be categorized visually to make it clear that homonymous terms exist. For this, knowledge of the classes that terms belong to is necessary. This is the kind of knowledge normally contained in ontologies. Secondly, the knowledge in the ontologies should be used to increase the precision of results, by making the suggested completions as discriminating as possible. One tool for making Web searches more focused is to use negative search terms in addition to the normal “positive” search terms. Naturally, the suggested search completions should not be over-specified to the point that the search engine would not return any results. As we do not have access to the “most common search terms” collected by commercial search engines, we cannot use them to generate suggested completions. Instead we use ontologies both for creating the suggested completions and for providing the knowledge needed to visually categorize them.

Including negative search terms in the search queries is a powerful tool for discriminating between wanted and unwanted results. In the past, negative search terms have not been used in suggested completions. We will discuss the generation of suggested completions with negative search terms and hint at the problems that arise out of this pursuit.

In Section 2 we present our previous work on ontology-supported Web search and explain some of its weaknesses. In Section 3.1 we describe an algorithm for generating suggested completions for homonymous search terms. Then, in Section 3.2, the Ontology-Supported Web Search (OSWS) System is introduced. Our approach to the use of negative search terms is discussed in Section 3.3. In Section 4, we briefly touch on the problem of ontology provisioning and present our musician and basketball player ontologies. Section 5 concludes the paper.

2 Background

In our previous research on an ontology-supported Web search system, the user was presented with a number of choices of additional search terms for her input. She could mark such terms as positive, i.e., they should be included in the Web search results, by clicking on associated check boxes (see Figure 1 and [2]). One problem with this approach was that users do not want to be bothered by (too many) questions. A more benign approach to eliciting additional information from a user can be seen in the use of suggested completions. While a user types in the first (few) word(s) of her search, the search engine displays up to ten suggested search completions, which will possibly describe the search that the user had in mind. These completions are presumably based on the observed frequencies of many searches of other search engine users [3]. While the user continues to type, the suggested completions change rapidly and are often limited to fewer than ten. Most major search engines have such a mechanism. Google calls them “query suggestions” appearing in the “search box” [3], Yahoo calls them “search assistant” [4], and Bing calls them “search suggestions” [5].

Another weakness of our approach in [2] was that it did not make use of the information that may be inferred by a form of closed-world assumption from the terms that

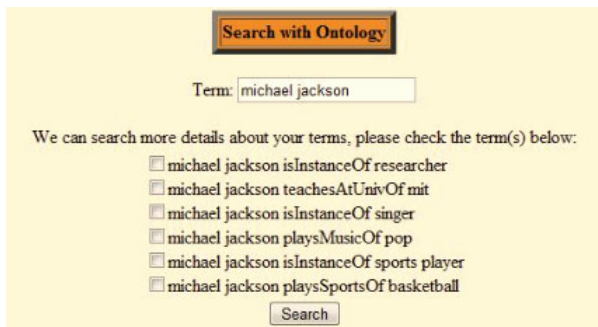


Fig. 1. Search screen example of previous ontology-supported Web search system

the user did *not* select with a check mark. According to the documentation of major search engines, the use of negative search words, marked with a minus sign before the word(s), constitutes a particularly powerful tool for discriminating between different results. Thus, we extend the approach in [2,6,7] in this paper by adding negative search terms.

Google's version of suggested completions has the problems described in Section 1. They do not reflect distinctions between different concepts that are expressed by the same word or the same multi-word term (homonyms). Suggested completions also do not appear to be optimized for discrimination between homonyms. Appending well chosen negative search words to a search term given by the user would result in improved discrimination between homonyms of that search term, if the appended words are characteristic for one of the homonymic senses. However, the use of too many negative search terms might exclude relevant results, i.e., the recall would suffer. It would be especially undesirable if the search is so over-specified that no results are reported at all.

The suggested additional search terms in [2] (see Figure 1) were derived from an ontology. For a given user input, all homonymous concepts were located in the ontology. Then choices of additional terms were generated by looking at neighboring concepts in the ontology. Thus "Michael Jackson" is categorized as a singer, or in more technical terms, Michael Jackson is an instance of the class "singer." Thus, a statement of this nature with a check box was suggested to the user. If the user checks this box, then the word "singer" was automatically appended to the Web search query before executing it.

Finding information about Michael Jackson the singer on the Web is clearly not a problem. There are millions of hits for this search term. However, when somebody is interested in Michael Jackson the basketball player, or in any one of the other over 20 Michael Jacksons that have achieved some kind of fame over the years, then the task of finding relevant information becomes much more difficult. Appending negative search words such as "-songs" and "-lyrics" makes this task easier, by excluding the most widely used homonym of Michael Jackson the singer.

3 Generating Ontology-Based Search Term Completions

3.1 An Algorithm for Suggested Completions with Positive Terms Only

The basic idea of generating suggested completions with *positive* search terms was not changed from [2], however the interface model was changed considerably, improving both on our previous work and on common search engines. The following pseudocode demonstrates the processing steps.

ALGORITHM *DISPLAY_SEARCH_SUGGESTIONS*

INPUT: *SEARCH_TERM*, *KNOWLEDGE_BASE*

OUTPUT: *Display of SEARCH_SUGGESTIONS*

BEGIN

NODE_COLLECTION = {}

FOR EACH *NODE* **IN** *KNOWLEDGE_BASE*

```

IF NODE contains SEARCH_TERM
    NODE_COLLECTION = NODE_COLLECTION U{NODE}
/* NODE_COLLECTION now contains all homonyms */
ITH_SUGGESTION = 1
IF size_of(NODE_COLLECTION) > 4
    NODE_COLLECTION = MOST_COMMON(NODE_COLLECTION)
/* NODE_COLLECTION now contains at most 4 homonyms */
FOR EACH NODE IN NODE_COLLECTION
    NEIGHBOR_LIST = {}
    FOR N IN NEIGHBORS_PLUS_GRANDPAR(NODE)
        /* We add one additional level in the IS-A
           hierarchy to the immediate neighbors. */
        NEIGHBOR_LIST = NEIGHBOR_LIST U {<REL, N>}
        /* Pairs of all neighbors and their connecting
           relationships are collected in a list. */
        PRIOR_LIST = PRIORITIZE(NEIGHBOR_LIST)
        /* Pairs with important relationships, such as
           IS-A are placed first in the list. */
        SEARCH_SUGGESTIONS[ITH_SUGGESTION] = PRIOR_LIST
        ITH_SUGGESTION++
    SEARCH_SUGGESTIONS = LIMIT_SIZE(SEARCH_SUGGESTIONS)
/* At most 12 lines are displayed over all
   homonyms. */
    DISPLAY_WITH_SEPARATORS(SEARCH_SUGGESTIONS)
/* Suggestions for each homonym are displayed,
   visually separated from each other. */
END

```

The algorithm *DISPLAY_SEARCH_SUGGESTIONS* uses the following sub-algorithms: *MOST_COMMON* returns at most four homonyms. The selection is done based on the number of hit counts for each homonym. These hit counts are recorded in the ontology during creation time. More details can be found in Section 3.2.

NEIGHBORS_PLUS_GRANDPAR returns for every instance in the ontology all neighboring nodes that are one link away from it, plus the “grand parent,” i.e., the IS-A parent of the class it is an instance of.

PRIORITIZE sorts the list of neighbors by importance. The importance is determined by the types of connecting relationships. Thus IS-A relationships to parent classes are considered more important than lateral semantic relationships. See Section 3.2 for more details on the importance of different relationship types. If several neighbors are connected by the same relationship type, then the order of the connected concepts is chosen arbitrarily.

LIMIT_SIZE controls the total size of the output. In order to avoid overloading the user with information and in order to achieve a behavior similar to existing search engines, the total number of search suggestions displayed is limited to at most 12. The number 12 is divisible by 2, 3, and 4, which makes it a good choice for 2, 3, or 4 homonyms.

Finally, the sub-algorithm *DISPLAY_WITH_SEPARATORS* creates the actual dropdown box that is shown to the user. It contains the computed search suggestions with appropriate separators to express the semantic distances between them.

Altogether the algorithm specifies the following behavior. A user types words of a search term into the search box. The algorithm locates nodes (classes or instances) in the stored ontologies that correspond to the input words. If only one node is located, then there is no problem with homonymy, at least according to the knowledge incorporated in the set of all loaded ontologies. On the other hand, if two (or more) nodes are located in the ontologies that match the user input, then additional processing is necessary.

For each located node, its neighbors¹ in the ontology network are retrieved, starting with the parent(s), if it is a class, or if it is an instance, the class that it is an instance of. Neighbors that are common to more than one sense (meaning) of the search term are eliminated, as they have no discriminatory power. The algorithm now appends subsets of these retrieved terms to the user terms to generate several suggested completions.

Knowledge from different domains is assumed to be stored in separate ontologies. However, when using this implemented knowledge, all ontologies are considered connected and combined into a single knowledge base.

Consider the following abstract example.

- The user types in two words *A B*, for example *A=Michael* and *B=Jackson*.
- The system identifies two concepts referred to as *A B*, let us call them *AB1* and *AB2*.
- *AB1* is an instance of *K*. *AB1* has a neighbor *L*.
- *AB2* is an instance of *M*. *AB2* has a neighbor *N*. The concepts *K*, *L*, *M* and *N* are distinct.
- The search engine generates the following suggested completions, three for *AB1* and three for *AB2*: *A B K*; *A B L*; *A B K L*; *A B M*; *A B N*; and *A B M N*.
- The total number of suggested completions is limited by a threshold and controlled by strict priorities in which order to select neighbors (Section 3.2).
- The suggested completions are presented to the user in a way that visually separates the *AB1* meaning from the *AB2* meaning, for example by using a bold line to separate them or by different background colors (see Section 3.2).

3.2 The Ontology-Supported Web Search (OSWS) System

The Ontology-Supported Web Search (OSWS) System for “famous people” provides search suggestions based on the user input, every time she types a new character. As seen in Figure 2, after the user completes the search term “Martina,” the system finds all the famous people in the knowledge base with “Martina” in their names. Additional background information about these famous people is extracted from the knowledge base for generating suggested completions. In this example, the tennis

¹ The immediate neighbors of a class are the following: parent classes (more general), child classes (more specific) and classes that are reachable from it by traversing a “semantic relationship.” The immediate neighbors of an instance are the class which the instance belongs to, and the object properties and the data type properties of the instance.

players Martina Hingis and Martina Navratilova and the singer Martina McBride are found. From the information related to these three famous people the suggested completions in the dropdown box are generated and displayed to the user.



Fig. 2. Interface of the OSWS System for search term "Martina"

In more detail, for each concept of a famous person of the same name, all immediate neighbors along with the connecting relationships are retrieved from the ontologies. In the OSWS System, the first proposed suggestion about a famous person is always based on the class (modeling the occupation) of the person, which defines the name of the domain that the person belongs to. For instance, Martina Hingis has the first suggested completion "Martina Hingis tennis player" and Martina McBride has the first suggested completion "Martina McBride singer."

Then the remaining suggestions about each famous person are constructed based on the knowledge retrieved from the ontologies. They may include the background information of a person like the date of birth and the place of birth, and sometimes the birth name. Besides, for musicians, the ontology stores the genres of music the artist performs. For sportsmen, the league and the team he or she belongs to are represented in the ontology. For instance, in Figure 2, from the suggested completions the user could learn that Martina McBride plays country music, adult contemporary music, and country pop music, which she may not have been aware of.

Different famous Martinas are separated by horizontal lines and background colors. This separation clearly expresses the fact that there are conceptual distances among the homonyms expressed by different sets of suggested completions. This makes it easier for the user to learn or remember that she is dealing with a homonym. Current search engines do not support such a separation. In fact, the visual display in the example expresses the fact that Martina Hingis is conceptually closer to Martina Navratilova (both tennis players) than to Martina McBride (the singer) by applying separating lines of different thickness.

Besides the separating lines, the background color design in the dropdown box also distinguishes famous people from different domains. In Figure 2, the suggestions for the two tennis players are generated by the system with a blue background, in contrast to the suggested completions of the singer that are displayed with a pink background.

Four preselected background colors are used for the at most four homonyms for which suggested completions may be displayed.

After the user chooses one suggestion that fits her search needs and clicks the “Google Search” button, she will be led to the result page of a “normal” Google search. Here we mimic the Google functionalities by having the “I’m Feeling Lucky” button, which will lead directly to the Web page with the highest Google ranking.

To avoid overwhelming the user with too many suggestions, and to stay close to the Google look and feel of the interface, we designed the system to show up to a maximum of 12 suggestions for a maximum of four famous people.

As noted in Section 3.1, there may be too many potential suggested continuations for one concept, and a selection process is required. The selection of lines for one homonym is achieved by assigning different priorities to different relationship types. For example, the IS-A link to the domain name (occupation) is considered to have the highest priority. For musicians, the genres of music they play have higher priorities than their dates of birth and places of birth. For basketball players, the team and league they play in are treated as more important than their birth information. Thus, we only show the high priority suggestions if there is more knowledge in an ontology than available space in the dropdown box. For example, in Figure 3, we show 12 suggestions in the search box by eliminating the date of birth and place of birth information of the singer Michael Jackson, since these have the lowest priorities.



Fig. 3. Interface of the OSWS System for search term "Michael Jackson"

If there are more than four homonyms (such as the over 20 Michael Jacksons) then the OSWS System selects up to four senses based on some criteria. There are two candidate approaches for this selection process. One possible selection criterion is the amount of information available in the ontologies about each sense. Thus, senses with a large amount of attached knowledge should be preferred over other senses. This is based on the pragmatic assumption that system implementers would not make the effort of including a large amount of information about a concept in an ontology if that concept is considered unimportant. However, this selection approach requires

mature ontologies covering many domains with rich knowledge. Unfortunately, such ontologies do not always exist, and it is still a big challenge to build them (Section 4).

In the absence of sufficiently complete ontologies a second approach is needed, which is the one used in the OSWS System. A possible criterion to select the most popular homonyms is by using the Google hit count estimates. We assume that people with higher hit count estimates are more popular and famous. For instance, the query “Michael Jackson singer” returns almost twice the number of Web pages than the query “Michael Jackson basketball.” Thus, Michael Jackson the singer should be preferred over the others. The representation of three homonymous “Michael Jacksons” in the knowledge base can be seen in Figure 4.

Classes in the figure are represented as boxes. Instances are shown as ellipses. IS-A links are drawn as arrows from the child class to the parent class. Dashed arrows connect instances to the classes that they are instances of. Finally, lines terminated by little black squares indicate semantic relationships other than IS-A and instance of relationships.

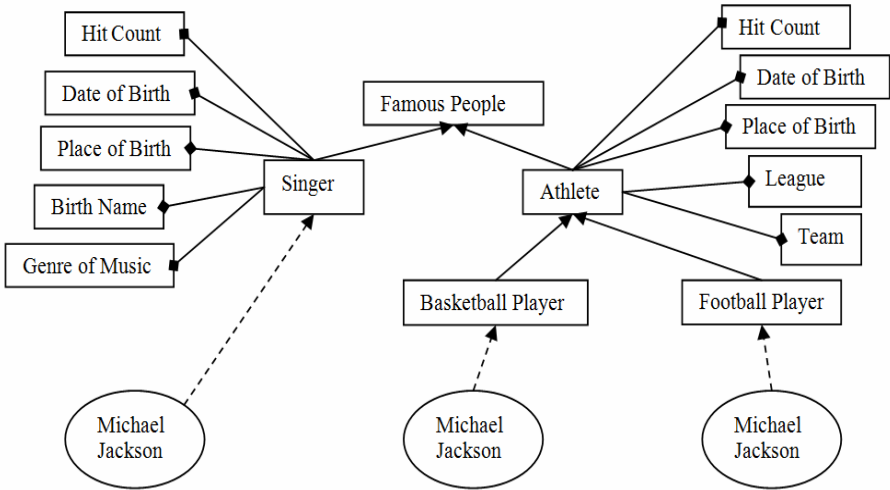


Fig. 4. Excerpt from the “famous people” knowledge base with homonym example “Michael Jackson”

We have collected the Google hit count estimates and assigned them to the appropriate instances of famous people while building the musician and basketball player ontologies. Thus, this information is available before the user starts with her search. However, this solution has several disadvantages. Hit counts are not stable. For example, after the singer Michael Jackson’s untimely death, the number of hits greatly increased. Thus the previously mentioned ontology-size-based criterion would be preferable.

The suggested completions in the search box of the OSWS System change dynamically after every single input character, just as in Google. The response time of

the OSWS System in the current implementation is near instantaneous, limited more by the typing speed of the user than by the response time of the system. The current ontologies in the system contain semantic information about more than 5000 musicians, more than 3000 basketball players and a sampling of sportsmen in other domains. Altogether, only three of the over 20 Michael Jacksons known to us are included in the ontologies. Clearly, with more and much larger ontologies in the OSWS knowledge base, the response time will degrade. More ontology building and experimentation is needed to investigate the effect of ontology size on response time.

3.3 Suggested Completions with Positive and Negative Search Words

In our previous work [2,6,7] we did not use negative search terms. The idea of using negative search terms is akin to mutual inhibition as it occurs in neural networks. If different neurons compete for achieving maximum activation, they inhibit neighboring neurons. This should be seen only as a metaphor, not as a technical model, as there are vast differences between the numeric approach of a neural network and the symbolic approach of an ontology. Based on this metaphor, if the user types in Michael Jackson and the ontology knows about Michael Jackson the singer and Michael Jackson the basketball player, then two useful suggested completions would be:

Michael Jackson Singer –Basketball
Michael Jackson Basketball –Singer

In both those suggested completions, we are using a bold font to indicate the words that have been entered by the user. Thus, neighbors of a node that are used as positive search terms for one homonym should be introduced as negative search terms for the other homonym. To our knowledge, none of the major existing search engines suggests completions with negative search terms to the users.

Many search engine users appear to be unfamiliar with the meaning of a minus sign (–) in front of a search word. Thus, suggesting a completion with a minus sign is syntactically unsatisfactory. Rather, the above completions need to appear as:

Michael Jackson Basketball [*but not*] Singer
Michael Jackson Singer [*but not*] Basketball

Using negative search words in suggested completions raises both conceptual and practical problems. One big practical problem that we have discovered in this research was that three major search engines do not process negative search terms as would be expected from their documentations or from a logical understanding of the meaning of “negative” words. This issue is, however, not central to the current paper. Results of this research are presented in [8].

4 Ontology Unde Venis?

(Ontology, from where do you come?) Probably the biggest problem with all ontology-based approaches is from where to take the necessary ontologies. Developing them in-house is time consuming and person-hour and/or budget intensive. Wide-scale ontology reuse has still not materialized, even though the Semantic Web [9], ontology search engines such as Swoogle [10] and ontology repositories [11,12] have

attempted to solve this problem. Many approaches to automatically generating or extending ontologies [2] have met with partial success but have also not reached the state of “shrink wrapped solutions.”

Outside of Medical Informatics, where huge terminological repositories are the norm, many ontologies are small. There appears to be little interest in building large, fact-oriented, regularly structured ontologies. Researchers prefer to focus on intricately structured, abstract “upper level ontologies” [13,14] or on rules and axioms. Building even one ontology with sufficient depth, breadth and domain coverage is a major challenge. Building ontologies just for the many existing categories of famous people in science, religion, art, history, politics, etc., and their many subcategories such as biology, chemistry, physics, computer science etc. in science, is a daunting task, even if we limit those ontologies mostly to simply-structured facts and instance categorizations.

Nevertheless, human intelligence relies on both rules and large numbers of simply-structured facts, including basic categorizations. For example, humans know about a large number of people how they are categorized, such that each one is known as a family member, friend, workmate, politician, sportsman or woman, somebody from “the history book,” a service provider (electrician, plumber, etc.), a teacher, student or classmate, etc., etc., or as completely unknown. We have, therefore, developed ontologies of musicians and basket ball players and are working on an ontology of sports stars to demonstrate the effectiveness of ontology-supported Web search.

The knowledge represented in these ontologies was mined from Wikipedia using a Web parsing program and stored in a temporary relational database. Not all information was available about every one of the over 5000 musicians.

A Protégé ontology was built, using the Protégé API, by extracting the mined data from the temporary database. One of the problems encountered in this process was the uniqueness requirement of Protégé. Thus, the city “Washington” and the state “Washington” could not both be represented by the same atom, and we had to append qualifiers to distinguish between them, in this case by appending the letters “DC” to the city.

To make our work available to the ontology community, we have submitted the musician ontology to Ontology Design Patterns (ODP) as an exemplary ontology: http://ontologydesignpatterns.org/wiki/Ontology:Musician_Ontology. In the future, we intend to publicize the sports star ontology in the same manner.

5 Conclusions

We have described an algorithm and its implementation in the OSWS System that improves both our own previous work on ontology-supported Web search and on the method that common search engines use for suggesting completions of user search terms for cases of homonyms. Our interface clearly separates between suggested completions for up to four homonymous concepts that fit the search terms that a user has already typed in. Furthermore, suggested completions in our interface may contain positive and negative search words. We have developed a knowledge base consisting of ontologies of musicians and basketball players used in the OSWS System and are currently working on a sports star ontology that covers additional popular disciplines.

Acknowledgments. We thank our students Christopher Ochs, Shrutee Singh, Mansi Pedgaonkar, Jalaj Asher and Anushri Mahajan for their work on the implementation of the OSWS System.

References

1. Radev, D.R., Fan, W., Zhang, Z.: WebInEssence: A Personalized Web-Based Multi-Document Summarization and Recommendation System. In: NAACL Workshop on Automatic Summarization, Pittsburgh, PA (2001)
2. An, Y., Chun, S., Huang, K., Geller, J.: Enriching Ontology for Deep Web Search. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 73–80. Springer, Heidelberg (2008)
3. Google Query Suggestion,
<http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=106230>
4. Yahoo Search Assistant,
<http://tools.search.yahoo.com/newsearch/searchassist.html>
5. Bing Search Suggestions,
http://help.live.com/help.aspx?project=w1_searchv1&querytype=keyword&query=tseggusotua&mkt=en-US
6. An, Y., Geller, J., Wu, Y., Chun, S.: Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources. In: Proceedings of the 2007 ACM Symposium on Applied computing, pp. 1667–1672. ACM-SAC, Seoul (2007)
7. An, Y., Chun, S., Huang, K., Geller, J.: Assessment for Ontology-Supported Deep Web Search. In: 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pp. 382–388. IEEE Computer Society, Los Alamitos (2008)
8. Tian, T., Geller, J., Chun, S.A.: Predicting Web Search Hit Counts. In: WIC, Toronto, Canada, (2010) (accepted for Publication)
9. Lee, T.B., Hendler, J., Lassila, O.: The Semantic Web. Scientific American Magazine (2001)
10. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management, pp. 652–659. ACM Press, New York (2004)
11. Ontology Design Patterns (ODP),
http://ontologydesignpatterns.org/wiki/Main_Page
12. Open Biological and Biomedical Ontologies (OBO),
<http://www.obofoundry.org/>
13. Niles, L., Pease, A.: Towards a standard upper ontology. In: Proceedings of the international conference on Formal Ontology in Information System, pp. 2–9. ACM, New York (2001)
14. Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove (2000)

How to Modify on the Semantic Web?

A Web Application Architecture for Algebraic Graph Transformations on RDF

Benjamin Braatz* and Christoph Brandt

SECAN-Lab, Université du Luxembourg
benjamin.braatz@uni.lu, christoph.brandt@uni.lu
<http://wiki.uni.lu/secan-lab/>

Abstract. In this paper, we show how formal methods of algebraic graph transformation can be made available in the technological environment of the Semantic Web. This new and promising approach allows to model and formulate transformation operations on a high level of abstraction. To demonstrate its feasibility, we develop a small book shop case study. We show how some of the necessary modification operations in this settings can be realised by the proposed SPARQL/Update language and by our rule-based graph transformation approach. In a comparison, we highlight the main differences between these two approaches. The most important benefit of our rule-based transformation approach is that it facilitates to move from application engineering to rule engineering and thereby support generic solutions, which lower the maintenance efforts by supporting the declarative specification of modification operations.

1 Introduction

The Semantic Web, based on the Resource Description Framework (RDF, as specified in [1] and related documents), is an emerging technology for managing and maintaining heterogeneous, distributed, structured data and metadata, which are stored as triples in RDF data stores or (from the more formal point of view) RDF graphs. While a lot of publications consider the deduction on RDF data by inference rules, the modification of RDF graphs has not been in the focus until recently. As an application scenario, we present a book shop example, show how the relevant data are represented in an RDF data store and motivate some modification operations, which are required in this context, in Sect. 2.

The SPARQL/Update language, which is proposed in [2,3] and currently developed, is one of the most promising approaches regarding modification operations on RDF. In Sect. 3, the operations from the application scenario are realised using SPARQL/Update. The development of SPARQL/Update is to a large extent practice-driven, i. e., the requirements of applications in the field of modifications motivate and determine the collection of language primitives.

* The first author is supported by the National Research Fund, Luxembourg, and cofunded under the Marie Curie Actions of the European Commission (FP7-COFUND).

In this paper, we propose the use of algebraic graph transformation (see [4] for a recent overview of the field) as an alternative technique to modify RDF graphs. The theoretical framework of algebraic graph transformation, which uses category theoretical methods to allow abstract formal reasoning about rule-based transformations, has been adapted to the needs of RDF in [5,6]. In Sect. 4, we show how the example operations can be implemented as algebraic graph transformation rules, while we discuss their realisation in a Semantic Web architecture in Sect. 5. This approach is theory-driven in the sense that the notion of transformation rule, which replaces the collection of language primitives found in SPARQL/Update, is motivated by a formal understanding of modifications on RDF data.

The two approaches to modifications on RDF, SPARQL/Update and algebraic graph transformation, are compared in Sect. 6, where we also shortly discuss other related work. Finally, in Sect. 7, we summarise the present work and give some possible lines of future work.

2 Application Scenario

Throughout the paper, we consider a book shop application based on Semantic Web technologies as an example scenario. The shop has two kinds of users: employees of the shop who can change the catalogue by adding and deleting books to and from it and customers who can buy these books by adding them to their shopping carts and later ordering the contents of the shopping cart.

For the representation of RDF data we use the Notation 3 (N3) syntax defined in [7]. Figure 1 shows an RDF representation of the book “The Lord of the Rings” by J. R. R. Tolkien on the one hand in N3 and on the other hand as a graphical visualisation with subjects and objects as vertices and triples as edges. Such an N3 representation might be returned when a client accesses the URI `http://shop.example.com/isbn/978-0-261-10238-5`. Representations of the same information in RDF/XML or HTML could, of course, also be provided by the application.

The representation of the shopping cart of a user is shown in Fig. 2, where we again give the N3 representation as well as a graphical visualisation. This representation (or an equivalent RDF/XML or HTML representation) will be returned when a client accesses the URI `http://shop.example.com/users/JohnDoe`. It should only be accessible if the corresponding user is authenticated with the application. The details of possible authentication schemes are outside the scope of this paper.

In the following, we will assume that the data store of our application is represented by a single RDF graph, which contains the graphs in Fig. 1 and 2 as parts. Furthermore, the exact selection of the whole RDF graph that is provided in response to a GET request has to be determined by some mechanism outside the scope of this paper.

The question arises how the operations of employees and customers on this data store can be specified and implemented. As examples, we will consider the following operations:

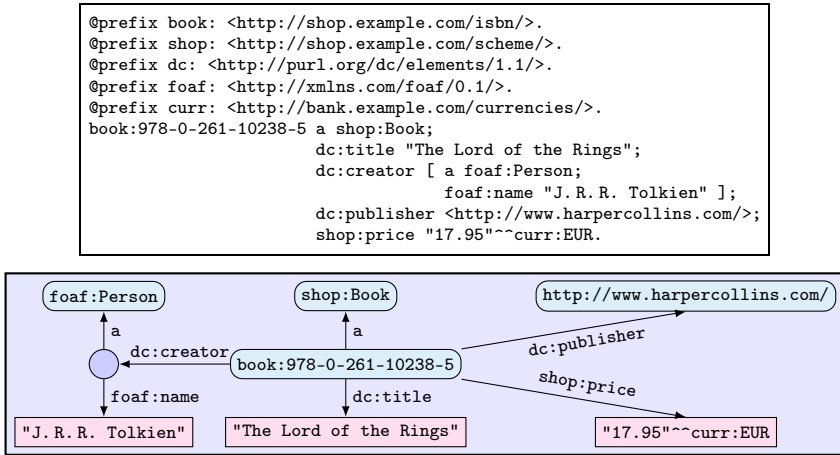


Fig. 1. Representation of a book in RDF

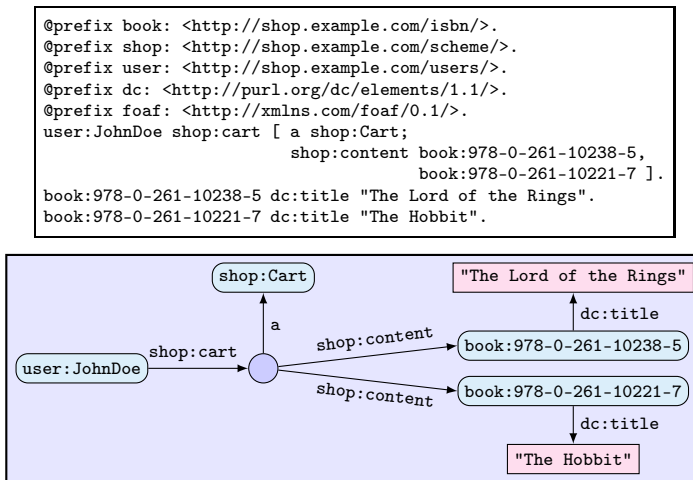


Fig. 2. Representation of a shopping cart in RDF

- The blank node representing the author in Fig. 1 is *replaced* by a URI for that author, such that other books of the author can also be linked to the same URI and a navigation to all books of an author becomes possible.
- A book of this author is *added* to the catalogue.
- A book is *added* to the shopping cart of a customer.

In the following sections, we will realise these operations first by the current version of the proposed SPARQL/Update language and then by the help of formal techniques from algebraic graph transformation.

3 Solution by SPARQL/Update

The SPARQL/Update language was proposed in [2] and is currently in the process of standardisation, where a working draft is available in [3]. It extends the RDF query language SPARQL (defined in [8]) by keywords for modifying the contents of an RDF triple store.

It can be used to realise the required operations on the data store as shown in Fig. 3 till 5. In Fig. 3, the blank node representing the author in Fig. 1 is replaced by the URI `author:Tolkien`. This is achieved by first binding the variable `?author` to this blank node in the WHERE clause, then removing the triples including the node as subject or object in the DELETE clause and, finally, inserting corresponding triples for `author:Tolkien` in the INSERT clause.

```

PREFIX book: <http://shop.example.com/isbn/>.
PREFIX author: <http://shop.example.com/authors/>.
PREFIX dc: <http://purl.org/dc/elements/1.1/>.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>.
MODIFY
DELETE { book:978-0-261-10238-5 dc:creator ?author.
        ?author a foaf:Person;
              foaf:name "J. R. R. Tolkien" }
INSERT { book:978-0-261-10238-5 dc:creator author:Tolkien.
        author:Tolkien a foaf:Person;
              foaf:name "J. R. R. Tolkien" }
WHERE { book:978-0-261-10238-5 dc:creator ?author.
        ?author a foaf:Person;
              foaf:name "J. R. R. Tolkien" }

```

Fig. 3. SPARQL/Update query for replacing blank node by URI

In Fig. 4, the insertion of another book by the same author is realised by an INSERT DATA query, which does not use any variables, but just lists the triples that are to be added to the data store.

```

PREFIX book: <http://shop.example.com/isbn/>.
PREFIX shop: <http://shop.example.com/scheme/>.
PREFIX author: <http://shop.example.com/authors/>.
PREFIX dc: <http://purl.org/dc/elements/1.1/>.
PREFIX curr: <http://bank.example.com/currencies/>.
INSERT DATA { book:978-0-261-10273-6 a shop:Book;
              dc:title "The Silmarillion";
              dc:creator author:Tolkien;
              dc:publisher <http://www.harpercollins.com/>;
              shop:price "10.80"^^curr:EUR }

```

Fig. 4. SPARQL/Update query for adding a book to the catalogue

Finally, in Fig. 5, we are adding the book that was just added to the catalogue in the data store to the shopping cart of Fig. 2. Since the shopping cart itself is represented by a blank node, which is not adressable from outside the data

```

PREFIX book: <http://shop.example.com/isbn/>.
PREFIX shop: <http://shop.example.com/scheme/>.
PREFIX user: <http://shop.example.com/users/>.
INSERT { ?cart shop:content book:978-0-261-10273-6 }
WHERE { user:JohnDoe shop:cart ?cart }

```

Fig. 5. SPARQL/Update query for adding a book to a cart

store, we are using a `WHERE` clause again to bind the variable `?cart` to it and afterwards just insert the corresponding `shop:content` triple.

The given SPARQL/Update queries only realise concrete instances of the desired operations. They replace a particular blank node by a URI, add a particular book to the catalogue and put it in the shopping cart of a particular user. If the operations have to be provided for generic situations—replacing all blank nodes that represent the same author by a corresponding URI, adding an arbitrary book to the catalogue or putting an arbitrary book into the shopping cart of an arbitrary user—then we need some other technology in order to generate SPARQL/Update queries from given parameters.

Regarding a possible Web application architecture, such a generation of SPARQL/Update queries could be done either on the client side, using a public SPARQL/Update end point, or on the server side, leading to application-specific service interfaces.

The first alternative, providing a public SPARQL/Update end point, poses serious security issues since clients are able to request arbitrary modifications on the data store, where objectionable queries can only be distinguished from allowed queries by a deep analysis of their respective effect.

In both cases, client-side as well as server-side generation of queries, an assumed generation procedure will be embedded into specific application code that constructs the query code and also realises the user interface or a Web service. This complicates the handling of security issues, the analysis of the impact of queries on the data store and the maintenance of the operations.

4 Solution by Algebraic Graph Transformation

As an alternative to SPARQL/Update, we propose the use of algebraic graph transformation for modifying RDF data stores. A comprehensive treatment of the theory of algebraic graph transformation, more specifically of the “double-pushout” (DPO) approach to graph transformation, can be found in [4]. Since this approach is not applicable as-is to RDF graphs due to differences in the formalisation of graphs—the DPO approach uses graphs that allow multiple edges with identical labels between two nodes—, we have developed a variant of this approach in [5] and [6] and have also shown first theoretical results regarding the composition of RDF transformation rules.

In Fig. 6, an RDF transformation rule realising the replacement of blank node representations of authors by URIs is shown with its full formal structure and a

compact notation, which will be used in the following. A rule consists of a left-hand side L , an interface I and a right-hand side R . The interface is connected to left-hand and right-hand side by homomorphisms l and r , respectively, which are required to be injective by the theory and are, for practical purposes, always chosen to be inclusions. This also enables us to draw L , I and R into one diagram in the compact notation, where the nodes and triples that are only present in L are marked by $\{\text{del}\}$ and those that are only present in R by $\{\text{add}\}$.

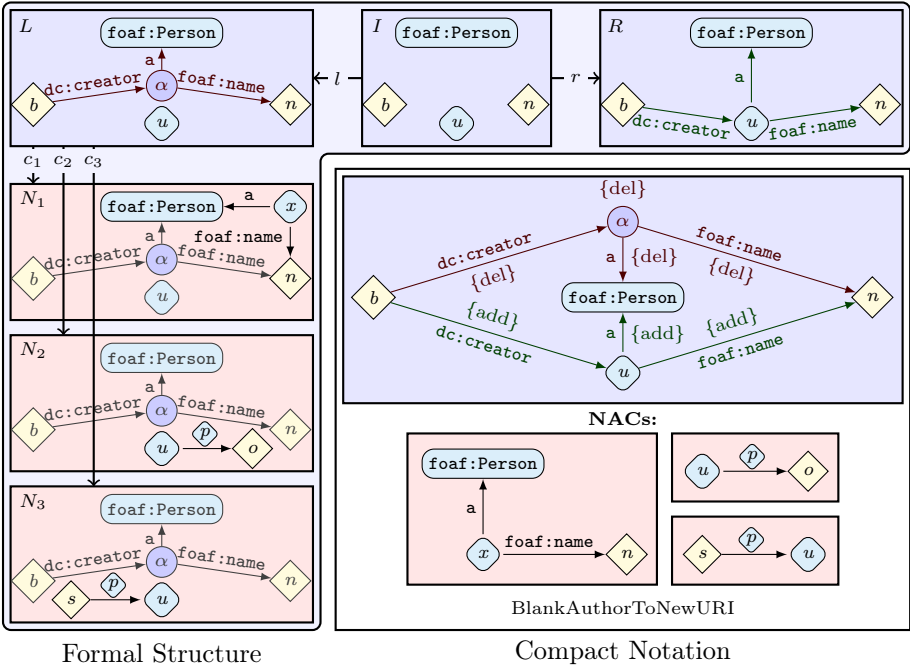


Fig. 6. RDF transformation rule for replacing blank node by new URI

The patterns used in these rules do not only contain URIs, literals and blank nodes, but also two kinds of variables. The general variables, depicted by sharp diamonds, can be assigned to URIs, literals and blank nodes, while the URI variables, depicted by rounded diamonds, can only be assigned to URIs. The variable sets of L , I and R are required to be bijectively mapped by l and r , which intuitively means that they are identical for the whole rule.

The additional blank nodes and triples in the left-hand side L of an RDF transformation rule are the ones that will be deleted when the rule is applied, while the additional blank nodes and triples in the right-hand side R are created. For example, the rule in Fig. 6 deletes a blank node α representing an author with name n of a book b with the corresponding triples and creates equivalent triples for a URI u representing that author. Observe that only blank nodes can be deleted and created, while URIs and literals are assumed to be globally and

eternally given. This is reflected in the requirement that variables, which can be assigned to URIs and literals, have to be the same in L , I and R which means that they are neither deleted nor created.

In addition to this, a transformation rule contains a set of negative application conditions (NACs), which describe situations in which the rule cannot be applied. These are formalised as extensions of the left-hand side by the structures that are forbidden, where the irrelevant parts of L are omitted in the compact notation. For instance, the NAC N_1 in Fig. 6 forbids that the rule introduces a new URI for an author if there is already another URI (represented by the URI variable x in the NAC) with the same name. The NACs N_2 and N_3 require that the URI that is used for the author (represented by the URI variable u in the rule) is not already used in any other triple as subject or object.

Figure 7 shows an application of this transformation rule, where the predicates in the lower row are abbreviated and the dots symbolise that the graph might be much larger than the shown part. The left-hand side L of the RDF transformation rule is found in the host graph G by a match homomorphism m , which also assigns the variables. The effect of the rule is now described in terms of constructions from category theory (see, e.g., [9] for an introduction). The deletion is obtained by a so-called minimal pushout complement (MPOC) D , which intuitively just deletes all additional blank nodes and triples of the left-hand side; this is only possible if the deleted blank nodes are not part of other triples in G (*dangling condition*) and deleted blank nodes are not identified to other blank nodes or variables (*identification condition*). The insertion is then obtained by a pushout (PO), i.e., a disjoint union of D and R over the interface I as common subpart.

Recall that the NAC N_1 forbids the application of the rule BlankAuthorToNewURI in Fig. 6 if there is already another URI for that author. Therefore, we need a second rule BlankAuthorToExURI, shown in Fig. 8, to allow the replacement of a blank node author representation by an already existing URI.

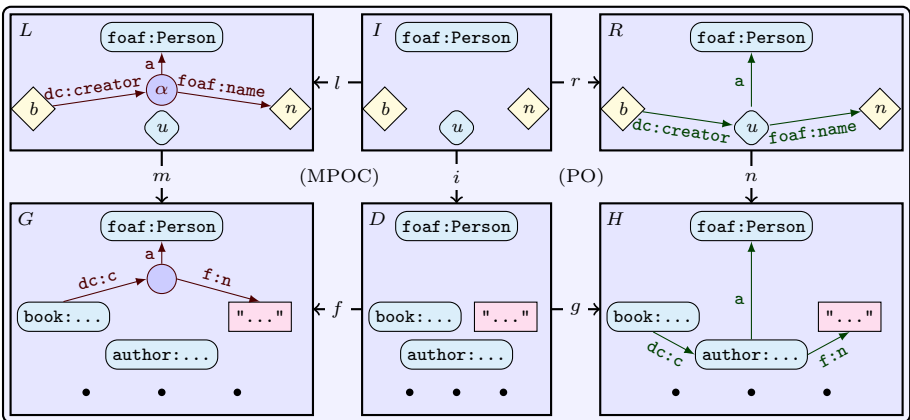


Fig. 7. Application of RDF transformation rule

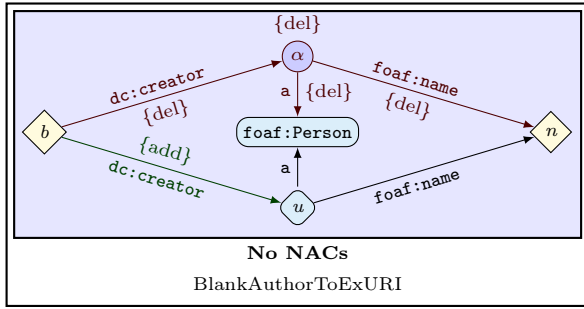


Fig. 8. RDF transformation rule for replacing blank node by existing URI

This rule uses a URI—represented by the URI variable u —that is already of type `foaf:Person` and connected to a `foaf:name` in the match of the left-hand side.

For the second example operation, adding a book to the catalogue, we specify several transformation rules in Fig. 9. The first one, `AddBook`, is used to insert the minimal required information, a title and the price, into the RDF data store. The NACs ensure that the URI for the new book is not already in use. The rules `AddAuthorToBook` and `AddPublisherToBook` can then be used to add authors and a publisher as optional information.

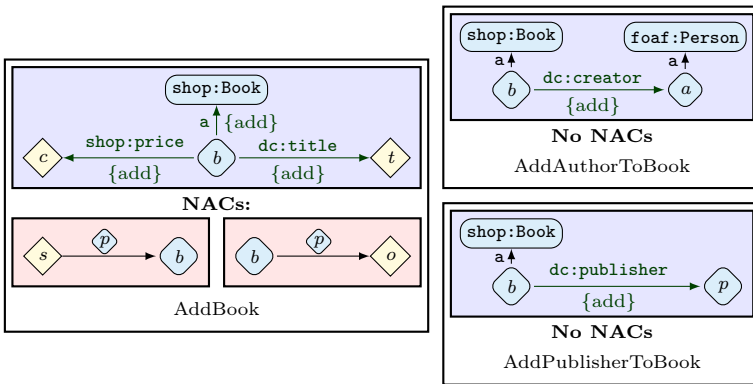


Fig. 9. RDF transformation rules for adding a book to the catalogue

Finally, the rather simple rule `AddBookToCart` adds a book to the shopping cart of a user (identified by the match of the URI variable u).

In this section, we have seen how graph transformation rules are formally structured and applied to RDF graphs as formal models of triple stores. Moreover, we have modelled the example operations from the application scenario

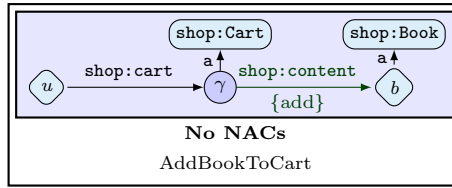


Fig. 10. RDF transformation rule for adding a book to a cart

by transformation rules. In the next section, we will sketch an architecture for providing and executing graph transformation rules using standard (Semantic) Web techniques.

5 Web Application Architecture for Algebraic Graph Transformation

Transformation rules can themselves be represented in RDF. Figure 11 shows an RDF representation of the rule `BlankAuthorToExURI` from Fig. 8. The rule is identified by the URI `http://shop.example.com/rules/BlankAuthorToExURI` and its constituents are connected to it by the `gt:preserve`, `gt:delete` and `gt:create` predicates, where triples are reified using the `gt:subject`, `gt:pred` and `gt:object` predicates. Variables and blank nodes in the rule are represented by blank nodes with corresponding types `gt:UVar`, `gt:Var` and `gt:Blank`, where variables also have a `gt:name`, which will be used to identify them in requests.

```

@prefix rule: <http://shop.example.com/rules/>.
@prefix gt: <http://rdfgratra.example.org/>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
rule:BlankAuthorToExURI a gt:Rule;
  gt:preserve [ a gt:Triple;
                gt:subject _:1; gt:pred a; gt:object foaf:Person ],
                [ a gt:Triple;
                  gt:subject _:1; gt:pred foaf:name; gt:object _:2 ];
  gt:delete [ a gt:Triple;
              gt:subject _:3; gt:pred dc:creator; gt:object _:4 ],
              [ a gt:Triple;
                gt:subject _:4; gt:pred a; gt:object foaf:Person ],
              [ a gt:Triple;
                gt:subject _:4; gt:pred foaf:name; gt:object _:2 ],
              _:4;
  gt:add [ a gt:Triple;
           gt:subject _:3; gt:pred dc:creator; gt:object _:1 ].
_:1 a gt:UVar; gt:name "author".
_:2 a gt:Var; gt:name "name".
_:3 a gt:Var; gt:name "book".
_:4 a gt:Blank.

```

Fig. 11. RDF representation of transformation rule `BlankAuthorToExURI`

Such a representation of the rule shall be the response to a GET request on its URI. Its execution is—in accordance with the principles of RESTful Web applications in the sense of [10]—triggered by a POST request. A (partial) match is given by using the `gt:name` attributes as the names of POST variables. For a given partial match, there are essentially three possibilities:

- The match is total or there is only one possibility to complete it to an applicable total match. In this case, the rule is executed and results in a response with the image of the right-hand side as content.
- The match is already not allowed (even if it still might be partial) due to a NAC, the dangling or the identification condition. This results in a description of the violated condition in the response.
- There are multiple possible completions of the given partial match. The response contains the possible completions of the match (if they are already restricted enough to make this feasible). It would also be reasonable to offer a possibility to execute a rule on all possible extensions of a given partial match—especially if they only differ in the matches of blank nodes, which cannot be identified from the outside.

The storage of the applicable transformation rules on the server allows for an advanced security architecture. It can be decided per transformation rule which users are allowed to execute it—akin to stored procedures in relational databases. If the users are also administrated in the RDF data store then this is easily achieved by a `gt:allowed` predicate and corresponding triples relating the rules and the users or user groups.

A more intricate problem arises in the case of the rule `AddBookToCart` in Fig. 10. All customers should be able to execute this rule, but only on their own cart. This can be solved by introducing a `gt:bind` predicate for variables, which is a replacement for `gt:name` and has the effect that the match for a variable is not chosen by the user but by the environment—in this case binding u to the URI of the user who is currently logged in.

In the following section, we will compare the solution by SPARQL/Update from Sect. 3 and the proposed solution by the help of formal techniques using algebraic graph transformation, given in Sect. 4 and the current section.

6 Comparison and Related Work

The differences between the SPARQL/Update approach and our algebraic graph transformation approach for modifying RDF graphs are summarised in Table 1. SPARQL/Update queries are supposed to be generated by applications in an ad hoc manner, while graph transformation rules are pre-defined and applied by providing a match of the left-hand side. The language primitives of SPARQL/Update are derived from the needs arising in practice, while the concepts of graph transformation are derived from a category theoretical model of transformation. All in all, this leads to SPARQL/Update being a language that is supposed to be used in application engineering, while algebraic graph transformation enables

Table 1. Comparison of solutions

SPARQL/Update	Algebraic Graph Transformation
<i>Ad hoc</i> generation of queries for <i>specific</i> modifications	Application of <i>generic, pre-defined</i> rules via matches
<i>Practice-driven</i> : language primitives derived from practical needs	<i>Theory-driven</i> : concepts derived from categorical model of transformations
Language used in <i>application engineering</i>	Formalism enabling <i>rule engineering</i>

rule engineering, which can be done independently of the application context on a higher level of abstraction.

Moreover, the formal apparatus of graph transformation allows to reason about the effects of transformations *ex ante*, define languages of allowed graph structures by grammars, compose transformation rules to model complex operations and analyse dependencies in transformation sequences.

In [11], the *Delta* ontology for differences between RDF graphs is developed. This approach leads to a structure that is very similar to our notion of RDF transformation rules. A preserved graph (part) is connected to a deleted part by a `delta:deletion` predicate and to an inserted part by a `delta:insertion` predicate, where the possibility to nest graphs inside graphs in N3 is used. While our motivation is to provide a generic transformation facility, Delta puts its focus on documenting and possibly reapplying changes between RDF graphs. Up to now, a formal semantics for Delta differences is not formulated. Furthermore, they are not intended to be used in multiple situations, which is why an equivalent to the application of a rule *via a match* is missing.

In both cases, SPARQL/Update as well as Delta, it might be worthwhile to consider algebraic graph transformations as a formal foundation for the concrete languages.

In [12], the *Griwes* framework for knowledge representation languages is proposed. It aims at providing a common formal model for formalisms such as RDF or Conceptual Graphs. Griwes provides a notion of mappings and proofs between graphs, which corresponds to the morphisms in our approach, but it does not have a notion of transformation rule or modification operation. While it would be possible to formulate a similar transformation concept on the level of Griwes, we prefer to work directly on the RDF graphs to reduce the formal overhead.

7 Conclusion and Future Work

In this paper, we have proposed algebraic graph transformation as a means to modify on RDF data stores. This leads to a shift from modifications that are developed as part of the Web application engineering to modifications that are specified in a dedicated rule engineering process and, thus, facilitate among other things the reuse and maintenance of transformation rules.

In future work, the relations between transformations and inferences should be examined. More specifically, situations in which inferred triples are deleted shall be analysed, since this leads to the counter-intuitive result that the deleted triple will be inferred again at the next possibility and is, thus, not effectively deleted. Similar situations also arise in SPARQL/Update.

On the theoretical side, the rich theory that is available in the field of algebraic transformation can be used. However, as already shown in [5,6], this requires some adaptations.

References

1. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax (W3C Recommendation). In: World Wide Web Consortium, W3C (February 2004), <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
2. Seaborne, A., et al.: SPARQL Update: A language for updating RDF graphs (W3C Member Submission). In: World Wide Web Consortium, W3C (July 2008), <http://www.w3.org/Submissions/SPARQL-Update/>
3. Schenk, S., Gearon, P.: SPARQL 1.1 Update (W3C Working Draft). In: World Wide Web Consortium, W3C (October 2009), <http://www.w3.org/TR/2009/WD-sparql11-update-20091022/>
4. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. Springer, Heidelberg (2006), doi:10.1007/3-540-31188-2
5. Braatz, B., Brandt, C.: Graph transformations for the Resource Description Framework. In: Ermel, C., Heckel, R., de Lara, J. (eds.) Proc. GT-VMT 2008. Electronic Communications of the EASST, vol. 10 (2008), <http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/158>
6. Braatz, B.: Formal Modelling and Application of Graph Transformations in the Resource Description Framework. Dissertation, Technische Universität Berlin (2009)
7. Berners-Lee, T.: Notation 3: A readable language for data on the Web. World Wide Web Consortium, W3C (October 2007), <http://www.w3.org/DesignIssues/Notation3>
8. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (W3C Recommendation). In: World Wide Web Consortium, W3C (January 2008), <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>
9. Adámek, J., Herrlich, H., Strecker, G.E.: Abstract and Concrete Categories: The Joy of Cats. Dover, New York (2009), <http://katmat.math.unibremen.de/acc/>
10. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine (2000), <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
11. Berners-Lee, T., Connolly, D.: Delta: an ontology for the distribution of differences between RDF graphs. In: World Wide Web Consortium, W3C (August 2009), <http://www.w3.org/DesignIssues/Diff>
12. Baget, J.F., et al.: Griwes: Generic model and preliminary specifications for a graph-based knowledge representation toolkit. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS (LNAI), vol. 5113, pp. 297–310. Springer, Heidelberg (2008), doi:10.1007/978-3-540-70596-3_21

T².O.M. T.O.M.: Techniques and Technologies for an Ontology-Based Mobility Tool with Open Maps

Michele Ruta, Floriano Scioscia, Saverio Ieva, and Eugenio Di Sciascio

Politecnico di Bari,
via Re David 200, I-70125,
Bari, Italy

{m.ruta,f.scioscia,disciascio}@poliba.it ieva@deemail.poliba.it

Abstract. Nowadays, vehicle navigation systems have a limited effectiveness in satisfying user interests and needs. They basically locate the user's GPS position on a map and calculate a path toward a given destination. This paper presents an innovative semantic-based navigation system, acting as intelligent agent offering an advanced assistance during a road trip. It integrates a mobile matchmaker, which allows a knowledge-oriented destination discovery, in order to fully accomplish requirements of a user during her travel. The framework we propose is based on open standards, and particularly it uses crowd-sourced maps available under the *OpenStreetMap* project, where Points Of Interest (POI) of the basic cartography are semantically featured in order to enable novel services supporting a traveling user.

1 Introduction

Road congestion, environmental pollution and uncontrolled expenditure of energetic resources are increasingly urgent problems in many urban areas. In the last few years, the application of ICT to the transportation domain has opened promising possibilities to enable smarter traffic management solutions, while also promoting new services for drivers and passengers. This world-wide effort is usually referred to ITS (Intelligent Transport System) initiative. ITS aims at improving travellers' safety and satisfaction, as well as reducing traffic and transportation times and costs. In the short to medium run, the best that could be reasonably achieved is a combination of (i) automated traffic surveillance and road charging in congested areas with (ii) a set of assistive measures exploiting information and communication technologies within vehicular equipments.

Particularly, assisted navigation systems support drivers in planning trips and during travels: they are up to now one of the most widespread parts of ITS technological effort. Navigation softwares allow to locate and track a user, to calculate a route toward a given destination and enrich the travelling experience through visual maps and voice guidance. Nevertheless, current navigators present several limits. First of all they calculate a path by using minimum distance criteria without taking into account factors influencing travel such as road

congestions, accidents, dangerous routes due to either weather conditions and vehicle state. Furthermore, the informative content supporting map navigation is currently very low, as the only advanced search option they enable is based on POI (Point Of Interest). Unfortunately a POI can be identified and selected only through a name-based or category-based search, filtered according to proximity criteria. Such syntactic-based resource retrieval is inherently affected by poor recall, since only exact matches are supported. Moreover, the user cannot indicate characteristics and properties the destination should have, so that her personal preferences and collateral requirements conditioning path calculation are not taken into account.

This paper introduces an innovative framework to enhance functionalities of navigation systems for better assisting users in their trips, envisioning a general approach to retrieve and provide information which may be useful not only to enhance travel satisfaction and safety, but also to regulate vehicular traffic and environmental impact. Borrowing from the Semantic Web initiative, we propose a knowledge-based system whose final goal is to provide a navigation based on semantics of both user's needs and profile and on context annotation. Knowledge Representation theoretical studies are adapted and applied to ITS field. This opens new interesting possibilities, including: formalization of vehicles, context and actors annotations that become machine understandable so enabling interoperability; reasoning on descriptions and inference of new knowledge; Open World Assumption (OWA) by-passing structured data models. By means of formal ontologies, expressed using Description Logics (DLs) formalisms [1] and particularly OWL [2], knowledge about transportation and travel domain can be modeled and exploited in order to derive new information from the one stated within metadata associated to POIs and user profiles.

Previous framework and approach has been implemented in an open source mobile navigator equipped with deductive reasoning capabilities, able to leverage semantically enriched crowd-sourced maps available under the *OpenStreetMap* project [2]. POI annotation enables a logic-based discovery of location features w.r.t. an articulate request expressed by the user. Based on the formal semantics of such descriptions, POIs are ranked according to the degree of satisfaction of the request. A lightweight version of non-standard inference algorithms, *i.e.*, *Concept Abduction*, *Concept Contraction* and *Bonus Calculation* [3] is used for that. Furthermore, such an approach allows to provide the user with an explanation of matchmaking outcomes, in order to increase her confidence in the quality of search results and to foster further interactions with the system.

The remaining of the paper is structured as follows: in the next section, current state of the art in ITS and navigation systems is briefly recalled, Section [3] outlines system features and architecture evidencing its peculiarities and in Section [4] a toy case study will explain the proposed approach and the rationale motivating it. Finally conclusion and future work terminate the paper.

¹ OWL 2 Web Ontology Language overview, W3C Recommendation 27 October 2009, <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>

² OpenStreetMap: <http://www.openstreetmap.org/>

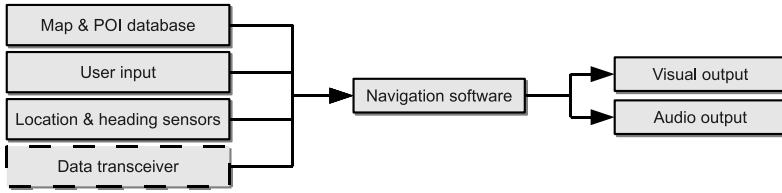


Fig. 1. General architecture of personal navigation systems

2 State of the Art

Current personal navigation systems share the common basic architecture depicted in Figure 1 [5]. The navigation software is responsible for location tracking, route calculation and update toward a destination, and output to the user through both map visualization and voice guidance. Input comes from the following main sources.

1. Map database: which contains geographic and cartographic data, as well as a catalog of POIs. The database can be either pre-loaded and stored locally in the navigation device (such an approach is currently followed in the majority of solutions), or retrieved in real-time during system usage. Hybrid approaches are also possible.

2. User which inserts information about destination, either as address, GPS coordinates or by selecting a POI from the catalog.

3. Location and heading sensors which are key elements in real-time navigation devices. In all current commercial solutions, location is determined through satellite-based technologies, such as *GPS* (Global Positioning System).

4. Data transceivers, usually based on cellular network connectivity, are currently exploited by many systems to acquire additional information during navigation. This is mainly the case of navigation applications for smartphones, but also several stand-alone device vendors integrate a data module in their high-end products. Downloaded data may include temporary map modifications –due to *e.g.*, road maintenance– as well as information about traffic, accidents and weather. That allows the navigation software to determine the best route based not only on static road information, but also on dynamic conditions.

Recent navigation systems provide value-added features for POIs such as phone numbers –directly usable by smartphones and other telephony-enabled navigation devices– and reviews. Nevertheless, significant limitations affect POI management in current solutions: they are organized in a rigid and often oversimplified classification and, as such, they allow category-based or name-based queries only. This prevents users from performing semantically richer searches, limits the quality of resource retrieval and hinders more advanced use cases, such as personalized suggestions based on user profiles. Studies about the behavior of users highlighted the importance of increasing familiarity with the surrounding environment through more sophisticated POI tagging, including social and contextual information [6]. Solutions are thus required for a more expressive

annotation of POI characteristics. Our proposal supports the above-mentioned features through a robust framework –based on Semantic Web technologies– for resource annotation and discovery, while also tackling practical issues related to the creation, maintenance and storage of world-scale POI catalogs.

A fundamental issue is that cartography databases are proprietary, owned and maintained by companies that license them to navigation system manufacturers. File formats and data structures are not open, hence augmenting data is not possible. On the other hand, creating and maintaining a detailed map database would require huge resource investments, even for a single municipality. Recently, a different approach to map database development was conceived, based on *crowd-sourcing*: anyone can contribute to a common worldwide cartography by uploading collected data about road segments and POIs to a website. The most mature project is *OpenStreetMap*: map data are published in an open format under the Creative Commons Attribution-ShareAlike 2.0 license³, granting anyone the permission to use, copy, share and modify. Contributing a road segment to a map region typically involves the following steps.

1. Record a *GPS trace* while traveling on the road, using *GPS logging* software which periodically records current geographical coordinates. A collection of data points is obtained.

2. Annotate relevant information about nearby POIs during the trip. This optional step could be performed with paper and pencil, but specialized *map collection* software has been recently developed for several mobile platforms, integrating a GPS logger and a user interface dedicated to terrain and landmark annotation.

3. Import collected data into an OpenStreetMap-compatible *map editor*, which reverts sets of raw geographic data points into road segments and allows the user to add road information (such as name, type and speed limit), mark different terrain areas and insert POIs. Finally, collected data are uploaded to OpenStreetMap web servers. After integration into the database, they become visible on the OpenStreetMap website and usable with all compatible software.

A wide range of freely available software tools exists on a wide range of smartphone and computer platforms⁴. In this paper OpenStreetMap is adopted because it proved itself a viable solution for global-scale collaborative cartographic data collection and enrichment, from both legal (licensing) and technical (scalability, interoperability, openness to new features) standpoints.

3 Framework and Approach

Main features of the proposed framework are: (i) to follow multi-platform code development rules; (ii) to adopt open source policies in both map data and software components creation; (iii) to enable a semantic-based discovery of POIs

³ Summary and link to full license text are available at <http://creativecommons.org/licenses/by-sa/2.0/>

⁴ They are listed in the OpenStreetMap Wiki, <http://wiki.openstreetmap.org/>

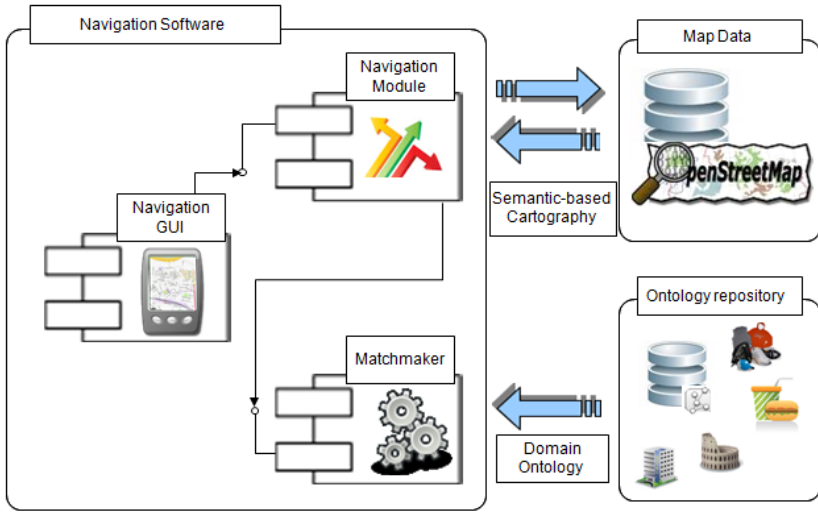


Fig. 2. System Architecture

enhancing the current trivial category-based search of them filtered by nearness criteria. In the following subsections the overall architecture and the map annotation procedure are reported, respectively.

3.1 System Architecture

The proposed system adopts a so-called *thick client* paradigm: differently from classical client-server model, most of the processing is done on the client side. The rationale for this approach is to grant full operation also in case of slow or congested data links. Anyway external interaction is also enabled, via wireless communication protocols, in order to set up further (either client-server or peer-to-peer) services. The overall system architecture is depicted in Figure 2.

The mobile client (a standard PDA or smartphone) is equipped with the navigation tool proposed here. Its main components are summarized in what follows.

1. Graphical User Interface (GUI). User can access all functionalities in a quick and straightforward way. Particularly, ontology-based requests addressed to the system can be composed exploiting a graphical toolbar adopting some UI elements familiar to PC users. In Section 4 toy example is presented to better explain how a semantic query is build.

2. Navigation engine. User requests are processed by the navigation module. It extracts information from installed map files and a map graphical view is rendered through the GUI, using the embedded Scalable Vector Graphics engine. We will give a more detailed description of this component later on.

3. Map data files. All map data are locally stored within one or more related files, which encapsulate both geographical data and a semantic annotation of resources. The file structure is described in detail in the next subsection.

4. Matchmaker. This component is aimed at processing semantic-based user requests exploiting lightweight *Concept Abduction*, *Concept Contraction* and *Bonus Calculation* algorithms. They are used by the navigation module to perform an advanced discovery of the best path for a user also providing a detailed explanation of matchmaking results in order to allow user to refine requests in an iterative way. Semantic Web technologies can provide significant benefits in resource retrieval problems [3]. W.r.t. standard Information Retrieval process, based on keyword search over semi-structured data, a semantic-based matchmaking: (i) overcomes the low recall because of the Open World Assumption paradigm, *i.e.*, a resource is excluded only if its description explicitly contradicts some axioms in the user request; (ii) provides logic-based ranking criteria to select the best matching resources from the available ones; (iii) allows an accurate logic-based explanation of outcomes thanks to the inferred knowledge.

As far as map navigation module is concerned, we decided to extend the *Navit*⁵ software navigation tool which is released as open source project under the GPL General Public License. It provides routes calculation and POI finding (but only via a category-based search and selection). Another noticeable feature is multi-platform portability over several mobile Operating Systems through available cross-compilers.

In order to enable semantic-based POI discovery functionalities, the original Navit codebase has been enriched with new software components whose features are explained hereafter while describing operation steps involving the proposed system:

1. User starts navigation tool, selecting the proper application icon on the screen. The navigation module is loaded; it reads a configuration file containing startup information and an annotated user profile expressed in DIG 2.0 [2], a more compact OWL-DL variant. It resumes personal information like interests and hobbies which could either be inserted by the user in a first-time setup phase or imported from other installed application or social network such as Facebook⁶ and LinkedIn⁷.
2. The navigation tool checks the matchmaker which runs in background and acts as a server listening for requests submitted by the navigation module (which acts as client).
3. The system starts keeping track of user position, by means of available location services, *e.g.*, internal/external GPS antennas or location APIs.
4. POI discovery can happen in two different ways. (i) User explicitly submits her request selecting related features from main menu. Basically she indicates a general category search (*e.g.*, Cultural Heritage, Accommodation, Entertainment, Dining), each of them corresponding to a given ontology. Then she browses all concepts and roles within the ontology so detailing her request. (ii) Alternatively, the navigation tool is able to retrieve nearby POIs best matching user interests, by simply referring to her profiles and preferences.

⁵ <http://www.navit-project.org/>

⁶ <http://www.facebook.com/>

⁷ <http://www.linkedin.com/>

In order to take geographical distance into account to grade semantic match-making results, we introduce a Score Combination Function (SCF), also known as *utility function*:

$$SCF(r, POI) = 100[1 - \frac{rankPotential(r, POI)}{rankPotential(r, \top)} (1 + \frac{distance(User_GPS, POI_GPS)}{max_distance})]$$

where $rankPotential(r, s)$ [3] is the semantic distance measuring the degree of correspondence between request r and point of interest annotation POI , computed solving the Concept Abduction Problem (CAP) [3]; $rankPotential(s, \top)$ is the maximum semantic distance, which depends only on the request and the axioms in the selected ontology. The second local score factor weights the influence of geographical distance: a ratio is used between POI-user distance and maximum distance the user can tolerate for discovery.

3.2 Map Enhancement

As said the proposed system is based on OSM. Map data are exported and edited in an XML format adhering to a simple schema which includes three basic elements: (i) **nodes**, *i.e.*, single geospatial points; (ii) **ways**, intended as ordered sequences of nodes; (iii) **relations**, which group multiple nodes and/or ways. Each element can include a given number of general tags having a key/value pair structure. There are no restrictions about attributes, *i.e.*, user can create new needed tags accommodating previously unforeseen uses of maps. With reference to POI description, OSM default representation is made by unconnected nodes, adding one or more additional key-value pairs tag. For instance the `amenity=place_of_worship, religion=catholic` annotation refers to a catholic church.

In order to enable novel semantic-based discovery of POIs, we have integrated an ontology-based annotation within current descriptions in OSM. Knowledge Representation formalisms have been used, and particularly DIG in a subset of OWL-DL corresponding to the \mathcal{ALN} (Attributive Language with Unqualified Number Restrictions) DL. An important goal of this work is to introduce semantic POI annotation in a way that best fits OSM storage infrastructure. Similarly, mobile computing limitations must be taken into account, and particularly the size of map data to be transferred from OSM server to PDA should be minimized. Therefore, due to the verbosity of XML-based ontological languages such as DIG and OWL, compression techniques [8] have been employed to make semantic metadata more compact.

In detail, two new tags have been introduced in OSM:

```
<tag k="semantic:ouuid_counter" v="DIG description" />
```

whose *key* is the OUUID (Ontology Universally Unique Identifier) code marking the domain ontology the description refers to [7]. The *value* field contains a semantically annotated DIG description. The counter suffix has been added to allow splitting DIG compressed description in two or more tags. All of them will have the same key value, but a progressively increased counter.


```
<tag k="ontology:ouuid" v="Ontology URI" />
```

whose value attribute contains the URI from where the ontology can be downloaded. This tag can also be omitted, and in that case the ontology will be searched and retrieved locally from the device storage.

The following three cases are possible: (i) the POI is described using only one ontology and the compressed annotation is shorter than 255 characters; (ii) the POI refers to a single ontology, but compressed description is longer than 255 characters; (iii) the POI is expressed w.r.t. more ontologies. For each of them, the compression procedure accepts in input a DIG/OWL annotation and generates a *Base64* string output. If it is shorter than 255 characters, it can be inserted within a single semantic tag, otherwise it is split in 255 characters segments and corresponding tags are inserted within consecutive node elements.

The OSM file including semantic annotations can be finally submitted to the OpenStreetMap server. Then the navigation system will use binary map files generated by the encoding tool, which are smaller than original textual ones. It directly extracts POI annotations from binary data, decodes the compressed description and joins all segments (in case they overcome 255 characters limit).

```
<defindividual name="Cattedrale_Santa_Maria_Maggiore" />
<instanceof>
  <individual name="Cattedrale_Santa_Maria_Maggiore" />
  <and>
    <atom name="Church" />
    <all>
      <ratom name="has_age" />
      <atom name="Middle_Age" />
    </all>
    <all>
      <ratom name="has_style" />
      <atom name="Romanic" />
    </all>
    <atleast num="3">
      <ratom name="has_aisle" />
    </atleast>
    <atmost num="3">
      <ratom name="has_aisle" />
    </atmost>
    <atleast num="3">
      <ratom name="has_apse" />
    </atleast>
    <atmost num="3">
      <ratom name="has_apse" />
    </atmost>
    <atleast num="3">
      <ratom name="has_altar" />
    </atleast>
    <atmost num="3">
      <ratom name="has_altar" />
    </atmost>
    <atleast num="1">
      <ratom name="has_portal" />
    </atleast>
    <atmost num="1">
      <ratom name="has_portal" />
    </atmost>
  </and>
</instanceof>
```

Listing 1. POI DIG description

```
<node id='413962293'
  lat='41.3205606'
  lon='16.2862326'
  version='4'
  timestamp='2009-12-28T10:28:46Z'
  uid='191809'
  visible='true'>
  <tag k='amenity'
    v='place_of_worship' />
  <tag k='denomination'
    v='catholic' />
  <tag k='name'
    v='cattedrale S.Maria Maggiore' />
  <tag k='religion'
    v='christian' />
  <tag k='semantic:44211a01_1'
    v='eNqFjUOKgzAQhc9UeoLWtZv2AGEwQ
    xxIE0nGRS/kz64bXRyIqIpr6ptKoS0
    2qm4H3vvfEvJ8RMKN0oFGcwTCIGBx
    NVvmyDu+vP4FmNlcJwK7aptASAoy
    zXAEp5EzIrIDU/BComcywGQDPKQm0
    H39SKGFmZ1kDD60CovysV988hr7qt
    uqemWZ36Bj1MuY56s0cyd7AUNJ2Nr
    MdmYdg+6rta4Z3Lc/q7r7idbF7Q08
    44cq' />
  <tag k='ontology:44211a01'
    v='http://sisinfab.poliba.it
    /cultural.dig' />
</node>
```

Listing 2. OSM XML excerpt of a semantically annotated POI

Furthermore, the system checks if the reference ontology is locally available, and if not it is downloaded from the given URI. Finally, a request for matchmaking is addressed to the inference engine running in background.

A basic example of a DIG description is shown in Listing 1, referred to the *S. Maria Maggiore* cathedral in Barletta. In addition, Listing 2 reports on an OSM excerpt of the same semantically annotated POI where semantic annotation has been compressed. It is important to remark that in these examples we used DIG because the matchmaker [4] embedded in our current prototype only supports that interface. However, the proposed approach is general and can work with any other Semantic Web language. A relevant objective for future developments of the prototype is the migration of the matchmaker component to a mobile-optimized inference engine, supporting the most recent standards: OWL 2 as ontology language and OWLlink⁸ as reasoner interface.

4 Case Study: Where Are You Going?

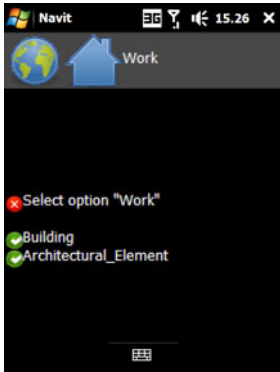
The extended *Navit* navigation system was tested in a case study concerning discovery of cultural heritage resources in the Apulia region. Hereafter implementation details are briefly reported and a simple system usage example is described, in order to highlight main features of the proposed approach.

Resource annotation. Map data were collected and added to OpenStreetMap for an uncovered portion of the city of Barletta. An \mathcal{ALN} ontology for description of cultural heritage resources –not reported here due to space constraints– was developed. Several POIs in the Apulia region were annotated w.r.t. it, with highest concentration in Barletta.

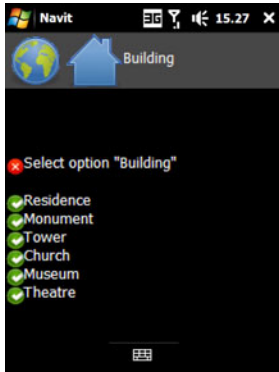
Navit evolution. Support for semantically annotated POIs (see Section 3.2) and semantic-based discovery was added to Navit. A very basic cross-platform GUI was developed for ontology browsing and semantic request composition: its purpose was only to allow testing of the solution, while a more sophisticated and user-friendly interface is planned for subsequent system releases. Microsoft Windows Mobile was selected as reference platform for evaluation. *MaMaS-tng* [4] matchmaker was ported to Windows Mobile and run as a server process. The Navit process acts as a client, interfacing to the inference engine via a local socket using the DIG interface.

Semantic-enhanced POI search. *Isabel is in Barletta for the first time, and she would like to visit interesting places. She is particularly fond of modern artistic palaces.* She launches Navit on her smartphone and starts a new semantic-based POI discovery, selecting “cultural heritage” as resource domain. The user can browse ontology classes and properties in a dynamically-generated hierarchical GUI structure. She is not only able to choose from a taxonomy of landmark types (see Figure 3(a), 3(b) and 3(c)) like in traditional navigation

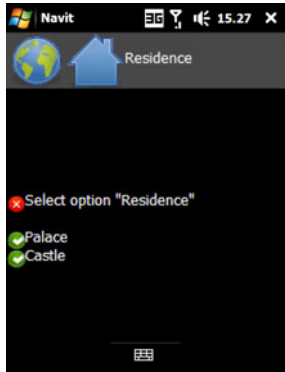
⁸ OWLlink: Structural Specification, Version 1.0, Working Group Recommendation, 16 November 2009, <http://www.owllink.org/owllink-20091116/>



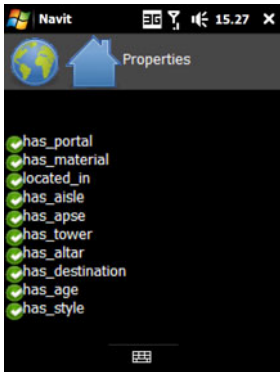
(a)



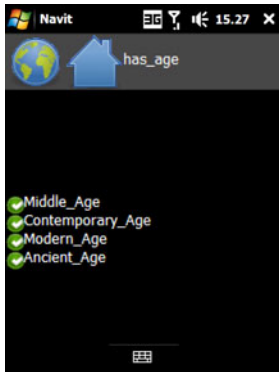
(b)



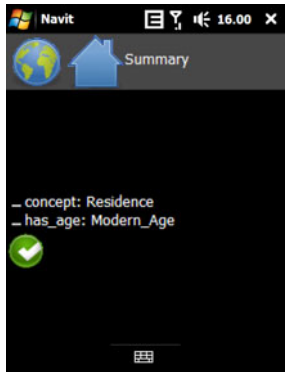
(c)



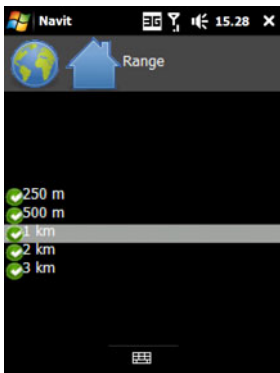
(d)



(e)



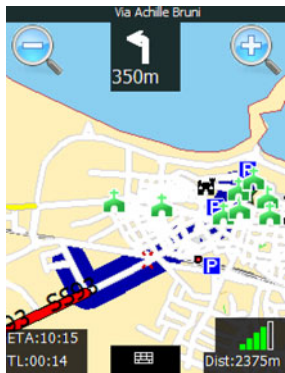
(f)



(g)



(h)



(i)

Fig. 3. Using the semantic-enhanced navigation system

systems, but also to express the desired resource properties (Figure 3(d)) in order to make her search more meaningful and accurate. In our example, after selecting the *Palace* POI class, she sets her constraint on monument age by navigating the appropriate portion of the ontology model (Figure 3(e)). In our very short example, the user ignores further available properties, such as style, materials and size. The final semantic request is summarized in Figure 3(f) and corresponds to the expression (in DL notation for the sake of conciseness): $Residence \sqcap \exists has_age \sqcap \forall has_age.Modern_Age$. The user can review and eventually confirm it. In addition to the semantically annotated expression, the user can insert a maximum acceptable resource distance from her location, as reported in Figure 3(g).

Results review and selection. A Knowledge Base is instantiated in the embedded semantic matchmaker. It consists of the reference domain ontology, of the user request and of POI annotations in the search range. Now inference services for semantic matchmaking are applied. After receiving the reply of the matchmaker, the navigation system computes the utility function, as explained in Section 3.1. In our example, Figure 3(g) reports the final outcome. Only one POI in Barletta, *Palazzo della Marra*, is a full match for the user request, because it is described as a modern age palace –which is a subclass of residence in the reference ontology– in Baroque style with one portal: $Palace \sqcap \exists has_style \sqcap \forall has_style.Baroque \sqcap \exists has_age \sqcap \forall has_age.Modern_Age \sqcap = 1has_portal$. The next best matching options are *Castello Svevo* and *Paraticchio*, which are a medieval castle and a medieval tower, respectively. Their good score is due to the fact that palaces and forts are sibling concepts in the reference ontology. Churches and museums, on the other hand, have higher semantic distance, because of their different nature and purpose. *Isabel selects Palazzo della Marra from the result list and navigation begins.* Figure 3(i) shows the planned route to the destination; the other POIs in the result list are also marked on the map for further reference.

5 Conclusion

We have presented a novel semantic-based navigation approach enabling knowledge-oriented path calculation, which takes into account needs and profile of a user planning a travel. The system we proposed is fully based on open source maps referred to the *OpenStreetMap* project, which have been semantically enriched to make possible advanced matchmaking services.

Improvements are planned for the embedded matchmaker in terms of performance and support to latest Semantic Web standards. Future work will also aim at extending the proposed system with further vehicle and road information retrieval, including traffic and weather conditions detected by means of Wireless Sensor Networks and OBD-II On-Board Diagnostics system interfaces. Semantic-based map annotation will be facilitated by integrating an ontology-based editor within the OpenStreetMap one. Finally, a throughout experimentation will be carried out in order to improve GUIs and navigation experience recalling user’s feedbacks.

Acknowledgments

The authors acknowledge partial support of Apulia Region Strategic Project PS 025 - Processes and technologies supporting quasi-markets in logistics.

References

1. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2002)
2. Bechhofer, S., Möller, R., Crowther, P.: The DIG Description Logic Interface. In: Proceedings of the 16th International Workshop on Description Logics (DL 2003), Rome, Italy, September 2003. CEUR Workshop Proceedings, vol. 81 (2003)
3. Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., Tinelli, E.: A non-monotonic approach to semantic matchmaking and request refinement in e-marketplaces. *International Journal of Electronic Commerce* 12(2), 127–154 (2007)
4. Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: A System for Principled Matchmaking in an Electronic Marketplace. *International Journal of Electronic Commerce* 8(4), 9–37 (2004)
5. French, R.: Maps on Wheels – The Evolution of Automobile Navigation. In: Akerman, J. (ed.) *Cartographies of Travel and Navigation*, pp. 260–290. The University of Chicago Press, Chicago (2006)
6. Leshed, G., Velden, T., Rieger, O., Kot, B., Sengers, P.: In-car gps navigation: engagement with and disengagement from the environment. In: Proceedings of the Twenty-sixth Annual SIGCHI Conference on Human Factors in Computing Systems, pp. 1675–1684. ACM, New York (2008)
7. Ruta, M., Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic based collaborative p2p in ubiquitous computing. *Web Intelligence and Agent Systems* 5(4), 375–391 (2007)
8. Scioscia, F., Ruta, M.: Building a Semantic Web of Things: issues and perspectives in information compression. In: *Semantic Web Information Management (SWIM 2009)*, Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), pp. 589–594. IEEE Computer Society, Los Alamitos (2009)

An Approach to Semantic Information Retrieval Based on Natural Language Query Understanding

Beniamino Di Martino

Second University of Naples
Dipartimento di Ingegneria dell' Informazione
beniamino.dimartino@unina.it

Abstract. In this paper we present an approach to semantic based Information Retrieval of semantically annotated documents, based on natural language understanding of the query, on its semantic representation through an OWL query ontology, and on exact and approximate retrieval of semantically annotated documents, through SPARQL inference and structural graph matching.

1 Introduction

Semantic Web is increasingly gaining momentum as a viable model (and accompanying technology) to represent and manage content and knowledge on the Web at the semantic level. One of the web activities and technology that Semantic Web promises to deeply renovate is Information Retrieval, especially when documents and web sites and resources (services) are described at the semantic level, with use of OWL-based *semantic annotations* [1,14].

In order to be able to express their information need, users of current Semantic Web technology and tools need to be familiar with Ontology languages' syntax (such as RDF and OWL), with formal query languages (such as RDQL and SPARQL), and with schema and dictionaries of target ontologies. In order to overcome such a gap, interfaces based on Natural Language Processing (NLI – *Natural Language Interfaces*) might be effective in expressing the information need, and thus in querying ontologies and semantic annotations in an intuitive and familiar mean, hiding to the non-expert user the complexity of formal grammars of ontologies, semantic annotations and query languages.

A number of problems need to be faced when addressing such approach; they include: the complexity and intrinsic ambiguity of the natural language, which limits the viability of grammar based parsers and stocastical parsers as well; the difficulty in translating the user query into a formal query expressed in formal languages such as SPARQL; the need to map terms expressing concepts and relations used in the user query with corresponding components of, possibly standard, domain ontologies; the need to map the user query with semantic based descriptions of the documents and web pages: semantic annotation models and tools are under development and still not mature enough.

In this paper we present an approach to semantic based Information Retrieval of semantically annotated documents and web sites, and a prototypical tool, which tries to tackle these problems. It is based on natural language query processing at syntactical level, from which semantic patterns are heuristically determined, and are matched (using graph based matching algorithms) against domain ontologies, in order to extract an ontology based representation of the user query (*query ontology*). It is then,

after possible refinement from the user, either translated into SPARQL query in order to perform an exact retrieval of documents annotated with the instances found, either matched against the semantic annotations which represents the documents' meaning, in order to perform an approximate retrieval, ranked with matching measures.

The paper is structured as follows: section 2 provides with a brief overview of methods and approaches to semantic based information retrieval and semantic annotation; section 3 describes the defined technique and developed architecture, section 4 illustrates the implemented prototype tool, *Semantic Searcher*, through an example.

2 Background and State of the Art

Semantic based models and systems for Information Retrieval try to augment (or in some cases to replace) traditional IR technology with Knowledge Engineering technology, in order to solve or at least improve the classical trade-off “precision vs recall” which traditional IR methods suffer [2]. Several approaches to IR based on semantics exist, including *profile-based* approaches (where user preferences and interest domains are collected and represented, and are measured for similarity with suitable representations of document content, see e.g. [3]), *collaborative* approaches (where similarity is measured among users, instead than document content, see e.g. [4]), and *classification* approaches (where documents are classified, automatically or manually, according to predefined categories/taxonomies, or are unsupervisedly clustered, see e.g. [5,6]). The approaches based on Ontologies relies on standardized machine-readable representations of knowledge, domain oriented but also *upper-level*, in standard and open languages like RDF and OWL. Main efforts and systems include (among others): OntoSeek [7], On2Broker [8], WebSifter [9], Score [10], Intellizap [11], Swoogle [12]. We cannot provide details on these systems here for reasons of space. In [13] a detailed overview and comparison of those systems is provided. Ontology-based annotation of documents and media is one of the most active research topics in making documents machine understandable and allowing for effective document information retrieval. First examples of documents annotated using metadata can be found in the (KA)2 initiative [14]. In [15] a detailed state-of-the-art overview is provided and a novel annotation model and a prototype tool is proposed.

3 A Technique and an Architecture for Semantic Information Retrieval

The technique we have devised is composed by the following phases:

1. Syntactical analysis of the query, expressed in Natural Language
2. Building from the syntactic tree of a Query graph (by means of Triple extraction)
3. Mapping of Query graph with domain ontologies
4. Query ontology graph production and user validation
5. Semantic querying with (alternatively):
 - a. Ontology query translation in a semantic query language (e.g. SPARQL) and running of a SPARQL engine (such as Jena ARP or Pellet)
 - b. Graph matching of query ontology with semantic annotations
6. Visualization of results

The defined steps are visually shown in figure 1.

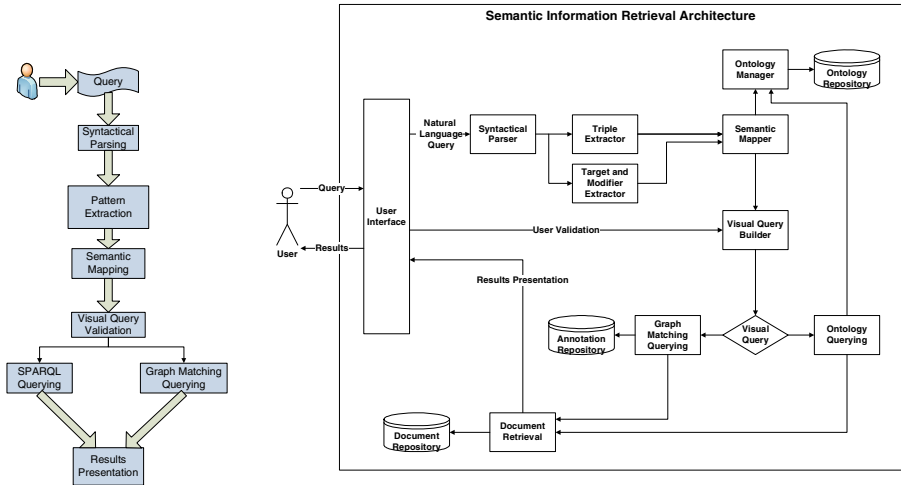


Fig. 1. The Semantic Information Retrieval procedure and Architecture

The above defined technique is detailed by means of the architecture depicted in figure 1. It consists of the following components:

- **User Interface:** it allows the user to input the query phrase (in natural language) and to drive her/him through the following phases.
- **Ontology Manager:** it manages the local ontologies and semantic annotations, by ensuring consistency and persistence.
- **Syntactical Parser:** it performs lexical and syntactic analysis of the user query.
- **Pattern Extractor:** such component extracts from the syntactic tree a skeleton composed by all syntagms found in the query, detects the noun syntagms and then detects all the relationships among them (represented by verb, adjective and other noun syntagms), thus extracting a set of patterns in the form of triples.
- **Semantic Mapper:** such component maps the patterns extracted from the query with components of the domain ontologies through graph matching algorithms.
- **Visual Query Builder:** this component presents the extracted query ontology to the user, who can possibly refine it, by browsing the domain ontologies from which the query ontology has been extracted, select other concepts and/or relationships from them, and add to the query ontology or replace components.
- **Semantic Querying:** this component performs the translation of the produced query ontology into a SPARQL query, and executes it, finding all the instances which fulfill the semantic query.
- **Graph Matching Querying:** this component (alternative to the previous one) performs the search in a different way, by doing a graph matching among the query ontology graph, and all the semantic annotation graphs managed by the ontology manager.
- **Document Retrieval and Presentation:** this component present the user with the results of the SPARQL querying or the graph matching querying, that is the

found instances, the annotations which present those instances as a component, and the annotated documents.

In the following subsections we illustrate in more details the functionalities of some of the main components of the architecture.

3.1 Ontology Manager

The *Ontology Manager* component manages the local domain ontologies and representations of semantic annotations. It maintains their persistence and versioning with respect to multiple accesses and modifications. It indexes the terms used to define concepts, instances and relations, in order to allow for a faster retrieval of the domain ontologies relevant to the user query, when the Semantic Mapping is performed. Ontologies, represented in OWL language, are parserized and then the terms indexed after stopwords removal and stemming.

3.2 Syntactical Parser

This component parses the natural language query, performing lexical and syntactical analysis, and outputs the Syntax tree of the query phrase. We have implemented a Context Free Grammar for the Italian language, making use of Prolog and its feature to easily implement CFGs through built-in Definite Clause Grammars.

3.3 Pattern Extractor

This component extracts from the Syntax tree heuristic patterns, to be then matched against domain ontologies.

The patterns we have defined are RDF-like triples, composed from syntax tree syntagms, of the form $\langle \textit{subject}, \textit{predicate}, \textit{object} \rangle$, where *subject* and *object* are composed from noun syntagms (in the following named as NS), while *predicate* can be composed by one or multiple Verb syntagms (VS), prepositional syntagms (PS), adjective syntagms (AS), conjunction syntagms (CS) and noun syntagms (NS) as well, or can be a predefined OWL relationship. The reason for choosing such triples as patterns is because OWL has as main component the RDF triples, which represent relationships (the predicate) between two classes or class instances (the subject and object). Thus matching patterns against the domain ontologies is equivalent to find subgraph isomorphisms between the pattern triples and subgraphs triples of the ontologies.

The pattern triples are identified from typical aggregations of syntagms in the syntax tree, with optional constraints. The most typical syntagms are of the form:

NS-VS-NS

NS-PS-NS

NS-NS-NS

NS-AS-NS

of which the first is the most frequent, but also:

SN-(SV,SN)-SN

or SN alone, composed by an adjective and a noun.

For instance, the pattern NS-VS-NS with VS an auxiliary verb (e.g. the italian verbs *essere* (to be) or *avere* (to have)) can be mapped to a pattern RDF triple where the predicate part is the predefined OWL relationship *subClassOf*, as in the example:

La birra <NS> è <VS> una bevanda <NS>
 (beer is a drink)

which is mapped to the RDF triple shown in figure 2.



Fig. 2. An RDF triple pattern

Another example is the same pattern NS-VS-NS with VS auxiliary verb and the first NS containing a proper noun. It can be mapped to the RDF triple where the predicate part is the predefined OWL *instanceOf* relationship, as in the example:

La Lager <NS> è <VS> una birra <NS>
 (Lager is a beer)

which is mapped to the RDF triple shown in figure 3.

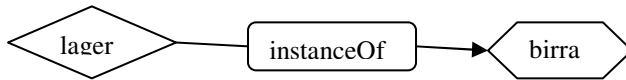


Fig. 3. Another RDF triple pattern

The pattern NS alone, composed by a noun and a qualificative adjective, can be mapped to two different RDF triple patterns; infact the adjective can be a specifier or an attribute of a noun. In the first case, the predicate of the RDF triple can be the *subClassOf* OWL relationship, while in the other case it can be any relationship or the *instanceOf* relationship, thus in the second case the predicate part is left undefined, and it will be “filled” when a match is found with a corresponding triple of the ontology graph by the graph matching algorithm. As an example:

La birra rossa <NS>
 (the red beer)

is mapped to the two RDF triples shown in figure 4.

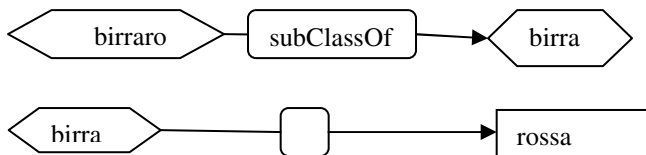


Fig. 4. Two RDF triples from <NS>

The pattern NS-VS-NS-NS where the VS is an auxiliary verb, is mapped to an RDF triple where the predicate part is composed by concatenation of the middle pair (VS,NS), as in the example:

La birra <NS> ha <VS> come ingrediente <NS> il malto <NS>
 (beer has malt as ingredient)

is mapped to the RDF triple shown in figure 5.

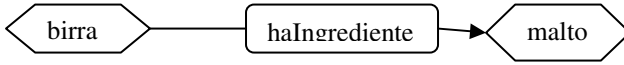


Fig. 5. An RDF triple from NS-VS-NS-NS

As a final example, the pattern NS-PS-NS where the prepositional syntagm PS is the italian preposition *di* and the second NS is composed by a noun and an adjective, can be mapped to two RDF triple patterns; infact the preposition *di* coupled with a noun can have a meaning of a specifier or an attribute. In the first case, the predicate of the RDF triple can be the *subClassOf* OWL relationship, while in the second case it can be the *instanceOf* relationship, as in the example:

La birra <NS> di <PS> colore rosso <NS>
 (the beer of red color)

Is mapped to the two RDF triples shown in figure 6.

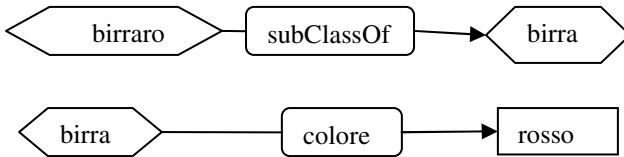


Fig. 6. Two RDF triples from NS-PS-NS

3.4 Semantic Mapper

This component tries to match the pattern triples found by the Pattern Extractor with the components of the domain ontologies stored in the Ontology manager.

First of all, the relevant ontologies are detected by traditional search of terms present in the query within the ontologies’ indices, possibly expanded by their synonyms (we use Multiwordnet [16] thesaurus at this purpose). Each of these ontologies is then parsed and a graph representation for each of them is obtained (OG – Ontology Graph). The set of triples detected by the Pattern extractor is also represented as a single graph (QG – Query graph). Then a structural matching algorithm, based on subgraph isomorphism at the structural level and on string similarity and use of synonyms at the node - link level, is applied to each couple (OG, QG). The details of the matching algorithm and the developed procedure cannot be illustrated here for reasons of space, but can be found in [17]. The subgraphs matched, weighted with a similarity measure, are returned to the user, through the following component, the Visual Query Builder, in form of a *Query Ontology*.

3.5 Visual Query Builder

This component visualizes the matched subgraphs to the user, in the form of a *Query Ontology*, which represents the meaning of the natural language query, together with the syntactic tree and the matched domain ontology(ies). The user can possibly refine it, by browsing the domain ontologies, select other concepts and/or relationships from them, and add to the query ontology or replace components of it. The resulting refined query ontology is then passed to the Semantic Querying component.

3.6 Semantic Querying

This component performs the actual retrieval of semantically annotated documents, by means of the constraints defined with the Query Ontology automatically produced from the natural language query, possibly refined by the user. The retrieval can be performed in two ways:

- with an *exact* approach, based on translation of the ontology into a SPARQL query, which, executed by an engine, returns the instances which exactly fulfill the constraints of the query, and subsequently the semantic annotations using those instances and the linked documents;
- with an *approximate* approach, based on graph matching of the ontology query with the semantic annotations linked to the documents. The matches are weighted and thus the documents corresponding to the matched annotations can be scored in order of similarity with respect to the query ontology.

The translation of the query ontology into a SPARQL query cannot be illustrated here for space limitations; please refer to [18] for details.

4 Semantic Searcher: A Prototype Tool for Semantic IR

We have implemented the procedure and architecture defined in the previous section, and the result is the prototype tool *Semantic Searcher*. It has been developed by using the Java development environment Eclipse, by utilizing Jena [19] parsers for OWL and SPARQL, Jena SPARQL engine ARQ, and reasoner Pellet [20]. The domain ontologies' indexing has been implemented by utilizing the Information Retrieval library *Lucene*. Multiwordnet [16] synonyms thesaurus has been utilized for expanding the graph matching and the ontologies' indexing. Natural Language parsing and conceptual graph representation has been performed (for the Italian Language) with use of SWI Prolog interpreter. We present in the following the tool's functionality by means of an example query process, performed in Italian language (we provide any-way English translation of all the phases of the querying procedure).

The chosen example domain is *beers*. As an example user wants to know which beers are made with a given ingredient and have a given colour. Then she/he expresses the following query: "Which beers of red colour have malt of wheat as ingredient?" (in Italian: "*Quali sono le birre di colore rosso che hanno come ingrediente il malto di frumento?*"). The natural language query is initially analyzed by the *Syntactical Parser* which outputs the syntactical tree (it can be visualized in figure 8). The parse tree is then inputted to the *Semantic Mapper* which performs the mapping to the domain ontologies available to the *Ontology Manager*. Once performed the mapping,

the tool returns (in the *Result Matching* panel) the matched subgraphs, together with quantitative results of the graph matching, as shown in figure 7.

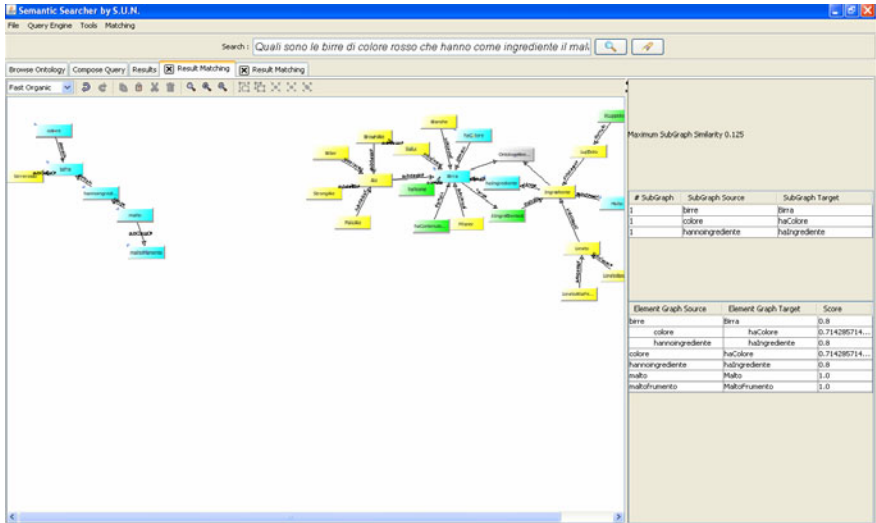


Fig. 7. Results of the matching phase

The best matched subgraph is then translated into a *Query Ontology*, which represents the meaning of the natural language query, and presented to the user, together with the syntactic tree and the matched domain ontology(ies) (in the *Browse Ontology* panel) as shown in figure 8, together with the query syntax tree.

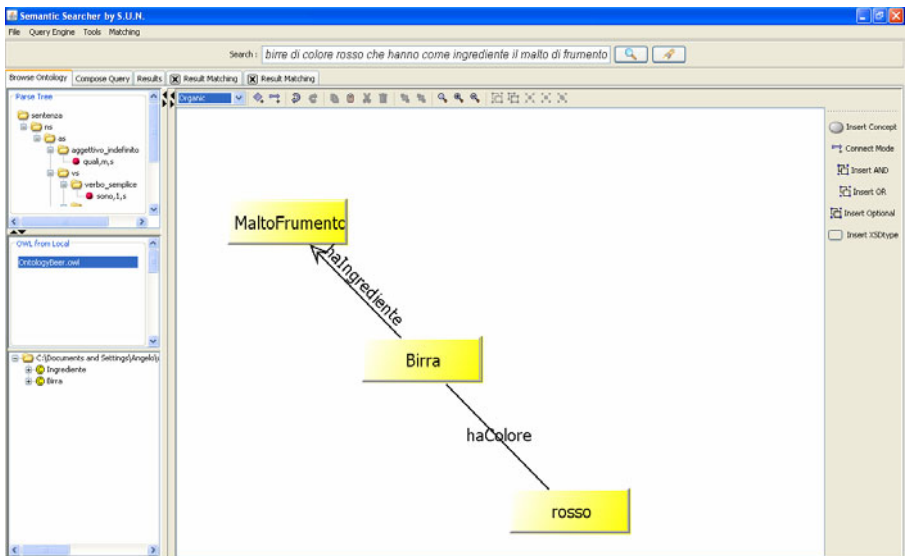


Fig. 8. The extracted query ontology, representing the meaning of the natural language query

As shown in the figure, the query ontology extracted from the example phrase is composed by three concepts - *Birra* (beer), *MaltoFruento* (WheatMalt) and *Rosso* (red) - linked by the relations *haIngrediente* (has ingredient) and *haColore* (has color), which are component of the domain ontology (OntologyBeer.owl) which has matched, as shown in left panel of figure 8.

Once visualised the Query ontology, the user can possibly refine it, by browsing the domain ontologies from which the query ontology has been extracted, select other concepts and/or relationships from them, and add to the query ontology or replace components by means of “drag and drop” action, as shown in figure 9.

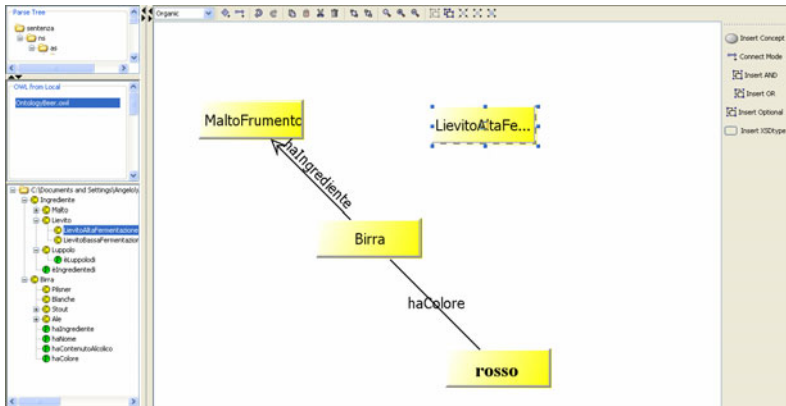


Fig. 9. “Drag and drop” from the domain ontology to the query ontology

Once completed the refinement of the query ontology, it is transferred to the *Compose Query* panel, where the semantic querying is performed in order to find the relevant semantically annotated documents. The user can choose to perform an (exact) semantic querying with a SPARQL engine – she/he can select the kind of semantic engine used to perform the semantic querying (the Jena SPARQL engine ARQ or reasoner Pellet) – or to perform a structural matching between the query ontology and the semantic annotations, with the same algorithms used in the previous phase.

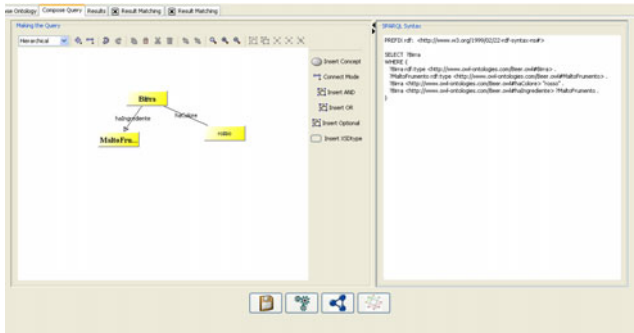


Fig. 10. Generated SPARQL query

In figure 10 it is shown the generated query in SPARQL language (which can also be manually modified), while figure 11 shows the returned instances found.

The screenshot shows a web interface with a top navigation bar containing 'Browse Ontology', 'Compose Query', 'Results', and two 'Result Matching' buttons with checkboxes. Below this is a section titled 'Results of Search'. The main content area displays a list of search results for the term 'Birra':

Birra
http://www.owl-ontologies.com/Beer.owl#Lambic
http://www.owl-ontologies.com/Beer.owl#Berliner_weisse
http://www.owl-ontologies.com/Beer.owl#Augustijn
http://www.owl-ontologies.com/Beer.owl#pilsnerUrquell

Fig. 11. Instances found from the semantic engine

The instances found (*Lambic*, *Berliner_weisse*, *Augustin* and *PilsenerUrquell*) satisfy the query constraint: they're red beers with WheatMalt as one of the ingredients. The search is *exact*, thus there is no need for ranking.

By clicking on an instance, the corresponding semantic annotations which have used that instance can be visualized, as shown in figure 12, and by clicking on an annotation, the corresponding annotated document is visualized, as shown in fig. 13.

The screenshot shows a web interface with a top navigation bar containing 'Browse Ontology', 'Compose Query', 'Results', and several 'Result Matching' buttons with checkboxes. Below this is a section titled 'Results of Search'. The main content area displays a table of annotations for a selected instance:

SubGraph	Annotation File
0.4	[casi di studio]annotation_repository/Annotation09.xml
0.2	[casi di studio]annotation_repository/Annotation08.xml
0.2	[casi di studio]annotation_repository/Annotation03.xml

Fig. 12. Annotations containing a selected found instance

The screenshot shows a complex web interface with a top navigation bar containing 'Browse Ontology', 'Compose Query', 'Results', and several 'Result Matching' buttons with checkboxes, and a 'CiriFPA' button. The main content area is divided into several sections:

- Ontology:** Shows the URL <http://www.arsabira.it/Birra/Birra.aspx?d=48>.
- CLASS BROWSER:** Displays an asserted hierarchy for the project, including classes like *owl:Thing*, *Birra*, *Lievito*, *base:fermentazione*, *Luppolo*, *Malto*, *MaltoFrumento*, *MaltoOrzo*, *owl:Individual*, *owl:SubClassOf*, *owl:Nothing*, *probege:ExternalResource*, *rdfs:Literal*, *rdfs:Property*, *rdfs:SubProperty*, *rdfs:Class*, and *rdfs:Container*.
- INSTANCE BROWSER:** Shows instances for the class *base:fermentazione*, including *base:fermentazione_Individual_2* and *lievitolibasso*.
- Document Content:**
 - Metadata:**
 - Nome:** **Birra**
 - Produttore:** Beamsch
 - Nazione di appartenenza:** Irlanda
 - Gradazione:** 4,3% Vol.
 - Stile:** Porter/Stout - Stout
 - Colore:** scura
 - Note:** Da quando la gloriosa tradizione delle stout ha invaso con decisione il mercato, molti prodotti hanno cercato di scendere a compromessi con i gusti di un pubblico sempre meno "selezionato". Una gradita eccezione è questa birra prodotta a Cork fin dal 1792, al contrario di molte "colleghe", la Beamsch ha infatti mantenuto un gusto secco e deciso, oltre ad un corpo generosissimo.
 - Main Text:** Ad un primo approccio, salta all'occhio il colore bruno opaco che, unito ad una schiuma assai corposa, rende alla perfezione la cremosità che accarezzerà il palato. Insomma, una birra che non mesole l'aroma ricorda il pane appena bruciato, con sentori di **pepe e cacao** forte, gusto e struttura sono estremamente soddisfacenti; il finale è decisamente più secco e liquoroso di quello di molte stout.
 - Footer:** L'abbinamento spesso indicato per questa birra è una cena a base di frutti di mare, ma l'occasione ideale è forse una chiacchierata con gli amici, la sera, magari accompagnata da un giro di bruschette ben tostate, possibilmente, però, senza saponi troppo imponenti.
- Image:** A placeholder for an image with the text 'image not available' and '[AB].it'.
- Filter Panel:** On the right, there is a 'Show/Hide' and 'Color' section with checkboxes for 'Ale', 'Birra', 'Luppolo', 'Lievito', 'MaltoFrumento', 'MaltoOrzo', 'Malt', 'Pilsale', 'Thing', 'blanche', 'base:fermentazione', 'brownAle', 'lager', 'pilsner', and 'stout'.

Fig. 13. The annotated document

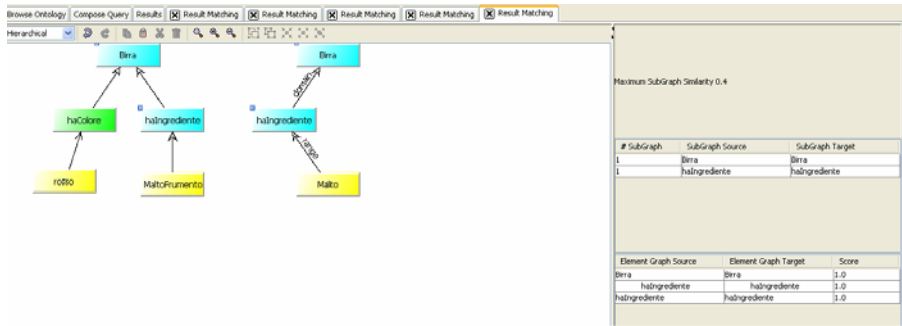


Fig. 14. Results of Graph Matching between query and an annotation

The alternative to the SPARQL semantic search is the search of annotations by graph matching with the query ontology. Figure 14 shows the results of matching the example query ontology graph with a semantic annotation graph, which represents a given annotation.

As shown in figure 14, the match (which is not exact since the applied algorithm is subgraph isomorphism) is measured with the similarity measure defined for subgraph isomorphism algorithm. This measure can be used to rank the results found (currently each matched annotation is shown on a separate panel, as seen in figure 14, in order to browse and inspect each of the graph matches).

5 Conclusions

We have presented a technique, an architecture and a prototypical implementation for Information Retrieval based on natural language understanding of query, on representation of semantics of the user need and of the documents by means of OWL ontologies and semantic document annotations, and on structural graph matching. It shows that an exact match between user need and semantics of document can be obtained, if documents are semantically annotated, with ontological representation of user need, obtained automatically with NLP and validated by the user, and with use of query language SPARQL and inference engine Pellet. Inexact matching can be obtained instead with graph matching of ontology based annotations and representation of user query. Similarity measures can be defined for the inexact graph matching, and used to rank the search results. Definition of such a measure, and validation with experimental campaign and analysis is subject of future work.

Acknowledgements

The work described has been supported by the PRIST project “Fruizione assistita e context aware di siti archeologici complessi mediante dispositivi mobili”. I would like to thank Angelo Martone for the prototype implementation.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic web. *Sci. Am.* 1(1), 68–88 (2000)
2. Baeza-Yates, R., Navarro, G.: Integrating contents and structure in text retrieval. *SIGMOD Rec.* 25(1) (1996)
3. Ackerman, M., et al.: Learning Probabilistic User Profiles - Applications for Finding Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities. *AI Magazine* 18(2), 47–56 (1997)
4. Bollacker, K.D., et al.: Discovering Relevant Scientific Literature on the Web. *IEEE Intelligent Systems* 15(2), 42–47 (2000)
5. Koshman, S., Spink, A., Jansen, B.J.: Web Searching on the Vivisimo Search Engine. *Journal of the American Society For Information Science And Technology* 57(14), 1875–1887 (2006)
6. Labrou, Y., Finin, T.: Yahoo! as an ontology: using Yahoo! categories to describe documents. In: *Procs of the eighth intl. conference on Information and knowledge management, Kansas City, USA*, pp. 180–187 (1999)
7. Guarino, N., et al.: OntoSeek: Content-based Access to the Web. *IEEE Intelligent Systems* 14(3), 70–80 (1999)
8. Fensel, D., et al.: On2broker: Semantic-Based Access to Information Sources at the WWW. In: *Proceedings of the World Conference on the WWW and Internet (WebNet 1999), Honolulu, Hawaii, USA*, pp. 25–30 (1999)
9. Kerschberg, L., Kim, W., Scime, A.: WebSifter II: a personalizable meta-search agent based on weighted semantic taxonomy tree. In: *Proc. of Int. Conf. on Internet Computing* (2001)
10. Sheth, A., Bertram, C., Avant, D., Hammond, B., Kochut, K., Warke, Y.: Managing Semantic Content for the Web. *IEEE Internet Computing* 6(4) (2002)
11. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppim, E.: Placing Search in Context The Concept Revisited. *ACM Transaction on Information Systems* 20(1), 116–131 (2002)
12. Finin, T., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A.: Swoogle: Searching for knowledge on the Semantic Web. In: *Proc. of Intl. Conf. on Artificial Intelligence* (2005)
13. Di Martino, B., et al.: Stato dell' arte ed analisi delle tecnologie e degli strumenti per l'Information Retrieval. Technical Report TR2.4.4, Progetto LC3 (2009)
14. Benjamins, V.R., Fensel, D., Decker, S., Gomez-Perez, A. (KA)2: building ontologies for the internet: a mid term report. *Intl. J. of Human Computer Studies* 51, 687–712 (1999)
15. Moscato, F., Di Martino, B., Venticinque, S., Martone, A.: OverFA: A collaborative Framework for Semantic Annotation of Documents and Web Sites. *International Journal of Web and Grid Services (IJWGS)* 5(1), 30–45 (2009)
16. Pianta, E., Bentivogli, L., Girardi, C.: MultiWordNet: Developing and Aligned Multilingual Database. In: *Proceedings of the First International Conference on Global WordNet, Mysore, India, January 21-25*, pp. 293–302 (2002)
17. Di Martino, B.: Semantic Web Services Discovery based on Structural Ontology Matching. *International Journal of Web and Grid Services (IJWGS)* 5(1), 46–65 (2009)
18. Di Martino, B., et al.: Definizione di una Architettura per il Semantic Information Retrieval, Technical Report TR2.4.9, Progetto LC3 (2009)
19. Jena – A Semantic Web Framework for Java, <http://jena.sourceforge.net/>
20. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner, Tech. Rep. CS 4766, University of Maryland, College Park (2005)

Slicing Linked Data by Extracting Significant, Self-describing Subsets: The DBpedia Case

Michele Minno¹, Davide Palmisano², and Michele Mostarda²

¹ Pro-netics

michele.minno@pronetics.it

² Fondazione Bruno Kessler

{palmisano,mostarda}@fbk.eu

Abstract. The Linked Data cloud is a huge set of data sources, the same objects being instances of many different ontologies and being described by different overlapping concepts and properties. This paper shows an innovative approach to represent in an unambiguous and homogeneous way instances of such large, highly unpredictable, linked ontologies. Instead of making use of external, static and ad-hoc devised knowledge bases, the approach followed in this paper leverages an existing ontology in the Linked Data cloud to univocally identify all the instances of all linked ontologies. Following this kind of approach, different views over the same set of instances can be devised, depending of which spot of the cloud we choose to see things through.

Keywords: LinkedData, Ontology, SKOS.

1 Introduction

The Linked Data cloud is a set of interlinked, loosely-coupled ontologies in a continuous incremental expansion. The positive aspects of this phenomenon are huge and easily understandable: knowledge formal representation grows along with their users', reflecting the typical proper nature of the Web, being a place where 'things happen', more than where things happened elsewhere are registered and flatly listed. The drawback of all this is that an actor (typically: a human through a software interface and service) who would like to access this big multi-layered knowledge base will easily loose the grasp over it. The approach here presented allows to get a partition of the whole ontology set, according to a reasonably fixed subset of it. This partition, informally depicted as a 'set of slices', adds a semantically meaningful description of the resources contained in the Linked Data cloud (the instances, we can say), because obtained through a subset of the Linked Data cloud itself. The use case explored in the following of the paper relates to subjects assigned to things, like 'actor' or 'business man' to a person. With the algorithm here presented we show how to obtain a partition of all Linked Data instances in terms of subjects, in which thus every instance

has got a single meaningful subject assigned to it. For prototyping purposes the DBpedia RDF ontology is used, more specifically a relational view of it. The rest of the paper is structured as follows. Section 2 describes the problem while section 3 justifies the adopted solution and details its main aspects, jointly with the algorithms used. Section 4 suggests some variants and applications. Conclusions and acknowledgements close the paper.

2 The Linked Data Slicing: Problem Statement

The actual Linked Data slicing problem can be reduced to the simpler one of partitioning a set of points placed in an n -dimensional space in a set of m equivalence classes (see figure 1). In mathematics, the equivalence class of an element a in the set X is the subset of all elements in X which are equivalent to a . In our case the set X comprises all Linked Data resources. We'll constrain ourselves to consider just DBpedia¹ for sake of simplicity and because it is one of the biggest and referred ontology within the Linked Data cloud.

The initial condition of the problem is the chaotic situation where each DBpedia resource belongs to several classes (not equivalence classes, but just sets), by means of being instance of several concepts and participating in many properties, which belong in a scattered fashion to different ontologies in the Linked Data cloud, according to the Linked Data best practices. The resources are the points represented in the cloud of figure 1, ideally standing for the whole Linked Data cloud. The 3-dimensional space stands for the different ontologies describing the same instances from different point of views, like geo-location, musical genres, and so on. Our aim is to see these points through a filter that is simply one of the ontologies of the cloud, represented in the figure by the external rectangle. This rectangle is the partition we want to achieve, where each point is of a certain color (concept or property value), even though in the chosen ontology in the cloud it participates of many different colored sets. As a first step of our approach, a subset of this Linked Data cloud has to be chosen, in order to divide the resources set in equivalence classes. The mentioned subset must be something that applies to the greatest number of resources, ideally to all of them. So it has to be quite generic and not narrowed to a specific domain. In the DBpedia case, we chose SKOS². SKOS stands for Simple Knowledge Organisation System. It is a very light model to express relations between concepts, basically 'broader' or 'narrower' relations. So if a set of concepts has a very straight hierarchy it would be easily modeled by a SKOS concepts scheme. Each DBpedia resource has at least a SKOS subject associated to it through the *skos:subject* property that links a resource to a subject. Some resources have even several tens of subjects, from the most generic (i.e.: *Living_people*) to the most specific (*Presidents_of_the_United_States*, for the resource: *Barack_Obama*). As it is within the Linked Data cloud, this SKOS categorization is simply another layer describing

¹ <http://dbpedia.org/>

² A good guide can be found at

<http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102>

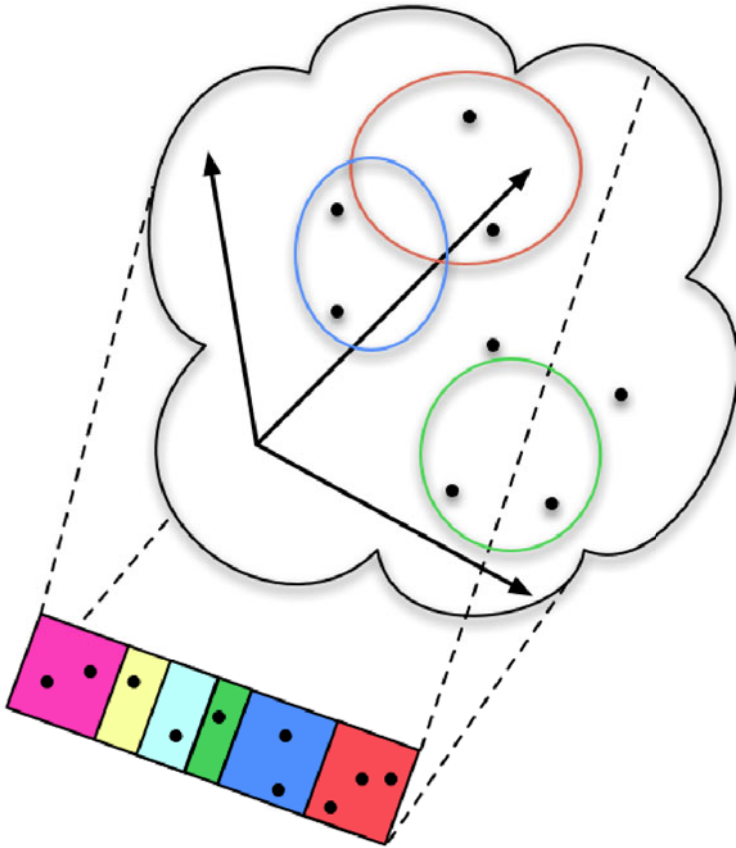


Fig. 1. The 'Slicing' problem

the DBpedia resources, and it doesn't give back much information. The situation is well sketched in the figure 2. To have the whole DBpedia resource set divided in equivalence class we have to achieve instead the situation sketched in figure 3, in which each resource belongs to exactly one SKOS subject. As we are going to see in section 4, the SKOS subject has to be chosen as the most representative one, according to an algorithm explained in the following of the paper.

Just a trivial consideration: choosing the narrowest category as the most representative one is a solution that, even if characterized by a straightforward computational complexity, often leads to undesirable results. Actually such kind of solutions don't take into consideration how the resource is linked with the rest of the data space, losing the opportunity to discover new relationships and links between different instances and concept. In one word, they lose the opportunity offered by such linked knowledge.

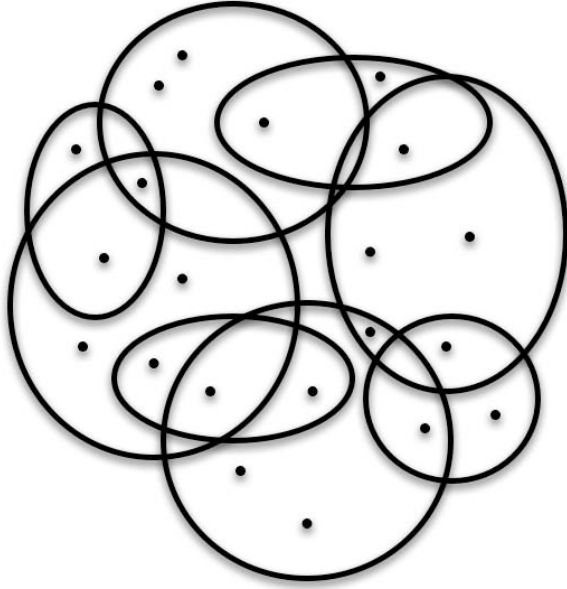


Fig. 2. The SKOS layer

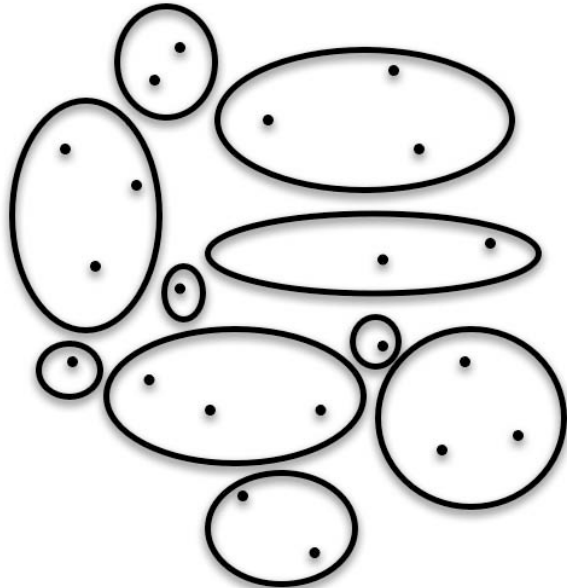


Fig. 3. The SKOS equivalence classes

3 The Relational Snapshot Solution

What we call here the relational snapshot is essentially an index built on an RDF ontology, where each tuple of the relation refers to a resource R in the ontology and can be represented as:

$$(ur, lr, us, ls)$$

where:

ur : URI (Uniform Resource Identifier) of R .

lr : the first one of the values of the property *rdfs:label* in which R participates.³

us : URI of the SKOS subject S describing R .

ls : the value of the property *skos:prefLabel* in which S participates.

in which *rdfs:label* is one of the core properties of RDF Schema⁴, whereas *skos:prefLabel* is one of the properties of SKOS⁵.

The adoption of a solution based on an index natively stored in a relational model is motivated mainly by the following considerations:

Responsiveness of the adopted solution - The values composing each tuple described above could be also on-the-fly computed by a series of SPARQL⁶ queries, without needing to be stored in a relational structure. Still, this holds in principle, but since the result of this slicing process has in a Web based environment its natural use and applications, the responsiveness raises a serious issue about the scalability of any adoptable solution. It comes obvious that a 30-years old consolidated technology is still the most suitable to address such problems, also from a mere computational point of view. Accessing a relational structure has a *LOGSPACE* complexity (this is the complexity of evaluating a First Order Logic query, also expressible in SQL, over a database¹¹), while accessing a graph-like structure as RDF by query languages as SPARQL is a *PSPACE*-complete problem¹², thus leading to intractable query evaluations.

Transparency - The relational view allows the potential end users of any application of the slicing process to access the indexed ontology without regarding how it is effectively stored and structured.

Index adaptability - Every change in the ontology subjected to the slicing should reflect onto the index itself. The adopted solution allows any modification of the relational view simply accessing the stored tuples with SQL. RDF resources cancellation or new insertion could be resolved trivially by one single SQL delete or update instruction on the index.

³ The first one is usually the most used.

⁴ The prefix *rdfs* stands for: <http://www.w3.org/2000/01/rdf-schema#>

⁵ The prefix *skos* stands for: <http://www.w3.org/2004/02/skos/core#>

⁶ The query language for RDF: <http://www.w3.org/TR/rdf-sparql-query>

3.1 A SKOS Based Indexing

Even if the identification of the most representative SKOS subject of a certain resource seems too much related to the perception that each person could have regarding that specific resource, several assumptions could be done around the concept of pertinence of a SKOS subject measured by the degree of similarity between two resource. More precisely,

- given two resources the degree of similarity is the number of SKOS subjects that they have in common and,
- the most linked resource MLR of a given SKOS subject S is the resource that has S among its SKOS subjects and that is referred by the highest number of resources in the ontology (here referred means: having a property P participation like: $P(r, MLR)$, with r any other resource)

With this two roughly defined properties is possible to informally state that the most representative SKOS subject of a given resource R is the one for which the highest similarity degree between its most linked resource and R has been found. A formalization of the algorithm that computes the most representative SKOS subject of a given resource is provided in the rest of this section, using the SPARQL syntax when needed.

3.2 The Most Representative SKOS Subject Identification Algorithm

Hereby follows the formal description of the algorithms that are the foundation of our indexing techniques. Even if a formal analysis of the computational complexity is not provided, this formalization is necessary to make this dissertation more readable and concrete. However, the computation complexity of the algorithm strongly depends on the underlying algorithm used to resolve the SPARQL queries on the targeted ontology.

MRSS: The most representative SKOS subject identification algorithm

Input: a resource URI r

Output: a SKOS subject URI s

```

subjects <- SkosSubjects(r);
integer similarity <- 0;
SKOS subject URI s <- subjects[0];
foreach subject in subjects do
    mostLinked <- MostLinkedResource(subject);
    actualSimilarity <- ResourceSimilarity(r, mostLinked);
    if similarity <= actualSimilarity then
        similarity <- actualSimilarity;
        s <- subject;
    end
end
return s

```

SkosSubjects: The SPARQL query to retrieve the SKOS subjects of a resource

Input: a resource URI *r*

Output: a set of SKOS subject URIs

```
subjects <- ExecuteSPARQLQuery("SELECT DISTINCT ?uri WHERE ?r
skos:subject ?uri")
return subjects
```

MostLinkedResource: The function that retrieves the most linked resource of a given SKOS subject

Input: a SKOS subject URI *subject*

Output: a resource URI *resource*

```
subjectResources <- ExecuteSPARQLQuery("SELECT DISTINCT ?uri WHERE ?uri
skos:subject ?s");
mostLinked <- subjectResources.first();
linkedSize <- GetLinkedResources(mostLinked).size();
foreach resource in subjectResources do
  actualLinkedSize <- GetLinkedResources(resource).size();
  if linkedSize <= actualLinkedSize then
    linkedSize <- actualLinkedSize;
    mostLinked <- resource;
  end
end
return mostLinked
```

GetLinkedResources: The SPARQL query that retrieves all the instances linked to a given resource

Input: a resource URI *resource*

Output: a set of resource URIs *resources*

```
resources <- ExecuteSPARQLQuery(SELECT DISTINCT ?uri WHERE ?uri ?prop ?r)
return resources
```

ResourceSimilarity: The function that measures the degree of similarity between two instances

Input: a resource URI *uri1*, a resource URI *uri2*

Output: an integer *N* ($0 \leq N \leq \text{SkosSubjects}(\text{uri1})$)

```
similarity <- 0
foreach subject in SkosSubjects(uri1) do
  if subject in SkosSubjects(uri2) then
    similarity ++
  end
end
return similarity
```


4 Variation on Theme and Potential Applications

In this paper it has been showed how to slice the whole DBpedia using SKOS. By the way, this is just an example of partition of an ontology instances set that can be achieved. The following notation can be used to abstractly represent a partition:

$$(I, K)$$

where I is the input ontology to be sliced and K is the ontology used to slice I (metaphorically, the knife). Thus several other partitions are possible, depending on which specific view over I is desired. For example, slices on an I containing music artists can be obtained through a K modeling music genres, and so on. This slicing process can be straightforwardly implemented as a service and used in applications where real semantic⁷ data coming from the Web are strongly preferred, but with the wish of avoiding all the real complexity involved in such retrieval (in terms of know-how, technologies and response efficiency). To give an example of a real application that would leverage such kind of services, an input field powered by a semantic autocompletion functionality can be devised. In such kind of field, every suggested completion of the string typed by the user would come with a category, i.e.: (Godfather - Male godparent), (Godfather - Movie), and so on. Moreover, all the suggestions can be grouped by the category to improve the response readability. As we have seen, using the defined MRSS algorithm is possible to couple each resource label with its MRSS label. This allows to achieve the autocompletion process, resolving each substring matching, as the following SQL query shows, where a *< substring >* is matched with the label of an instance coupled with its MRSS:

```
SELECT label, category FROM index
```

```
WHERE label REGEXP (" < substring > %") GROUPBY category
```

In this way the user is enabled to uniquely and explicitly identify instances of an ontology that represent what she/he really has in mind during typing.

5 Conclusions and Future Work

In this paper a Web of Data partition enabling algorithm has been justified, detailed and prototyped using the DBpedia and the SKOS ontologies. The deeper idea behind is that the Linked Data cloud is a special data source, somehow unique under many points of view, which are first of all its collaborative linked nature and its semantic formalization. But unfortunately it has not been so far fully exploited, its potential being hardly understood and appreciated. So this effort may be considered an attempt to see the Linked Data machinery at work,

⁷ Here semantic stands for meaningful, in the most generic acceptance.

producing the high-quality data which are its own strength, but hiding or internally managing all the tricks, subtleties and efficiency lacks that prevent it from growing as a mainstream technology and solution. To enforce this idea of realizing something in the Semantic Web world that simply 'works', as other killer applications in other computer science fields did, parallelization implementations using the map-reduce algorithm [3] are under investigation.

Acknowledgements

This work has been partially founded within the NoTube Project (EU FP7 Integrated Project 231761), during research activities focused on the Semantic Web technologies application in the field of user modeling.

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data Complexity of Query Answering in Description Logics (2006)
2. Perez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL (2006)
3. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters (2004)

Automatically Identifying Bounds on Semantic Annotations for Bioinformatics Web Service Input Parameters

Ravinder Singh, Sean Bechhofer, Khalid Belhajjame, and Suzanne M. Embury

School of Computer Science, University of Manchester
Oxford Road, Manchester, UK
{singhr, seanb, khalidb, sembury}@cs.man.ac.uk

Abstract. Semantic annotation of web services is achieved by associating parts of service interface descriptions to concepts defined in a semantic model, such as an ontology. Annotating web services with semantic metadata is currently a manual procedure that is conducted by human annotators. In practice, however, service annotation can be both time-consuming and error-prone to implement on real-world web services. We have investigated this problem in the bioinformatics domain, where certain characteristics shared by many web services make them a difficult breed to annotate with semantics. By taking these characteristics into consideration, we present a probing-based technique that identifies upper and lower semantic bounds for the annotation of web service input parameters. The resulting bounds can serve as a useful starting point for human annotators and could help reduce the required effort in the semantic annotation process.

Keywords: semantic web services, semantic annotation, bioinformatics.

1 Introduction

Semantic web services [1] are seen as the next step in the life cycle of web service technology. The primary goals driving semantic web service research are to automate the discovery, composition, monitoring and execution of web services. To achieve these goals, web services must first be annotated with semantic metadata. Semantic metadata for web services is provided alongside a semantic model, such as an ontology. Semantic annotations for a web service can then be created by mapping parts of a web service description file (hereafter referred to as WSDL file) to appropriate concepts in an ontology that represents the domain of the web service.

Semantic annotation of web services is a task that is primarily conducted by human annotators. Typically, annotators were not involved in the development of the web services they annotate. As a consequence, they face certain hindrances in the annotation process. First, to annotate a web service, a human annotator must be familiar with the functionality that the web service provides. Such information is obtained by reading any available documentation for the web service

to be annotated. In cases where no documentation exists, annotators can inspect the WSDL file for potential clues to the web service functionality. However, due to the high degree of automation that many web service development platforms provide, there is no guarantee of the quality of a WSDL file. This problem is particularly evident in the bioinformatics domain, where many web service elements are inappropriately named in the WSDL file. For example, an operation parameter may automatically be named “In0” rather than a more meaningful mnemonic such as “protein_sequence”. Secondly, the annotator must also be familiar with the domain ontology to be confident that the best possible annotations are provided. In scenarios where the ontology is large, as is the case with some domain ontologies, it can become highly impractical for a human annotator to know each concept in an ontology. At the time of writing, for example, the *my*Grid bioinformatics domain ontology¹ that was used in our experiment contains a total of 475 concepts. Such an ontology is, however, relatively small when compared to others such as the Gene Ontology² that contains well over 27,000 concepts. Thus, a consequence of these hindrances is that ontology-based annotation becomes a time-consuming and error-prone task when conducted manually.

In our work, we have sought to address these issues in the annotation of bioinformatics web services. Using a probing-based technique, our method seeks to assist human annotators in the annotation of bioinformatics web services (or web services from other domains that possess similar characteristics). More specifically, the presented method identifies upper and lower bounds on the possible annotation (ontology concepts) that a web service input parameter can be annotated with. We argue that for every ontology concept an annotator examines that is not the correct annotation is wasted effort. Thus, we envisage that the output bounds from our process can be a step towards reducing the required effort to annotate bioinformatics web services.

2 Context

Annotating bioinformatics web services with semantic metadata would be a natural step towards achieving the goals set out for semantic web services in the bioinformatics domain. It has, however, been observed that bioinformatics web services have certain characteristics that can further hinder the process of their semantic annotation [2]. Since many services in the bioinformatics domain are developed on an as-needed basis, such web services typically have poor quality descriptions [3]. For example, many web services have been created by converting existing command-line programs using automated conversion software such as SoapLab³. Such software can output poor quality WSDL files in which operations, parameters and datatypes are given meaningless identifiers. Also, as a result of their *ad hoc* creation, such web services can be deployed with little or no

¹ <http://www.mygrid.org.uk/ontology>

² <http://www.geneontology.org>

³ <http://www.ebi.ac.uk/soaplab>

associated documentation [2]. The lack of an agreed upon data model in bioinformatics also means that such web services usually have weakly typed web service operation parameters [4]. For example, many parameters for existing bioinformatics web service operations accept input parameters that are typed using the XML-schema datatype `xsd:string`. In many cases, such operations accept data that typically consists of a complex underlying structure that would be more suitably encoded in a complex XML-schema datatype. Parameters defined using complex data-types can potentially give better clues to the type of data the web service accepts and hence give an annotator a better chance of understanding the functionality of the web service. Collectively, these characteristics result in a manual semantic annotation process which can be both error-prone and time-consuming.

The time-consuming aspect arises because: (a) multiple sources of information may need to be searched for and acquired to understand the functionality of a web service and (b) the annotator must be familiar with the entire ontology, if this is not the case, browsing or navigating a large ontology for the correct concept can become a highly time-consuming task in itself.

Furthermore, the error-prone aspect arises because: (a) Sources of information about a web service's functionality may be scarce, which may lead to an incorrect understanding of the service's functionality and (b) the information in such sources may also be scarce, which may also similarly lead to an incorrect understanding of a services' functionality.

To acquire an understanding of a typical real-world semantic annotation process for web services in the bioinformatics domain, we interviewed an expert annotator on the BioCatalogue project [5]. The first step in the annotation process described by the annotator is concerned with understanding the functionality of the web service operation being annotated. Here, the annotator typically has to use multiple sources of information. One source of information is the annotator's own background knowledge of the domain. If such background knowledge does not exist or does not help to determine the functionality of the web service, the annotator can refer to the documentation of the web service. In cases where no documentation exists, the annotator can also inspect the WSDL file for any potential clues on the functionality of the web service. However, if the web service is described poorly, even this last step becomes a fruitless option. When such sources of information are unavailable or do not help in determining the functionality of the web service, the annotator can attempt to contact the web service provider to acquire information that would help in annotating the web service. Finally, if no contact details are available, the annotator can manually experiment by invoking the web service operation, in the hope that by observing the output of the operation some clues can be derived to the functionality of the web service. Once the functionality of a web service operation has been determined, the next step is to map the various parts of the operation to appropriate concepts in the domain ontology. If the annotator is not completely familiar with the concepts in the ontology, the process of locating the best concept for each annotation can also be a time-consuming task.

To avoid time costs and yet still produce accurate annotations, it is important that human annotators are assisted in the manual annotation process. Various tools and methods have been proposed. In [6], Heß *et al.* introduce the ASSAM web service annotation tool. ASSAM utilises a machine learning method to assist in the annotation for a set of similar web services. Using previously annotated web services as training data, ASSAM is able to suggest candidate annotations for new web services. The machine learning algorithm uses the elements in a WSDL file (operations and their parameters) as features for classification. In [7], Lerman *et al.* also present a machine learning approach to identify semantic labels for input and output parameters of web services. Both these machine learning approaches, however, are dependent on how well the parts of a web service are named in the WSDL file. In [8], Patil *et al.* present a schema matching algorithm to suggest annotations for web services. In this work, the XML-based web service description and the domain ontology are both treated as schemas to be matched to each other. Elements in the WSDL file, such as operation and parameter names, are mapped onto concepts in the ontology using string similarity matching techniques. In [9], Afzal *et al.* use a semi-automated text-mining approach to extract semantic descriptions of bioinformatics services. Using scientific corpora such as journals, they apply various natural language processing techniques to create semantic profiles of web services that can later help in the manual annotation effort. Finally, in [10], Belhajjame *et al.* show that semantic annotations can be inferred from partially annotated workflows. They show that it is possible to infer parameter level annotations based on the data flow connections between operations in the workflow. In scenarios where a non-annotated operation is connected to an annotated operation, constraints can be inferred on the annotations that the non-annotated operation can potentially have.

Unfortunately, approaches based on string-similarity matching and machine learning rely on WSDL files where the elements of a web service are meaningfully named. Thus, the use of such approaches produce ineffective results on web services that have the characteristics described earlier. Some method is therefore required to assist annotators of bioinformatics web services by taking the characteristics of such web services into explicit consideration.

3 Automatically Identifying Annotation Bounds

Our aim is to provide assistance in the annotation of web services that possess the characteristics outlined in section 2. The goal is to see if accurate candidate annotation sets for web service input parameters can be determined by probing operations with annotated input data. The experimented method outputs an upper and a lower bound concept, both of which belong to the ontology being used for the annotation (the domain ontology). Each ontology concept which falls within the upper and lower bound (inclusive) is a potential candidate annotation for the web service input parameter. To arrive at the upper and lower bound concepts, we use a probing technique that consists of three steps.

In the first step, a web service operation is probed with a pool of annotated instance data i.e., raw data values that are mapped to concepts from a

domain ontology (e.g. the value “AB000100” would be mapped to the concept `DDBJ_accession` from the *myGrid* ontology). Thus, the pool will consist of a set of `<instance_value, concept>` pairs. Given a web service operation that is to be annotated, each instance in the pool of annotated instances is used as a value for the input parameter to execute the web service operation. Once executed, an operation typically returns a result. Thus, the next task in the automated process is to use the returned result of the operation to determine if the operation naturally accepted or rejected the input annotated instance. Here, we define acceptance of an instance by a web service operation to mean that the operation was developed to expect such an instance as input (discussed further in section 3.2). In the second step, we use the acceptance results for each instance to label each concept in the ontology. The labels provide an abstraction over the number of instances accepted for a particular concept. In the final step, the labels are used to identify the upper and lower bound concepts that can be used for the annotation of the input parameter. To identify the upper and lower bounds, our process is dependent on the constructs provided by the Web Ontology Language (OWL); thus, the domain ontology that we use is hierarchically modeled in OWL.

In the subsequent sections, we describe each of the stages of the process in more detail. First, in section 3.1 we further discuss the notion of annotated instance data and how such data can be collected.

3.1 Annotated Instance Data

Annotated instances are used to probe a web service operation and thus are a key input to the process. Ideally, for each concept in the domain ontology being used for the service annotations, we would have a set of instances that represent the concept as broadly as possible. For example, for the `protein_sequence` concept in the *myGrid* ontology, we were able to obtain protein sequence instances in different formats such as FASTA and GenPept, as well as the Plain sequence format. As one of the characteristics of the web services in this domain is that the operation parameters are weakly typed (i.e., as plain strings even when the internal structuring of the data passed through the parameters is complex), we did not need to concern ourselves with storing any additional data type information for each instance.

We were able to gather such data from two sources. As proposed in the QuASAR project⁴, one source of such instance data are provenance logs for workflows. Some existing workflow execution engines may keep execution traces (provenance logs). If such logs are kept from workflows that contain annotated web service operations, intermediate data values that pass through the parameters of such operations can be retrieved through these logs. A second method, which typically results in more manual labor, is to browse online databases for such instances. We were able to retrieve instances for concepts by performing manual deep web searches on bioinformatics databases. For example, instances

⁴ <http://img.cs.manchester.ac.uk/quasar>

for concepts such as `DDBJ_accession` and `SWISS-PROT_accession` were easily obtained from online databases such as DDBJ⁵ and UniProtKB⁶, respectively.

3.2 Probing and Instance Classification

The first stage of the process is primarily concerned with categorising each annotated instance as either being accepted or rejected by the web service operation. For the purpose of this initial investigation, we have only focused on identifying annotations for operations with single input parameters and do not yet consider the problem of attempting to identify candidate annotations for operations with multiple input parameters.

An important part of this stage is determining whether a particular annotated instance was accepted or rejected by the web service operation. To distinguish accepted instances from rejected instances, we use the returned result of the web service operation. Determining whether an input to a web service operation is accepted or rejected, based solely on observing the operations output, is a difficult task to automate. It is difficult primarily because there is no widely adopted convention on how a web service operation handles normal termination compared with abnormal termination. To signify abnormal termination, some operation's return the `null` data value or an empty string value, whereas others may return a custom string error message such as "Error: Invalid input" or even throw an exception. Furthermore, we also came across operations with weak input validation. Such operations typically attempted to perform their computations even with invalid input values, and returned string message results like "No results found". In such cases, it is extremely difficult, even manually, to determine whether the input instance was valid but the operation produced no result or whether the input instance was invalid and hence there was no result. As our aim was to automate this process, we formulated a heuristics-based criteria for classifying each annotated instance's effect on a web service. Based on the effect of a web service operation, input annotated instances are assigned to one of two classes, namely, `accepted` or `rejected`. The criteria are outlined below:

Rejected: An annotated instance is considered rejected if the operation output is a null data value, an empty string value or a thrown exception.

Accepted: Conversely, an annotated instance is considered accepted if the operation output does not match the criteria for `Rejected`.

Using the above criteria, each annotated instance can then be associated with a class that indicates whether the operation accepted or rejected it. Hence, the output from this stage of the process is a collection of `<<instance, concept>, class>` pairs.

⁵ <http://www.ddbj.nig.ac.jp>

⁶ <http://www.uniprot.org/help/uniprotkb>

3.3 Concept Labeling

In the next stage of the process, the level of abstraction is raised from the instance level to the concept level. To accomplish this, each concept in the domain ontology is labeled depending on the number of accepted instances for the concept. The labels provide guidance on arriving at the upper and lower bound concepts (next step in the process). Below, we further discuss each label and the information that it provides.

BEST CASE: At any time, only one concept in the domain ontology would be labeled as **BEST CASE**. In scenarios where instances are only accepted for a single concept in the ontology, we have reason to speculate that the best case concept is a good candidate for annotating the input parameter. In such scenarios, it is possible that the web service performs strong input validation and thus only accepts instances belonging to the identified concept.

GENERALISE: A concept labeled as **GENERALISE** indicates that the parent concept (of the labeled concept) is a more promising candidate annotation. For a concept C to be labeled as **GENERALISE**, all of its instances (including the inferred instances from its descendants) must be accepted by the operation and there exists another concept in the ontology that is not a descendant of C , for which at least one instance is accepted.

SPECIALISE: A concept is labeled as **SPECIALISE** whenever a child concept (of the labeled concept) is a possible candidate annotation. For a concept C to be labeled as **SPECIALISE** a non-empty subset of the instances associated to it must be accepted.

3.4 Identifying Upper and Lower Bounds

In the final stage of the process, the labeled concepts from the previous stage are used to identify the upper and lower bound concepts in the ontology. We define an upper bound concept as the most general concept that can be considered for the annotation of the web service input parameter. Conversely, the lower bound concept is defined as the most specific concept that can be considered an annotation of the input parameter. Let LC be the set of labeled concepts from the ontology, $ubound$ and $lbound$ to be the upper and lower bound concepts, then the process of locating the upper and lower bound concepts can be expressed using the algorithm in listing 1.

As shown in the pseudo code, the upper and lower bounds are easily identified if a best case concept resulted from the probing (i.e., all the instances for a single concept and only those instances were accepted). In such a scenario, both the upper and lower bound will be the same concept (the best case concept).

If the best case scenario is not realised, then the process of locating the upper and lower bound is dependent on the concepts labeled as generalise and specialise, respectively. The upper bound concept is the first common ancestor

of the generalise concepts (*GC*). In practice, this is accomplished by constructing a new anonymous OWL concept that represents the union of each generalise concept. Once constructed, the super concept of the constructed concept is the upper bound concept (first common ancestor). Conversely, the lower bound concept is the first common descendant of the specialise concepts (*SC*). In practice, to find the lower bound concept, we construct a new OWL concept that represents the intersection of each specialise concept. Once constructed, the sub concept of the constructed concept gives the lower bound concept (first common descendant).

```

inputs: LC - outputs: ubound, lbound
1  foreach concept ∈ LC
2    if (labelOf(concept) = "BEST CASE") then
3      ubound := concept
4      lbound := concept
5    exit-foreach
6  else if (labelOf(concept) = "GENERALISE") then
7    GC := GC ∪ concept
8  else if (labelOf(concept) = "SPECIALISE") then
9    SC := SC ∪ concept
10 end-foreach
11 if (ubound = null and lbound = null) then
12   if (GC ≠ ∅)
13     ubound := getLowestCommonAncestor(GC)
14   if (SC ≠ ∅)
15     lbound := getGreatestCommonDescendent(SC)
16 end-if

```

Listing 1. Pseudo code for identifying the upper and lower bound

The candidate concepts for the annotation of the input parameter are those concepts which fall within the upper and lower bound concepts (inclusive) in the ontology. In the possible scenario where we do not have an upper bound (i.e., no concepts meet the generalise label criteria) but have a lower bound, then the upper bound is all the ancestor concepts of the lower bound. If we do not have a lower bound (i.e., no concepts meet the specialise label criteria) but have an upper bound, then the lower bound is all descendant concepts of the upper bound. Finally, in the case where we do not have an upper and a lower bound, then the upper bound is the most general defined concept in the ontology (i.e., the direct sub concept of `owl#Thing`) and the lower bound is all the descendant concepts of the upper bound – i.e., the entire ontology becomes the candidate set.

4 Results and Qualitative Analysis

To conduct our experiment we used a set of 19 semantically annotated bioinformatics web service operations. Each operation was annotated manually by an

expert annotator from the bioinformatics domain. The existing annotations served as a gold standard for us to measure the accuracy of the output bounds from the automated process. The input parameters for each operation are annotated using concepts from the *my*Grid bioinformatics domain ontology (consisting of 475 concepts). In total, we were able to collect 183 raw instances for a total of 84 concepts in the ontology.

4.1 Experiment and Results

In our experiment, we tested the effectiveness of the process by running each web service operation through the process using the aforementioned instance data. We hypothesise that when it is possible to automatically distinguish between normal and abnormal termination of operations, then accurate candidate annotation sets which are significantly smaller than the domain ontology can be identified. To test this hypothesis, we measure how accurate the resulting candidate concept sets are (i.e., do the sets contain the gold standard annotation?) and how much of the ontology we are able to rule out for the annotator (i.e., is the size of the candidate concept set significantly less than the size of the ontology?). A summary of the results is presented in Figure 1.

For each operation, the graph shows the number of candidate concepts identified for each operations input parameter, i.e., the concepts that appeared within the upper and lower bounds (inclusive). The candidate sets all contained the gold standard annotation concept, thus giving a recall of 100%. For 10 of the operations we were able to obtain candidate sets that contained significantly fewer concepts when compared to the size of the full ontology. In two of these cases, the best case scenario was realised, since the operations only accepted instances for a single concept. Upon analysis of the probing results for these two operations, we observed that these operations performed strong validation of input values

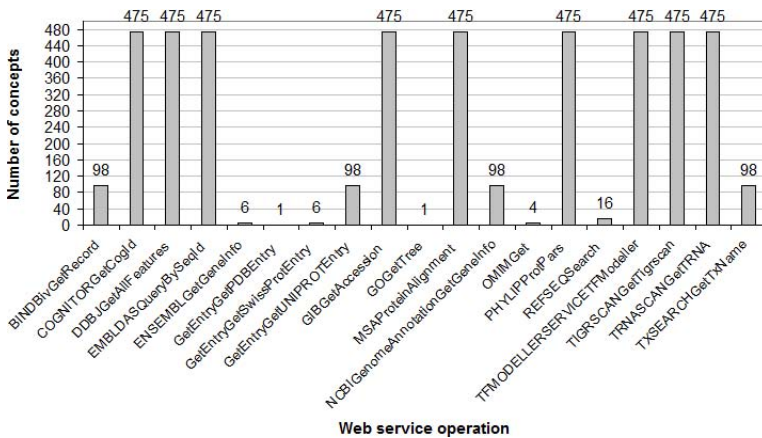


Fig. 1. Results from the experiment

and abided by the acceptance criteria that we used. For example, both operations returned an empty string response when terminating abnormally.

In the four cases that produced a candidate set size of 98, we found that although these operations performed strong input validation (i.e., accepted a low number of instances) the format of the instances of their gold standard annotation overlapped with the format from the instances of other “similar” concepts. For example, the gold standard annotation for the `BivGetRecord` is the concept `BIND_id`. In total, for this concept, we had two raw instances “240947” and “185171”. Although both of these instances were accepted, the `BivGetRecord` operation was unable to discriminate between similarly formatted instances for other “id” concepts. For example, the operation also accepted instances from other concepts such as `chEBI_term_id`, `KEGG_record_id` and `MaizeGDB_id`. The instances for these concepts have the same format as the instances for `BIND_id` (positive integers), thus the `BivGetRecord` operation incorrectly—but understandably—recognised such instances as valid `BIND_id`’s. As a result, each of the four operations that had a candidate concept set size of 98 obtained the same upper bound annotation of `bioinformatics_record_id` (first common ancestor of the id concepts) and did not produce a lower bound.

The results also show that for 9 of the operations the process was ineffective as the identified upper and lower bounds resulted in the entire ontology being returned (475 concepts). The reason for these large candidate concept set sizes was that these operations did not abide by the acceptance criteria that was used for classifying the instances as accepted or rejected. For some of these operations, there is evidence that they attempted to validate input values. However, their method of returning an abnormal response was typically using a proprietary string error message which we could not have anticipated when defining our generic acceptance criteria. In other cases, due to a lack of validation of the input parameters, the operations attempted to continue their computation regardless of whether the input value was valid or not. In these particular cases, the operations did not strictly terminate abnormally and hence returned a proprietary response that did not match the defined criteria for rejecting an instance.

In summary, the results show that the success of such a probing technique is highly dependent on being able to automatically distinguish between normal and abnormal termination of web service operations. When strict conventions on how to terminate an operation are adhered to, such a probing-based technique has the potential for becoming a cost effective solution for acquiring parameter-level semantic annotations.

4.2 Instance Pool Design

In addition to the experiment, an investigation was conducted to determine if characteristics of a “good pool” of annotated instances can be identified. The aim of this investigation was to attempt to identify guidelines for collecting annotated instances as they are an integral input to the process. To do this, we generated 200 instance pools using the original collection of 183 annotated instances. Using a total of 8 different pool sizes (20, 40, 60, ..., 160) which gave us a spread of

pool sizes, 25 random instance pools of each size were created. We reduced the number of web service operations used to 10, by disregarding the 9 operations which achieved a candidate concept set size equal to the size of the ontology. The reason behind this decision was that these particular operations did not abide to the acceptance criteria, and hence would only cause noise in the output of this investigation. Each generated instance pool was then used as input into the process for each web service operation. Using the output candidate concept sets, we selected the instance pools that resulted in good candidate concept sets and the pools that resulted in bad candidate concept sets. To determine the instance pools that performed well (the “good” pools), the following criteria was applied: (1) all candidate concept sets from a good instance pool must contain the gold standard annotation for each operation and (2) all candidate concept set sizes are less than half the size of the ontology. The bad instance pools were selected using the criteria: (1) more than half of the candidate concept set sizes are equal to the size of the ontology and (2) there exist some candidate concept sets that do not contain the gold standard annotation for an operation.

Using the above criteria, 18 good instance pools and 32 bad instance pools were found. In our analysis of these two sets of pools, we found that on average the instances in the good pools represented more concepts in the ontology (77) than the bad pools did (26). By representation of concepts, we mean the number of concepts for which instances existed in the pool. We also found that the good pools contained more instances than the bad pools did; for good pools the average number of instances per represented concept was approximately two, whereas for the bad pools the average number of instances per concept was approximately one. Perhaps unsurprisingly, these results provide some evidence to suggest that the more concepts represented and the more instances per concept within instance pools, are two potentially useful guidelines to follow when collecting such instance data.

5 Conclusions

In this paper, we have presented a technique to obtain candidate annotations for bioinformatics web service operations. We envisage that the technique could help to reduce the possibility of inaccurate annotations during the manual semantic annotation of such web services. The conducted experiment showed that for a sample of 19 bioinformatics web services, the process achieved a 100% recall with the gold standard annotation always being a member of the candidate annotation set. For the 10 web services that abided to the acceptance criteria, we were able to rule out more than half of the ontology through the output bounds. In some cases, even achieving candidate concept sets of less than 20 in size.

In cases where our acceptance criteria for distinguishing between the two did well, we were able to achieve candidate concept sets that were on average an approximate 90% reduction of the ontology. However, in cases where the acceptance criteria failed, the resulting output bounds were equal to the size of the ontology, and thus would have been no help to a human annotator. Thus,

for such a technique to be useful in practice, an effective means of being able to automatically distinguish between abnormal and normal termination of web service operations must be realised. One solution to this problem is to actively encourage the web service development community to follow strict guidelines when designing and developing web services. Furthermore, the success of this technique is also dependent on how good a set of annotated instances can be obtained cost effectively. Our investigation provided some evidence to suggest characteristics of good collections of annotated instances, we envisage that these observed characteristics can be used as guidelines when collecting such data.

References

1. McIlraith, S., Cao, T.S., Zeng, H.: Semantic web services. *IEEE Intelligent Systems* 16, 46–53 (2001)
2. Lord, P., Alper, P., Wroe, C., Goble, C.: Feta: A light-weight architecture for user oriented semantic service discovery. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005*. LNCS, vol. 3532, pp. 17–31. Springer, Heidelberg (2005)
3. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.* 34(Web Server issue) (July 2006)
4. Hull, D., Stevens, R., Lord, P.: Describing web services for user-oriented retrieval (2005)
5. Belhajjame, K., Goble, C., Tanoh, F., Bhagat, J., Wolstencroft, K., Stevens, R., Nzuobontane, E., McWilliam, H., Laurent, T., Lopez, R.: Biocatalogue: A curated web service registry for the life science community. In: Microsoft eScience conference (2008)
6. Heß, A., Johnston, E., Kushmerick, N.: Assam: A tool for semi-automatically annotating semantic web services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) *ISWC 2004*. LNCS, vol. 3298, pp. 320–334. Springer, Heidelberg (2004)
7. Lerman, K., Plangprasopchok, A., Knoblock, C.: Automatically labeling the inputs and outputs of web services. In: *AAAI 2006: proceedings of the 21st national conference on Artificial intelligence*, pp. 1363–1368. AAAI Press, Menlo Park (2006)
8. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework. In: *WWW 2004: Proceedings of the 13th international conference on World Wide Web*, pp. 553–562. ACM, New York (2004)
9. Afzal, H., Stevens, R., Nenadic, G.: Mining semantic descriptions of bioinformatics web resources from the literature. 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. 535–549. Springer, Heidelberg (2009)
10. Belhajjame, K., Embury, S.M., Paton, N.W., Stevens, R., Goble, C.: Automatic annotation of web services based on workflow definitions. *ACM Trans. Web* 2(2), 1–34 (2008)

REST Inspired Code Partitioning with a JavaScript Middleware

Janne Kuuskeri and Tommi Mikkonen

Department of Software Systems, Tampere University of Technology
P.O. Box 553, FI-33101 Tampere, Finland
{[janne.kuuskeri](mailto:janne.kuuskeri@tut.fi), [tommi.mikkonen](mailto:tommi.mikkonen@tut.fi)}@tut.fi

Abstract. The Web has become a commonly used programming environment for a number of types of applications, and to a great extent, applications that would have formerly been implemented as desktop systems are now run inside the browser. Numerous approaches have been proposed for application development on the Web, but it is not obvious how the different approaches fit together. In this paper, we introduce a middleware platform intended for composing JavaScript applications. When using the middleware applications are implemented solely using JavaScript in a fashion, where the client side of applications runs inside the browser, whereas the server side gains advantage from a newly emerged activity, CommonJS.

1 Introduction

The software industry is currently experiencing a paradigm shift towards web-based software. The Web has rapidly become a de-facto programming environment for a number of types of applications, and to a great extent, applications that would have formerly been implemented as desktop systems are now run inside the browser.

Unfortunately, the browser was intended to be a hypertext distribution environment, not a runtime system for serious software systems. Consequently, there are numerous restrictions on how the Web should be used as an application platform. One of the fundamental limitations is that if one wishes to build applications relying on client-side functionality, the only programming language that can be used is JavaScript, since it is included in all the major browsers.

At the same time, the Web also lends itself to numerous approaches that can be adapted when developing applications. Perhaps the most prominent and widely accepted approaches are Web Services [2] and Representational State Transfer (REST) [3].

In this paper, we introduce a middleware intended for composing JavaScript applications using a paradigm inspired by REST for client-server communication. Applications are implemented solely using JavaScript, The client side of applications runs inside the browser, whereas the server side gains advantage from a newly emerged activity, CommonJS[1].

¹ <http://commonjs.org/>

2 Background

Our middleware is built by combining Representational State Transfer (REST) and JavaScript programming language in a new and innovative fashion. Instead of using JavaScript only inside the browser, we have extended its role to server-side programming as well. Moreover, REST based interfaces will be used for enabling the interaction of JavaScript code running in the client and the server.

2.1 REST

Representational State Transfer (REST) as a term, originally introduced by Fielding [1], refers to an architecture suited for distributed hypermedia systems, such as the Web. The architecture consists of clients, which initiate requests, and servers, which process requests and reply with suitable responses. A RESTful architecture is commonly thought of as being resource-oriented, where a resource is basically anything that can be considered as a meaningful concept and is addressable by a URI. Requests and responses are then built around the transfer of representations of resources, which in the case of Web are most commonly documents that capture the state of the resource. A RESTful Web API is a web service that is implemented using HTTP. It consists of three elements:

- *Unique Resource Identifier (URI)*. Since resources form a key concept in REST, each of them must be addressable.
- *Operations*. To manipulate resources, clients use a set of operations supported by the service. When using HTTP, these operations are associated with HTTP methods, such as GET, PUT, POST, and DELETE.
- *Data representation*. The representation of the data is needed in order to interpret it correctly. This is commonly implemented using Multipurpose Internet Mail Extension (MIME) types.

The simplicity of REST – together with a number of other qualities, such as scalability and generality of interfaces – has made it a commonly advocated approach for developing resource oriented web sites.

2.2 CommonJS

CommonJS is an effort to standardize the runtime environment for JavaScript running outside the browser. Conventionally JavaScript is run inside the browser and thus enjoys the presence of DOM, the document object model. The DOM exposes all relevant parts of the browser and the web page in order to make JavaScript a powerful tool for programmers when creating rich and dynamic client side web applications.

However, JavaScript running outside the browser does not have the DOM in its side and thus lacks the tools to implement any meaningful applications. Most importantly JavaScript lacks the module system so there is no standard way of importing other modules. Furthermore, there are no means to publish an API

for external modules to consume while preserving part of the functionality internal. Finally, standard APIs for accessing the file system, networking, databases, web server and all other resources commonly available in other programming languages are also missing from JavaScript.

The CommonJS initiative is set out to fix the above problems by defining the module system and common APIs and interfaces to external systems. This is done without any modifications to the language and its syntax. Everything is implemented in terms of libraries that applications may import when needed. Moreover, as few globals as possible are exposed into the runtime environment. The most notable globals are the ones needed by the module system (`exports` and `require`). Being true to the nature of JavaScript the module system itself can be implemented in JavaScript using object-oriented design and functional programming with closures. This provides for seamless and natural introduction of interoperable modules to the language. Listing 1 shows an example of exporting and requiring functions.

```
// -- in file called myfuncs.js --
exports.myPublicFunction = function () { ... }

// -- in another file --
var mymodule = require("myfuncs");
mymodule.myPublicFunction();

// or simply:
require("myfuncs").myPublicFunction();
```

Listing 1. Exporting and Requiring

Utility libraries outside the scope of CommonJS, such as different test frameworks, parsers and database adapters, can be constructed as packages and distributed using the package management tool that is also specified by CommonJS. The package manager takes care of the dependencies and automatically installs all required packages, similarly to Python's EasyInstall and Ruby's RubyGems.

3 Architecture

The nature of our implementation (which is described in more detail in the next section) encourages us to work closely – if not directly – with the HTTP protocol. We will need to intercept HTTP requests before they are delivered to applications, and we wish to keep the design simple when implementing the interface. Moreover, the platform should support easy configuration to be able adapt into varying requirements. Driven by these premises, we have chosen to interface directly with the web server and to implement a stack of independent and interchangeable middleware components between the web server and the web application.

3.1 JSJI

JSJI (JavaScript Gateway Interface) is a specification driven by the CommonJS group to standardize the interface between the web server and the web application. The specification defines the format for creating JSJI compatible applications and their request and response objects. The request object can be thought of as the *environment* of the request as it contains all the parameters that CGI specification passes using environment variables, such as the requested URL, request method, and HTTP headers.

A JSJI application is simply a JavaScript function which takes exactly one parameter, the request, and returns an object with three properties: `status`, `headers`, and `body`. Listing 2 shows a minimalistic JSJI application.

```
var helloapp = function (req) {
  return {
    status : 200,
    headers : {"Content-Type":"text/html"},
    body : ["<html><body>Hello world!</body></html>"]
  };
}
```

Listing 2. Simple JSJI Application

3.2 Middleware

Generally the term middleware refers to network oriented integration software. For many web applications and more specifically in the context of this paper, the term “middleware” refers to a software component that sits between two other components and performs a task specific to that middleware component either before the HTTP request is given to the web application or after the application returns the HTTP response or both. Granted, this kind of usage could also be defined as layered architecture but we have chosen to use the term middleware as it has become somewhat pervasive in the realm of web server interfaces such as JSJI, WSGI and Rack. Usually a middleware component is generic component that may be applied to any web application but they can also be bespoke components used to configure or amend the web application it supports. A good example of a middleware component is the one that implements HTTP Basic authentication: it is generic to all web applications and it does not affect the behavior of the application itself.

A very useful feature of middleware components is that they may be chained one after the other to compose the desired behaviour for a given web application. This is a powerful paradigm for code reuse and configurability of applications. Fig. 1 shows an example of middleware chaining. In the example Content Length and HEAD middleware components operate on HTTP responses while Basic Authentication takes place before the HTTP request may be delivered to the application. After the authentication middleware component has verified the

client’s credentials the request is delivered to the application to carry out the business logic. When the application has completed the request, the generated response is first checked by the HEAD middleware component to make sure that if the requested method was an HTTP HEAD the body of the response must be empty. After this, the content length middleware component calculates the size of the response body and adds the Content-Length attribute to the HTTP headers.

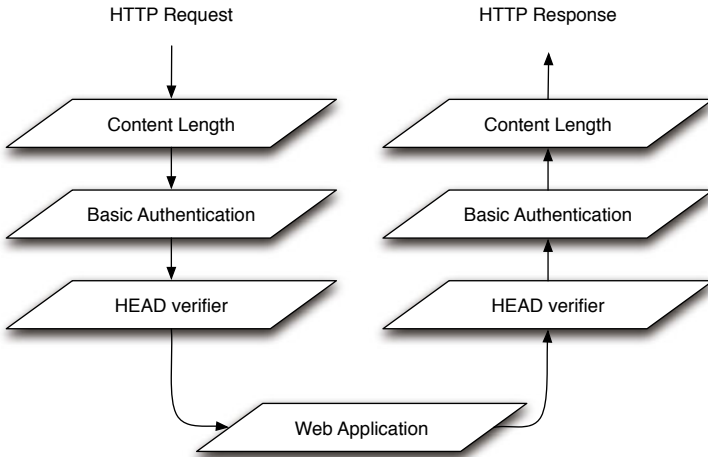


Fig. 1. Middleware Chaining

The JSGI interface has been designed to support middleware chaining. Conventions used for writing JSGI middleware follow the structure of writing JSGI applications themselves. The middleware usually stores the application – which can actually be another middleware component – in its closure in order to call it when the HTTP request comes in. At runtime, the middleware must look like an application to the calling party but at the same time act like the JSGI “framework” to the application it invokes. The common procedure when writing JSGI middleware is to

1. Have a function which takes in the request object.
2. Use the request object to optionally perform some pre-processing.
3. Invoke the original application (or the next middleware component).
4. When the lower level is finished, optionally perform post-processing on the response object.
5. Return the three element array specified by the JSGI to the upper layer.

Listing 3 shows a simple piece of middleware that logs all authentication headers into the application’s log file. In the end of the example, we demonstrate how to construct an application with two middleware components.

```

var AuthLogger = function (app, authkey) {
  return function (req) {
    var i, res, auths, authkey = authkey || "authorization";

    // print authorization headers if present
    if (authkey in req.headers) {
      auths = req.headers[authkey].split(",");
      for (i = 0; i < auths.length; i += 1) {
        log.info(trim(auths[i]));
      }
    }
    // invoke the app and return whatever the app returns
    return app(req);
  }
};
exports.app = ContentLength(AuthLogger(helloapp));

```

Listing 3. Logger Middleware

4 Implementation

Our system – a platform codenamed “Groke” – is mainly targeted for, but not restricted to, rich and dynamic single page web applications. Hosted applications may be client side only, server side only, or split so that there are components running on both sides (Fig. 2). The platform helps in hiding the networking details when communicating between the client and the server; even if the application runs solely on the client, it is able to easily consume common services provided by the server side of the platform. The platform is implemented in JavaScript only and consists of mainly the server side components but also has an optional client side library for easy utilization for the developers.

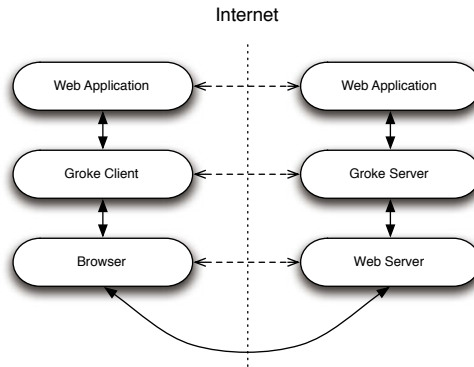


Fig. 2. System Overview

4.1 Partitioning

Code partitioning is implemented using the same `exports/require` mechanism as the module system specified by CommonJS. The platform makes no restrictions as to what may be exported or required remotely. So, a client side application may require and consume any public server side module or function it chooses. The philosophy is that everything that is not explicitly loaded into the client using `<script>` tags or some other means provided by the client environment are left on the server. The server side modules remain accessible to the client (can be called remotely) but they are not transferred over the network. This gives the developers full control over which components should be run on the server and which on the client. At the same time the platform does its best in hiding the networking details and thus minimizing the boilerplate code.

4.2 REST Inspired Interface

By default, the server side of the platform exposes all of its modules and their exposed variables (may they be functions, objects or properties) through a resource oriented interface. That is, the module hierarchy is simply turned into a hierarchy of resources. For example, the `list` function of the `file` module would respond to resource name `/file/list`. In order for a client to actually invoke this function, it needs to send an HTTP POST request to that URL with all the necessary parameters encoded as a JSON string in the body of the request. At the time of writing, the platform also has limited support for exposing objects and anonymous functions that may be created dynamically as part of return values from various operations. Table 1 summarizes the different types of resources exposed by the platform.

Table 1. Resource Identifiers

Resource type	URL scheme	Example
Functions provided by modules.	<code>/groke/module/ <module>/<function></code>	<code>/groke/module/ file/list</code>
“Constructor” functions, i.e. functions invoked with the <code>new</code> operator.	<code>/groke/ctor/ <module>/<function></code>	<code>/groke/ctor/ file/File</code>
Objects that are exposed as a return value of a function invocation.	<code>/groke/obj/ <object-id>/ <property></code>	<code>/groke/obj/ 8327/write</code>
Functions exposed as a return values. This is analogous with the previous case.	<code>/groke/func/ <function-id></code>	<code>/groke/func/9284</code>

The interface supports HTTP POST method only. Surely, this seems like a violation to the rules of REST. However, the platform cannot have any understanding about the modules and functions it exposes; it only knows that it serves a set of operations as resources that clients may invoke. Furthermore, these operations may require input parameters which, by REST convention, are POSTed to these operations. Modules and functions as resources also have a static nature in that they cannot be explicitly created, replaced or deleted using the methods of HTTP. Although, as suggested by the Table [1](#), new objects and functions can be created and exposed dynamically but it is always more or less a side effect of a function invocation issued with HTTP's POST method.

Another controversy is presenting functions as resources. Albeit, the REST guidelines define that a resource can be anything, we do realize that presenting functions as resources makes the interface seem more like an RPC type of service. In the REST world it is highly discouraged to use verbs as the names of resources and this is what inevitably happens when exposing functions as resources. On the other hand this is not the platform's choice, the names of the resources are unknown to the platform when the system starts, everything is exposed dynamically and therefore also the names of the resources are chosen by the programmers providing the operations.

Because of these conflicting aspects we feel that it is not justified to call the interface truly RESTful but something that is inspired by REST.

For a client to consume server's functions by manually sending HTTP queries to various URLs would be quite cumbersome. This is why the platform adds a level abstraction on top of the RESTful communication layer by automatically generating the required code for sending and receiving requests. This is achieved by intercepting the request [2](#) that originally loads the client side application into the browser. When the request is intercepted by the server side platform, it reads the application from the disk and inspects the code in it. During this inspection the application loader finds all dependent modules and replaces them with generated stubs that are actually proxies to server side implementations. After that the instrumented application is sent to the browser.

The client application is now able to invoke the required functions as if they were local. In the background, however, the function parameters are encoded into JSON and POSTed to the RESTful interface of the server. On the server, the request is unmarshalled and requested operation is carried out. The result is again encoded in JSON along with some platform specific meta data about the result (such as possible exceptions) and sent back to the client.

4.3 CommonJS Compliance

Our implementation conforms to the CommonJS specifications and is built on top of the JSGI specification using middleware components to carry out the processing or requests. As for the runtime environment we have been using libraries

² This is an HTTP GET as opposed to POST mentioned earlier, which is used for consuming the RESTful interface after the application has been loaded.

called Narwhal³ and Jack⁴ running on the Rhino JavaScript engine. Narwhal is most likely the most complete implementation of CommonJS specifications currently. It is itself implemented in JavaScript and has support for different JavaScript engines. At the time of writing Narwhal has the best engine support for Rhino.

Jack can be thought of as a reference implementation of the JSGI specification. It is implemented as a CommonJS package and thus can be easily installed into the narwhal environment using the package manager. Jack has support for various web server technologies such as Java Servlets, CGI and the Simple framework⁵. So far, we have been using the Simple framework because of its speed and simplicity.

To enforce code reusability and to follow the JSGI mindset we have implemented several middleware components for the platform. For example, one that automatically invokes appropriate function on the underlying application based on the received HTTP method. This is very useful when implementing RESTful interfaces. Another convenient piece of middleware automatically strips the module and function information from the request URI and function parameters from the request body and hands them to the application as function parameters. The former of these middleware components is presented in Listing 4. The example demonstrates how easy it is to create useful JSGI middleware components that web applications may choose to use when appropriate.

4.4 Example Application

Our minimal example application shows how to write an application that makes use of the platform. We have written our own CommonJS compliant implementation of the `XMLHttpRequest` that is able to run on the server. The implementation is written in file called `xmlhttprequest.js` and small portion of the code is shown in Listing 5. We have left out all the implementation details and most of the functions and properties but for the curious, the `XMLHttpRequest` is implemented using Narwhal's `http` and `event-queue` modules.

Applications running in the client are now able to use the server side `XMLHttpRequest` as a proxy to access data from other sites. An example client is written in Listing 6. There, the client simply requires the `xmlhttprequest` module and starts using it. Even though the application does nothing useful, it demonstrates how easily server side modules become available for the client. Note that the example uses synchronous requests, but the platform, and our own `XMLHttpRequest` both have support for asynchronous requests too.

4.5 Limitations

In the following we describe some of the limitations that the platform still has. To a large extent, they are not much limitations of the approach per se but more limitations of the current implementation.

³ <http://narwhaljs.org/>

⁴ <http://jackjs.org/>

⁵ <http://www.simpleframework.org/>

```
// this middleware takes the request method and calls
// corresponding function on the app
var MethodGrab = function (app) {
  return function (env) {
    var method = env.method.toLowerCase();
    if (typeof app[method] === "function") {
      // call the app
      return app[method](env);
    } else {
      // "method not allowed"
      return {
        status: 405, headers: {}, body: []
      };
    }
  };
};

var message = "";

//the application can now have functions like 'get' and 'put'
var myapp = {
  get: function (req) {
    return {
      status: 200,
      headers: {"Content-Type": "application/json"},
      body: [JSON.stringify({"message": this.message})]
    };
  },
  put: function (req) {
    message = JSON.parse(req.body().decodeToString()).message;
    return { status: 204, headers: {}, body: [] };
  }
};

// expose the application with our middleware
exports.app = Jack.ContentLength(MethodGrab(myapp));
```

Listing 4. RESTful Middleware Component

Parameters and Return Values. Currently only basic types and strings are supported in remote function parameters and their return values. This is partly by design because we want to leave all functions to be executed in their intended environment. However, we could allow objects that have no functions (i.e. associative arrays) to be serialized and sent over the network.

Server Push. Ideally, from the communications perspective, the platform should be symmetrical. That is, the server should be able to remotely call client's functions the same way that the client is able to call server's functions. So far, we have been


```

exports.XMLHttpRequest = function () {
  ...
  this.onreadystatechange = null;
  this.open = function (method, url, async, user, pwd) {...};
  this.send = function (data) {...};
  ...
};

```

Listing 5. Example application running on the server

```

// require xhr (the implementation stays on the server)
var xhrlib = require("xmlhttprequest");

var xhrtest = function () {
  // creates new xhr object but leave it on the server
  var xhr = new xhrlib.XMLHttpRequest();
  xhr.open("GET", "http://www.somesite.com/rest/", false);
  xhr.send();
  alert(xhr.responseText);
};

```

Listing 6. Example application running on the client

testing different Comet approaches and the new WebSocket standard but these tests have been implemented as applications utilizing the Groke platform and not as part of the platform itself.

Garbage Collection of Remote Objects. As already mentioned, the platform currently has limited support for objects and functions as part of the RESTful interface. This is mostly due to lack of proper garbage collection of these objects. In the current version, we have degraded into using only timeout based garbage collection of objects. We leave the design and implementation of a proper garbage collection as future work.

5 Conclusions

Many approaches have been proposed for application development on the Web, but it is not obvious how the different approaches fit together. In this paper, we have introduced a middleware platform for composing JavaScript applications using a REST inspired interface. Unlike in many other approaches, we use JavaScript also on the server side to provide a uniform development model.

At present, the implementation is at a level of research software. By the time of the workshop, we hope to be able to release the system in open source under some commonly used license for wider audience. In the process, we have made code contributions to Narwhal and Jack projects but we have not really participated in the CommonJS specification itself.

References

1. Fielding, R.: Architectural Styles and Design of Network-based Software Architectures. Doctoral Dissertation, University of California, Irvine, USA (2000)
2. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice-Hall, Englewood Cliffs (2007)
3. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly, Sebastopol (2007)

The SOA Paradigm and e-Service Architecture Reconsidered from the e-Business Perspective

Stanisław Ambroszkiewicz, Waldemar Bartyna, Marek Faderewski,
Dariusz Mikułowski, Marek Pilski, Marcin Stepniak, and Grzegorz Terlikowski

Institute of Computer Science, Polish Academy of Sciences
Al. Ordona 21, PL-01-237 Warsaw

and Institute of Computer Science, University of Podlasie, PL-08-110 Siedlce, Poland

sambrosz@ipipan.waw.pl

<http://www.ipipan.waw.pl>

Abstract. A business service has well founded structure where its operations (corresponding to request-quote, order-contract, invoice-payment) are related to each other. These relations cannot be expressed in WSDL. The request-quote operation corresponds to SLA negotiations and can be performed in a universal description language such as OWL that can also express all the relations between service operations mentioned above. Generally, from the e-business perspective the following notions are important: (1) *Service architecture*. (2) *Communication protocols in e-business processes*. These notions are crucial for providing standards necessary for creating open, heterogeneous and scalable systems for realizing complex e-business processes. These notions are discussed in the paper.

Keywords: business process, service arrangement, service composition, ontology, task execution.

1 Introduction

The classic version of the SOA (Service Oriented Architecture) paradigm, see [1], may be summarized as follows. *SOA provides a standard programming model that allows self-contained, modular software components residing on any network to be published, discovered, and invoked by each other as services. There are essentially three components of SOA: Service Provider, Service Requester (or Client), and Service Registry. The provider hosts the service and controls access to it, and is responsible for publishing a description of its service to a service registry. The requester (client) is a software component in search of a component to invoke in order to realize a request. The service registry is a central repository that facilitates service discovery by the requesters.*

The key point of the SOA paradigm is service integration, that is, "... other applications (and other services) can discover and invoke the deployed service ..." In order to realize this vision, simple ubiquitous and automatic process modeling techniques are needed. The service integration is interpreted in several ways,

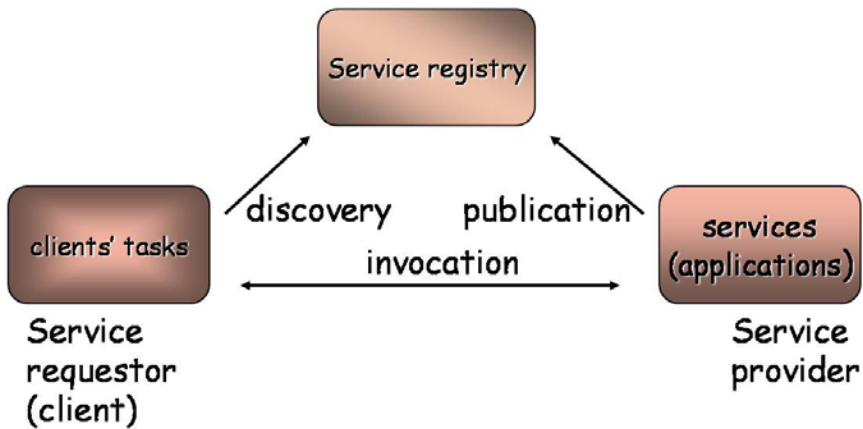


Fig. 1. The classical version of the SOA paradigm

e.g., as service composition, service orchestration, service choreography, process modeling, etc. These terms have been widely used to describe business interaction protocols. Generally, business processes comprise collaborative services that can provide much more complex functionalities, than the single services [2].

One of the classical definitions of a business process is following [3]:

It is a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focuss emphasis on what. A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs: a structure for action. ... Taking a process approach implies adopting the customers point of view. Processes are the structure by which an organization does what is necessary to produce value for its customers.

Another definition of electronic business process is given in IST CONTRACT Project [4]: *A business process specifies the potential execution order of operations from a collection of Web Services, the data shared between these Web Services, which partners are involved and how they are involved in the business process, joint exception handling for collections of Web Services, and other issues involving how multiple services and organizations participate. In other words it defines the composition of Web Services in order to provide higher functionalities.*

The terms *higher functionalities* and *customers point of view* from the above definitions may be interpreted as a client (customer) task the business process is supposed to accomplish.

Although languages and technologies such as WSDL (Web Service Description Language), BPEL4WS (Business Process Execution Language four Web Services), and OWL-S (Web Ontology Language for Services) seem to be adequate for modeling and execution of the electronic business processes, automated process composition is still a challenge in Information Technologies.

The notion of *e-business process* (as the main point of the discussion) is strongly related to the several crucial issues, such as the SOA paradigm, and the concept of e-business service in SOA. The classic version of the SOA paradigm has its origins in software engineering. Service is viewed as server application (in the client-server model) waiting to be invoked by clients. From the point of view of e-business process, a service may be also active and looking by itself for clients tasks that can be accomplished. This corresponds to the reverse auctions in business practice.

A business service has a well founded structure where its operations (corresponding to request-quote, order-contract, invoice-payment) are related to each other. These relations cannot be expressed explicitly in WSDL. Hence, the concept of service architecture as well as communication protocols in e-business processes should be discussed.

According to the SOA paradigm, services and clients are engaged in several activities, i.e., publication, discovery and service invocation. Usually, the invocation is preceded by a negotiation for a service level agreement (SLA) that is (by its nature) "output" based, i.e. the result of the service, as required by the client, is the subject of the agreement. The SLA negotiations require semantic interoperability, that is, understanding between the client and the service provider. Actually all activities related to how a service is used by a client (also publication and discovery) require understanding between the client and the service provider. These activities must be realized in the form of communication protocols as it is usually done in distributed systems. Since the system is supposed to be open and heterogeneous, the semantic interoperability cannot be hard-coded in the protocols. It must be incorporated in the contents of the protocol messages. The contents must be expressed in a generic declarative language. The client task as well as the service description, and the output (final result) description of service performance must be expressed in this language. The important question is what this language is about? In other words, what is the grounding, i.e., the real semantics (not an abstract model-theoretic one) of the language. In our approach the grounding is a class of XML documents that services process (input documents and output documents). Once the documents are processed, they have precisely defined impact in the real world. Since this impact is hard (perhaps not possible) to describe formally, the proposed grounding is simple and sufficient to provide semantic interoperability. OWL was chosen as the language for describing XSD schemata of the documents processed by services. In our previous research projects the language was Entish [5], our own invention. The rest of the paper is devoted to describe our approach more precisely.

The structure of the paper is as follows. In Section 2 our approach to e-businesses is presented with OWL as the description language. Section 3 presents our discussion on active services and Task Oriented Architecture. Since the paper presents the work in progress, the conclusion is short and indicates only some future work.

2 Our Approach to SOA with OWL as the Description Language

The key assumption of our approach is that a service may be described externally, that is, by the local change in the environment, caused by a performance of the service. The very *local change* is usually described by a precondition and a post-condition (effect) of the performance. The environment should be represented in a formal way, and then described in a formal language. In the case of business processes, the environment is represented by documents that are exchanged and processed by the partners involved in the process. The language describes the documents. The crucial notions are: *environment of interaction between services and clients*, its *representation* denoted by (Service Environment Representation, SER for short), and *a language describing the representation*. The language is grounded (has semantics) in the environment representation (SER). The OWL syntax may be used as the syntax of the description language. Although OWL has well defined model-theoretic semantics (see <http://www.w3.org/TR/owl-semantics/>), the proposed semantics for the description language (based on the OWL syntax) has another semantics; it is SER. For this very reason the description language used in our approach is denoted by OWL+SER. OWL+SER is complementary to WSDL where the input and output documents, to be processed by a service operation, are defined. Actually, these documents are also in SER. OWL+SER allows to specify service precondition and effect as OWL formulas in the same way as it is done in OWL-S, however, in the case of OWL+SER the precondition formula describes the input documents of the WSDL operations of the service, whereas the effect formula describes the corresponding output documents of the operations.

In fact, we follow the IOPE (Input Output Preconditions and Effects) approach from OWL-S. However, contrary to OWL-S and its *ProcessModel*, the internal service structure cannot be specified; it is treated as a *black box*.

OWL+SER is also used to define tasks to be accomplished by (composite) services. Tasks are expressed in a declarative way as a pair of OWL formulas (initial situation formula and intention formula) describing the client initial situation, and a final situation intended by the client. This is similar to (*precondition, effect*) of the service description in OWL-S. OWL+SER serves also to arrange necessary conditions (to be fulfilled by the client) for service invocation in order to accomplish the clients task.

2.1 Service Architecture

WSDL is regarded as the standard for describing web services. General structure of service in WSDL 2.0 consists of the following elements:

- The data types used by the web service.
- The abstract interface as a set of abstract *operations*, each operation representing a simple interaction between the client and the service. Each operation also specifies a message exchange pattern that indicates the sequence

in which the associated messages are to be transmitted between the parties. Usually, there are input and output messages.

- The communication protocols used by the web service.

The usual non automatic use of a business service is decomposed onto the four following phases:

1. The service requester sends a query to a service specifying roughly what she/he wants from the service, and then gets back a quotation (a pro forma invoice) from the service provider. The quotation specifies details of what (and how) the service can be performed for the requester.
2. Having the quotation, the requester creates an order and sends it to the provider. Usually, the provider replies with an appropriate contract.
3. If the service is performed according to the contract, the provider sends expertise or carriage letter to the requester, whereas the requester sends back a delivery note (or acknowledgement of receipt) or a complaint.
4. Finally, the service provider sends invoice and the requester realizes the payment for the service performance.

Sometimes, after sending the order, the service provider sends back an invoice. In this case the payment must be done before service delivery.

In order to automate the use of a service, all those four phases above must be automated. For any specific type of service this may be done by a dedicated software application implemented, for example, in BPEL. Note that in each of the phases above, there is a document flow, that is, query and quotation in the first phase, order and contract in the second phase, carriage letter, delivery note, and complaint in the third phase, and finally invoice and payment. These documents can be created in XML. For each of the phases, the document flow can be described in WSDL. Finally, the complete process of using a single service can be implemented in BPEL. This can be done (hardcoded) separately for any service type in a dedicated way. However, our goal is to do so generically, that is, to construct tools that allow automation of service use for any service type.

Note that all the above phases are interrelated with each other, that is, a request (in the second phase) is created on the basis of the quote from the first phase, whereas the carriage letter, delivery note, complaint and payment are strictly related to the contract. Each of the phases must be implemented as a separate WSDL operation. There are no means to explicitly express these interrelations in WSDL. An implicit way to do so is to use the same types of elements in documents from different phases. However, to automate the service use these interrelations must be explicitly expressed in a formal language; in our approach the language is OWL+SER. These interrelations, as OWL formulas, can be processed automatically.

The first phase is of special interest for the automation in question. Actually it concerns SLA (Service Level Agreement) negotiations. The agreement is about the service output, i.e. the result of the service required by the client. Hence, the negotiations are about the contract, delivery, complaint, and payment. It is natural to have a generic language for such negotiations independent of the type

of service to be used. The language should describe documents to be processed in the next phases; it is clear that it must be OWL+SER. Then, the negotiation may proceed as follows. Given its task, the client sends its *intention formula* (a OWL formula) to a service, describing roughly what is to be accomplished by the service. This intention formula corresponds to the query in the first phase of business service. Service sends back a commitment consisting of two OWL formulas: *precondition formula*, and *effect formula*. This corresponds to the quote from the first phase. The precondition allows to create, by the client, XML-document required as input by the service. Actually, the precondition formula proposes several options (one to be chosen by the client) of the task accomplishment, and describes the input document corresponding to the client choice. The effect formula specifies the service output with relation to the input. Actually, the effect formula implies (logically) the client intention formula. It means that the intention formula describes (roughly) the output document, whereas the effect formula describes the output completely, usually, in several options. If one of the options is chosen by the client, SLA is set and its terms are incorporated in the order and in the contract (or invoice) in the next phases.

The conclusion from the above discussion is the following. The first phase of service use (corresponding to SLA negotiation) can be done, in a universal automatic way, using OWL-SER language. Based on this, the next phases of service use can also be automated. The crucial points of the proposed approach are as follows.

- Service interaction environment is represented by XSD schemata of the XML documents processed in the order-contract phase, and in the invoice-payment phase of service use.
- The language (OWL+SER) describing the representation (i.e., XSD schemas) is based on the syntax of OWL. The grounding (semantics) of the language is constituted by the above XSD schemata.
- Query and quote (of the first phase of the service use) are expressed in this language.

An automation of the use of a single service is relatively simple. If, however, several single services must be composed to perform a complex task, then the problem is how this composition can be done automatically and in a universal way.

Once the first phase is realized according to our approach, the automatic service composition is possible to some extent. The composition is done in the following three steps.

- **Step 1:** Given an initial situation and a goal (that constitute together a request) expressed in OWL+SER, construct generic plans that may realize this goal starting with the initial situation.
- **Step 2:** Choose one generic plan and discover appropriate services. Then, arrange them into a workflow. The arrangement is done as query-quote phase in OWL-SER, and corresponds to SLA negotiations.
- **Step 3:** Present the workflow to the client specifying the initial situation more precisely. Once the initial situation is satisfied by the client, execute

the workflow, control the execution, and realize the distributed transaction, if the execution is successful.

Note, that if a business process is more complex, the client interaction may be necessary in the Step 3. In this case, depending on a client decision, a new task must be realized, and the Steps 1, 2, 3 must be repeated.

To realize the Step 1, an automatic reasoning for plan construction is needed. There is a considerable work done in this area, see for example PDDL (planning domain definition language) [6] for Estimated-regression planning, and SHOP2 domain-independent planning system [7].

The Step 2 is realized by sending queries (called also intentions in our approach) and getting back quotations (called also commitments in our approach). Step 3 may be realized by using a process modeling language, i.e., by BPEL.

In our approach we propose a universal protocol for accomplishing simple client's requests (that do not require client interactions) according to the Step 2 and Step 3. Generally, the protocol specifies message exchange between services, agents (realizing the requests on behalf of the clients) and service registry.

The protocol consists of two general phases: The first one is called query phase (corresponding to SLA negotiations) whereas the second one is called execution phase. The query phase consists of the following two steps:

1. The agent sends a query (intention) to the service specifying the desired output (effect).
2. Then, the service answers with the quotation (commitment), i.e., specification of the input (precondition) required to produce the desired output.

The execution phase consists of the following three steps:

1. The agent (dedicated for the task accomplishment) creates input data (an order) according to the quotation and sends it to the service.
2. The service receives the order and produces output data (a contract) that is sent back to the agent.
3. If the contract is fulfilled, the next phases (i.e., delivery note, and payment) follows.

The previous versions of our approach (where Entish was used as the description language instead of OWL+SER) were verified by several prototype implementations. The first implementation (called enTish [8]) focused merely on composition of services that process data, i.e., services were ordinary software applications. The last completed implementation (called ELA-enT [9]) is dedicated to realization of Electronic Market for simple services. The complete description language Entish can be found in [5] and on the project web site [8].

3 Active Services and Task Oriented Architecture

The classical version of the SOA paradigm has its origins in software engineering. From the point of view of e-business processes (that are supposed to be also software applications), the concept of service need not be related to RPC (as it is in

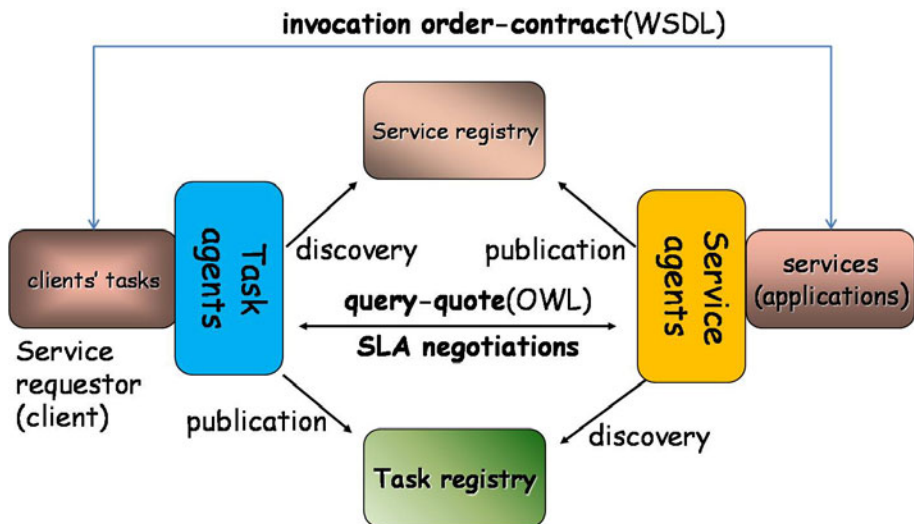


Fig. 2. SOA and TOA together

SOA) where a service is passive and is waiting for a client to be invoked. In other words, a service may be active and looking by itself for clients tasks that can be accomplished. This corresponds to the reverse auctions in business practice. In the SOA paradigm the services must be described (as it is done in WSDL and OWL) in order to be published in a service registry, and discovered there by clients. Task oriented architecture, as opposed to Service Oriented Architecture, requires tasks to be expressed declaratively. These tasks can be published by the clients in a special task registry, and discovered there by services. Unfortunately the use of the term *Task Oriented Architecture* is a bit restricted by the US patent [10].

In Fig. 2, the classic version of the SOA paradigm (presented in Fig. 1) is augmented with its TOA counterpart. That is, several components are added: Task repository for task publication and discovery, as well as task agents and service agents, representing clients and service providers (respectively) in business processes. These agents together with their interactions, constitute a multiagent system that is interesting to investigate by itself. One of the fundamental points of our approach is the separation of the query-quote phase (corresponding to SLA negotiations) and its generic realization in declarative language (as shown in Fig. 2) from the execution phase (invocation order-contract in WSDL). The execution phase is realized by direct interaction between client and service on the basis of the service level agreement done in the query-quote phase. All the interactions corresponding to task publication and discovery, service publication and discovery, as well as SLA negotiation are delegated to task agents and service agents. It seems that this very separation makes the revised SOA paradigm more generic and flexible.

3.1 A Revision of the SOA Paradigm

The main and starting point of the proposed SOA revision is that a service, in the SOA paradigm, should be considered as a business service. It has well defined structure corresponding to the following consecutive phases of service use: query-quote, order-invoice, payment, and so on. These phases may correspond to operations in WSDL. However the relations between the phases cannot be expressed in WSDL explicitly. Since a service may be active (as opposed to a passive service as it is in classic SOA), a declarative language is necessary to express client tasks. Once we have such declarative language; it may also be used to describe service interfaces in a declarative way. Although, the WSDL (by its name) is supposed to describe Web services, actually it serves to define XSD schema of the input and output documents processed by service operation.

Separation between WSDL and the description language is crucial in the proposed revision. Although this was done in Semantic Web services in OWL-S, the semantics of OWL is not directly related to the WSDL, that is, to the XSD schema of the documents processed by services. In our approach, introduction of the notion SER (Service Environment Representation) as the direct grounding (concrete semantics) of the description language, allows to explicitly express the relations between the phases of service use.

Fig. 3 summarizes the proposed approach and a revision of the SOA paradigm. The basis is the representation of service interaction environment consisting of XSD schemata of the documents processed by services and clients presented respectively in the right and the left side of the figure. In the next layer there is OWL+SER (grounded in the layer below) as a declarative language for expressing client tasks and service interfaces. In the third layer the separation between

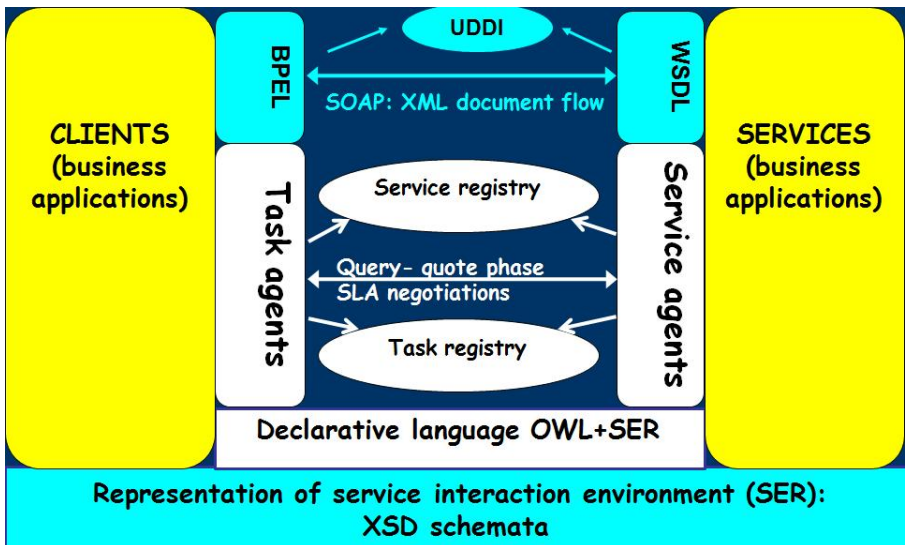


Fig. 3. The general architecture of our approach

the execution phase (XML document flow by BPEL, SOAP, WSDL, and UDDI), and SLA negotiation phase (as a multiagent system) is shown.

4 Preliminary Conclusions

We have described our ongoing project that aims at creating tools to automate (as much as possible) modeling, planning, constructing, execution and control of sophisticated business processes. The work is in progress, and is based on the approach presented above. To verify our approach as well as the tools, two testing environments are being built. The first one concerns crisis management, whereas the second one for business processes related to real estate development.

The idea presented in the paper seems to be a novel approach to SLA negotiations as well as to e-business processes. A potential application of the proposed solution may improve automation of modeling, creation and executions business processes for accomplishing clients tasks.

Acknowledgements. The work was done within the framework of the IT-SOA POIG01.03-01-00-008/08-00 project supported by the European Commission and the Polish Government.

References

1. IBM Services Architecture Team, Web Services architecture overview: The next stage of evolution for e-business (2000), <http://www.ibm.com/developerworks/webservices/library/w-ovr/>
2. Peltz, C.: Web Services Orchestration and Choreography. *Computer* 36(10), 46–52 (2003)
3. Davenport, T.: *Process Innovation: Reengineering work through information technology*. Harvard Business School Press, Boston (1993)
4. Napagao, S.A., Biba, J., Confalonieri, R., Dehn, M., Kollingbaum, M., Jakob, M., Oren, N., Panagiotidi, S., Solanky, M., Salceda, J.V., Willmott, S.: Contract based electronic business systems state of the art. In: IST Contract Project (April 10, 2007)
5. Ambroszkiewicz, S.: Entish: A language for describing data processing in open distributed systems. *Fundamenta Informaticae* 60(1-4), 41–66 (2004)
6. McDermott, D.: PDDL - the planning domain definition language. In: *The AIPS 1998 Planning Competition Committee* (1998), <http://www.cs.yale.edu/homes/dvm>
7. Wu, D., Sirin, E., Hendler, J., Nau, D., Parsia, B.: Automatic Web Services composition using SHOP2. In: *Workshop on Planning for Web Services* (June 2003)
8. Project enTish, <http://www.ipipan.waw.pl/mas/>
9. Project ELA-enT, <http://ent.ipipan.waw.pl>
10. Apparatus, method and architecture for task oriented applications (Fujitsu Limited). Estimated Patent Expiration Date: October 30, 2018: US Patent 6442749 - US Patent Issued on August 27, 2002 (2002)

Semantic Annotation of RESTful Services Using External Resources

Victor Saquicela, Luis. M. Vilches-Blázquez, and Óscar Corcho

Ontology Engineering Group, Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{vsaquicela, lmvilches, ocorcho}@fi.upm.es

Abstract. Since the advent of Web 2.0, RESTful services have become an increasing phenomenon. Currently, Semantic Web technologies are being integrated into Web 2.0 services for both to leverage each other strengths. The need to take advantage of data available in RESTful services in the scope of Semantic Web evidences the difficulties to cope with syntactic and semantic description of the services.

In this paper we present an approach to tackle the problem of automatic the semantic annotation of RESTful services using a cross-domain ontology, a semantic resource (DBpedia) and additional external resources (suggestion and synonyms services) to annotate the parameters of the RESTful services. We also present a preliminary evaluation that proves the feasibility of our approach and highlights that it is possible to carry out this semantic annotation with satisfactory results.

Keywords: RESTful service, semantic annotation.

1 Introduction

In recent years, since the advent of Web 2.0, RESTful services have become an increasing phenomenon. They also play an important role in the Semantic Web by providing data to semantic software agents, as can be seen in [10, 12]. This rapid growth of RESTful services available on the Internet and the fact that the majority of the existing service descriptions have no semantic annotations, makes it possible to think of semantic description activities for them.

However, using RESTful services still requires much human intervention since the majority of their description pages are usually given in the form of unstructured text in a Web page (HTML), which contains a list of the available operations, their URIs and parameters (also called attributes), expected output, error messages, and a set of examples. The description includes all the details needed for a developer to execute the service or use it in applications such as mashups [3, 20, 21].

Traditionally, service semantic annotation approaches have focused on defining formalisms to describe these services [1, 5, 8]. These approaches take into account the description page of a RESTful service to carry out their semantic annotation processes.

The vast majority of RESTful services being reasonably well documented with respect to the functionality, programmers have been encouraged to supply an HTML

page with their services. Likewise, many approaches have resolved basic problems of RESTful service semantic description, but all related processes with RESTful service annotation are manual. Different examples of them can be found in [2, 11]. This is one of the main challenges of RESTful services that needs to be addressed in order to provide automation of semantic annotation tasks and to be able to interoperate with others applications or services.

In this paper, we focus on two main challenges: (1) to provide syntactic descriptions of RESTful service that allow their automatic registration and invocation, and (2) to interpret and enrich the RESTful services' parameters, by means of their semantic annotation.

Our main contribution is the proposal of an automatic approach for the semantic annotation of RESTful services using diverse types of resources: a cross-domain ontology, DBpedia, and diverse external services, such as suggestion and synonyms services.

The remainder of this paper is structured as follows: Section 2 presents related work in the context of semantic annotation of Web services and RESTful services. Section 3 introduces our approach for the automatic annotation of RESTful services, including deriving their syntactic description and semantic annotation. Section 4 presents a brief experimentation of our system. Finally, Section 5 presents some conclusions of this paper and future work.

2 Related Work

Most research in the semantic annotation of RESTful services has focused on the definition of formal description languages for creating semantic annotations. The main proposed formalisms for describing these services are: the Web Application Description Language¹ (WADL) which describes RESTful services syntactically, MicroWSMO [3] which uses hREST (HTML for RESTful services) [3, 5], and SA-REST [2, 8] which uses SAWSDL [1] and RDFa² to describe service properties.

In [19] the authors introduced an approach to annotate WADL documents linking them to ontologies. Among these approaches, some authors propose rather heavy-weight approaches for semantic description, which are normally derived from Web Service (WS-*) semantic description frameworks like WSMO or OWL-S. An example is proposed in [10], which makes use of a specific selection of existing languages and protocols, reinforcing its feasibility. Firstly, OWL-S is used as the base ontology for services, whereas WADL is used for syntactically describing them. Secondly, the HTTP protocol is used for transferring messages, defining the action to be executed, and also defining the executing scope. Finally, URI identifiers are responsible for specifying the service interface. Nevertheless, these languages are strongly influenced by existing traditional Web Services.

Other approaches are more lightweight, for instance, the proposals of [1, 2]. The authors advocate an integrated lightweight approach for formally describing semantic RESTful services. This approach is based on use of the hREST and MicroWSMO

¹ <http://www.w3.org/Submission/wadl/>

² <http://www.w3.org/TR/xhtml-rdfa-primer/>

microformats, which enable the creation of machine-readable service descriptions and the addition of semantic annotations. Furthermore, the authors present SWEET, a tool which effectively supports users in creating semantic descriptions of RESTful services based on the aforementioned technologies.

Our work can be considered as an extension of the work presented in [13, 14], in which the development of domain-independent approaches to semantically label Web services is described. The authors propose to automatically learn the semantics of information sources labelling the input and output parameters used by the source with semantic types of the user's domain model. In our approach we have dealt with RESTful services and have used semantic repositories to try to semantically label these services.

Another approach is presented in [17]. This approach classifies data type using HTML treated Web form files as the Web service's parameters. They use Naïve Bayes to classify assigned semantic types to the input and output parameters.

Likewise, our work is also similar to the approaches related to schema matching or integration [18, 16]. In these proposals the main goal is to establish semantic mappings between two different schemas. In our approach the developed system sets matchings between different parameters of a RESTful service and the DBpedia ontology³.

3 An Approach for Automatic Semantic Annotation of RESTful Services

In this section, we present our approach visualized in Figure 1 for automating the syntactic and semantic annotation of RESTful services. Our system consists of three main components, including invocation and registration, repository, and semantic annotation components, which are enriched by diverse external resources. Next, we briefly describe the different components, illustrating the descriptions with some sample services on the geographical domain.

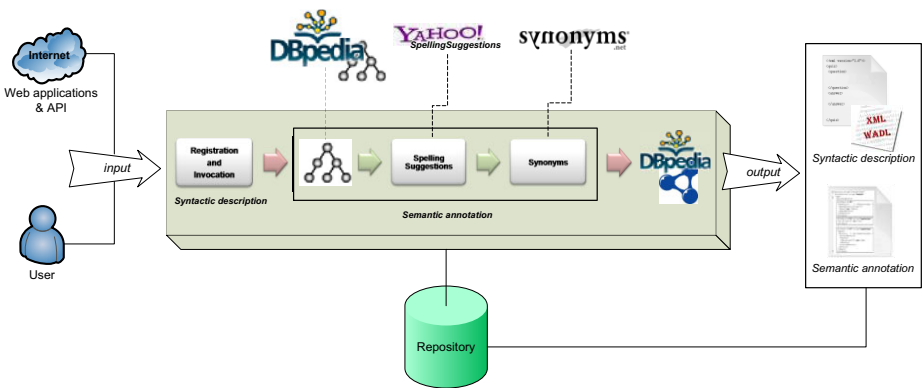


Fig. 1. RESTful Service Semantic Annotation System

³ <http://wiki.dbpedia.org/Ontology>

3.1 A Sample Set of RESTful Services in the Geospatial Domain

The following services are two representatives of RESTful services in the geospatial domain taken from programmableweb.com:

- **Service 1.** <http://ws.geonames.org/countryInfo?country=ES>

This service retrieves information related to 'country', specifically, it returns information about the following parameters: 'capital', 'population', 'area' (km²), and 'bounding box of mainland' (excluding offshore islands).

- **Service 2.** http://api.eventful.com/rest/venues/search?app_key=p4t8BFcLDtCzpxdS&location=Madrid

This service retrieves information about places (venues), specifically, it returns parameters like: 'city', 'venue_name', 'region_name', 'country_name', 'latitude', 'longitude', etc.

3.2 Syntactic Description: Invocation and Registration

Our system takes as input Web applications and APIs, which are known by users, or users can add manually a URL of an available RESTful service. In this case, we add manually different URLs of services and obtain automatically information related to each of the aforementioned RESTful service. Once URLs have been added, our system invokes the RESTful service with a sample of parameters and analyzes the response to obtain a basic syntactic description of a parameter set, which is used like inputs and outputs.

In this process our system uses the Service Data Object⁴ (SDO) API to perform the invocation of the RESTful service and determine whether it is available or not. SDO is a specification for a programming model that unifies data programming across data source types and provides robust support for common application patterns in a disconnected way [22]. The invocation process is performed as follows: first, it takes the input parameters and their values, which are given to the service as part of a URL. Then, the system invokes the service which translates our "RESTful service call" into a query to specific service, including the URL and related parameters.

The service invocation of a specific RESTful service may return diverse formats, such as HTML, JSON, XML, etc. In our work we use only XML response for describing the service. The results of invocation of both services are showed in Table 1.

These XML responses are processed using SDO, which enables to navigate through the XML and extract output parameters of each service. The result of this invocation process is a syntactic definition of RESTful services in XML, which can be expressed in description languages like WADL or stored into a relational model. In this work we use a relational model as data model as a consequence of the simplicity of WADL for showing concepts. Table 2 shows the different output parameters of each service.

The output parameters are registered and stored into a repository. This repository is a database specifically designed to store syntactic descriptions of RESTful services. We selected this storage to increase efficiency in the recovery of the RESTful services.

⁴ <http://www.oasis-opencsa.org/sdo>

Table 1. XML response of two sample RESTful services

Service 1	Service 2
<pre> <geonames> <country> <countryCode>ES</countryCode> <countryName>Spain</countryName> <isoNumeric>724</isoNumeric> <isoAlpha3>ESP</isoAlpha3> <fipsCode>SP</fipsCode> <continent>EU</continent> <capital>Madrid</capital> <areaInSqKm>504782.0</areaInSqKm> <population>40491000</population> <currencyCode>EUR</currencyCode> <languages>es-ES,ca,gl,eu</languages> <geonameId>2510769</geonameId> <bBoxWest>-18.169641494751</bBoxWest> <bBoxNorth>43.791725</bBoxNorth> <bBoxEast>4.3153896</bBoxEast> <bBoxSouth>27.6388</bBoxSouth> </country> </geonames> </pre>	<pre> <venue id="V0-001-000154997-6"> <url>http://eventful.com/madrid/venues/la- ancha-/V0-001-000154997-6</url> <country_name>Spain</country_name> <name>La Ancha</name> <venue_name>La Ancha</venue_name> <description></description> <venue_type>Restaurant</venue_type> <address></address> <city_name>Madrid</city_name> <region_name></region_name> <region_abbr></region_abbr> <postal_code></postal_code> <country_abbr2>ES</country_abbr2> <country_abbr>ESP</country_abbr> <longitude>-3.68333</longitude> <latitude>40.4</latitude> <geocode_type>City Based GeoCodes </geocode_type> <owner>frankg</owner> <timezone></timezone> <created></created> <event_count>0</event_count> <trackback_count>0</trackback_count> <comment_count>0</comment_count> <link_count>0</link_count> <image></image> </venue> <venue id="V0-001-000154998-5"> </pre>

Table 2. Syntactic description of RESTful service

Service 1:
countryInfo(\$country,bBoxSouth,isoNumeric,continent,fipsCode,areaInSqKm,languages, isoAlpha3,countryCode,bBoxNorth,population,bBoxWest,currencyCode,bBoxEast,capital, geonameId,countryName)
Service 2:
rest/venues/search(\$location,\$app_key,id,link_count,page_count,longitude,trackback_count, version,venue_type,owner,url,country_name,event_count,total_items,city_name,address,name, latitude,page_number,postal_code,country_abbr,first_item,page_items,last_item,page_size, country_abbr2,comment_count,geocode_type,search_time,venue_name)

3.3 Semantic Annotation

Once the RESTful service is syntactically described with all its identified input and output parameters, we proceed into its semantic annotation. We follow a heuristic approach that combines a number of external services and semantic resources to propose annotations for the parameters as show in Figure 2. Next, we describe the main components of the semantic annotation.

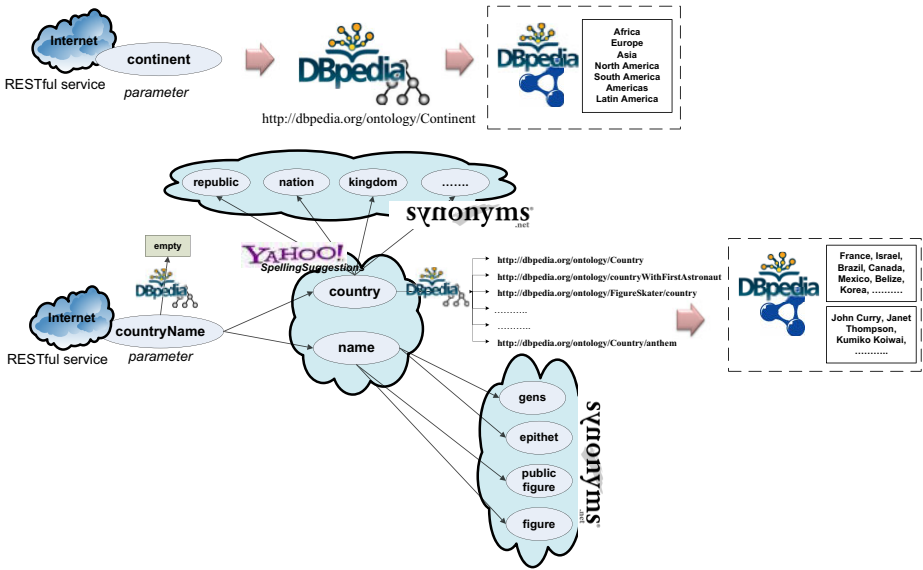


Fig. 2. Semantic annotation process

3.3.1 Using DBpedia in the Semantic Annotation

Currently, the RESTful service semantic annotation has some difficulties, which are briefly described in [1, 11]. In order to cope with them, we rely on techniques and processes that permit: a) semantic annotation only using syntactic description and, input/output parameters, or b) identification of some right example values that allow the invocation RESTful service automatically.

The starting point of the semantic annotation process is the list of syntactic parameters obtained previously. These parameters are used to query the DBpedia SPARQL Endpoint and retrieve the associated results to each parameter, as follows:

- First, the system retrieves all the classes from the DBpedia ontology whose names have an exact match with each parameter of the RESTful service. If the system obtains correspondences from the matching process, it uses these DBpedia concepts individually to retrieve samples (concept instances) from the DBpedia SPARQL Endpoint. The resulting information (RDF) is suggested automatically to the system and registered as a possible value for a certain parameter. When a parameter matches more than once in the DBpedia ontology, our system only considers concepts that have information (instances), and automatically discards those ontology concepts without instances.

In order to retrieve information about identified parameters of RESTful services the system has registered the DBpedia SPARQL Endpoint as a service. This service enables automatically invocation of SPARQL queries over DBpedia Endpoint. Next, we present queries used by the system for retrieving DBpedia information.

Table 3. SPARQL query for retrieving classes of the DBpedia ontology (*filter omitted*)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?class where { ?class rdf:type owl:Class. FILTER ..... }
```

This SPARQL query (see Table 3) enables to retrieve classes of the DBpedia ontology. The results of this query are compared to the concepts with each parameter of a service.

- Next, the system tries to find correspondences between parameters of the RESTful service and DBpedia properties. If the system obtains some correspondences, it uses these DBpedia properties individually to retrieve information of the DBpedia SPARQL Endpoint. Furthermore, this information is registered as a possible right value for a certain parameter.

This SPARQL query (see Table 4) enables to retrieve properties of the DBpedia ontology. The system uses results to compare them with each parameter identified in the syntactic description.

Table 4. SPARQL query for retrieving properties of the DBpedia ontology (*filter omitted*)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?property where { ?property rdf:type owl:ObjectProperty. FILTER ..... }
```

- Finally, with the classes and properties matched, the system calls the DBpedia SPARQL Endpoint to retrieve values (instances) for classes and properties. Next we show some query examples:

Table 5. SPARQL query for retrieving possible values for a class

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?val where { <" + c + "> a owl:Class. ?val a <" + c + "> }
```

This SPARQL query (see Table 5) enables to retrieve possible values for a certain class of the ontology.

Table 6. SPARQL query for retrieving possible values for a property

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?val where { <" + c + "> a owl:ObjectProperty. ?val <" + c + "> ?b }
```

This SPARQL query enables to retrieve possible values for a certain property of the ontology.

3.3.2 Enriching the Semantic Annotation

Since we request exact matches with DBpedia classes and properties, our system does not normally establish correspondences with ontology classes or properties for all parameters of the RESTful service. In order to annotate semantically the parameters that did not match any DBpedia resource, we add different external services to enrich the results. Below we describe the main characteristics of the external services added to the system.

Spelling Suggestion

Web search engines (e.g. Google, Yahoo, and Microsoft) usually try to detect and solve users' writing mistakes. The suggestions services, also called "Did You Mean", are spelling algorithms which solve these mistakes. For example, when a user writes 'countryName' these algorithms suggest 'country' and 'name' separately.

In our system we use the Yahoo Boss service⁵ to retrieve suggestions about the parameters. Thus, for each parameter that the system did not find a correspondence with classes or properties, this service is invoked for obtaining a list of suggestions to query DBpedia again. The output is registered and stored into the repository. Following the previous example, the parameter 'countryName' is not found in the DBpedia ontology. Nevertheless, the added service allows separating this parameter in 'country' and 'name', and then it calls to the DBpedia SPARQL Endpoint for obtaining results.

Use of Synonyms

This external service⁶ is incorporated to the system to retrieve synonyms of a certain parameter. This service tries to improve the semantic annotation process when our system does not offer results for the previous steps, that is, when we still have parameters in a RESTful service without annotations.

In the next example we find a parameter called 'address'. The invocation process uses the synonyms service to retrieve a set of synonyms of 'address' such as extension, reference, mention, citation, denotation, destination, source, cite, acknowledgment, and so on. These outputs are registered and stored into the repository, and then, the service calls to the DBpedia SPARQL Endpoint for results.

3.4 Checking the Semantic Annotation of RESTful Services

In order to check the collected semantic annotations of the previous process our system invokes the RESTful service, which was registered previously (as we describe in Section 3.2) with a random value of instances obtained from the queries to the DBpedia SPARQL Endpoint. If the system collects an instance value from the DBpedia SPARQL Endpoint and it is not empty, then our system considers that the invocation response is right based on the syntactic description. The system does not check all the collected instances as a default option, because there are many amount RESTful services with invocation limitations.

⁵ http://developer.yahoo.com/search/boss/boss_guide/Spelling_Suggest.html

⁶ <http://www.synonyms.net/>

The correspondences established between different parameters of a RESTful service and the DBpedia ontology (classes and properties) are registered and stored in the repository. In this way, the RESTful service is annotated semantically and it will allow generating semantic documentation of the type of service. An example of this can be seen in Table 7. This repository is a database specifically designed to store semantic annotations of RESTful services. As mentioned above, this storage is selected to increase efficiency in the recovery of RESTful services.

Table 7. Semantic annotation of a RESTful service

```
( $country, bBoxSouth, isoNumeric, http://dbpedia.org/ontology/Continent, fipsCode, http://dbpedia.org/property/areaMetroKm, languages, isoAlpha3, http://dbpedia.org/ontology/country, bBoxNorth, http://dbpedia.org/ontology/populationDensity, bBoxWest, http://dbpedia.org/ontology/Currency, bBoxEast, http://dbpedia.org/ontology/capitalgeonameId, http://dbpedia.org/ontology/country)
```

4 Experimental Results

In order to evaluate our approach we use 12 different RESTful services founded in <http://www.programmableweb.com/>, which are characterized to contain geospatial information. The list of RESTful services can be seen in this website⁷.

This analysis follows the three steps described in the semantic annotation. First, our system identifies correctly 16 parameters calling directly the DBpedia ontology but it fails to recognize 161 parameters. Second, the system uses the suggestion service and calls the DBpedia ontology. In this case, it identifies 41 correspondences, but it fails to recognize 120 parameters. Third, the system uses the synonyms service and calls the DBpedia ontology. It identifies 19 correspondences, but fails to recognize 101. A detailed view of these results is shown in Table 8.

Table 8. Results of the service test

RESTful service	Parameters	DBpedia ontology	Remaining parameters	Suggestions service	Remaining parameters	Synonyms service	Annotated parameters
Source1	17	3	14	3	11	2	8
Source2	24	2	22	7	15	1	10
Source3	7	0	7	3	4	1	4
Source4	13	2	11	5	6	0	7
Source5	14	1	13	2	11	0	3
Source6	11	1	10	4	6	1	6
Source7	7	0	7	0	7	0	0
Source8	8	1	7	1	6	0	2
Source9	43	1	42	14	28	5	20
Source10	4	0	4	2	2	1	3
Source11	7	0	7	0	7	0	0
Source12	22	5	17	0	17	8	13
Total	177	16	161	41	120	19	76

⁷ <http://castor.dia.fi.upm.es/ev/RESTfulservice.html>

We cannot guarantee the success of the system in all the cases, because in some cases the system has not found any correspondence between RESTful service parameters and the concepts or properties of the DBpedia ontology.

After having analysed our findings, we have seen that some parameters are useless, because they refer to a navigation process through RESTful service results, for example: page, total, hits, etc. These parameters make it difficult to carry out the right annotation semantic process. We are planning to discard these types of parameters in the future.

To the best of our knowledge, there are no available results from existing research works to compare our results against. Likewise, these preliminary results prove the feasibility of our system and highlight that is possible to carry out an automatic semantic annotation of RESTful services.

5 Conclusions and Future Work

In this paper we have proposed an approach to perform an automatic semantic annotation process of RESTful services. This process is implemented in a system which takes into account the DBpedia ontology and its SPARQL Endpoint, as well as different external resources such as synonyms and suggestion services. We use combinations of these resources to discover meanings for each of the parameter of the RESTful services and perform semantic annotations of them.

In this work we use two different RESTful services related to the geospatial domain to guide the explanation of the proposed semantic annotation process. Finally, we have presented some preliminary experimental results that prove the feasibility of our approach and show that it is possible to carry out a semantic annotation of RESTful services automatically.

Future work will focus on the development of a GUI that will connect to services provided for the system. This online tool will be able to register and invoke new RESTful services, and annotate a lot of existing RESTful services with semantics. This can be useful for creating new applications (for instance, mashups) or for its use in the Semantic Web. Moreover, we also plan to make several improvements to the proposed system, related to the matching process and the use of similarity metrics. In the same sense, we also aim at improving in the SPARQL queries to DBpedia to better explore the knowledge of this resource in the annotation process, and optimize the use of suggestion and synonyms services.

Acknowledgments

This work has been supported by the R&D project España Virtual (CENIT2008-1030), funded by Centro Nacional de Información Geográfica and CDTI under the R&D programme Ingenio 2010. We would also like to thank Boris Villazón-Terrazas for his valuable comments.

References

1. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services. In: Workshop: Beyond SAWSDL at OnTheMove Federated Conferences & Workshops, Vilamoura, Portugal (2009)

2. Maleshkova, M., Pedrinaci, C., Domingue, J.: Semantically Annotating RESTful Services with SWEET. In: Demo at 8th International Semantic Web Conference, Washington D.C., USA (2009)
3. Maleshkova, M., Gridinoc, L., Pedrinaci, C., Domingue, J.: Supporting the Semi-Automatic Acquisition of Semantic RESTful Service Descriptions. Poster at ESWC 2009 (2009)
4. Pedrinaci, C., Domingue, J., Krummenache, R.: Services and the Web of Data: An Unexploited Symbiosis. In: Workshop: Linked AI: AAAI Spring Symposium “Linked Data Meets Artificial Intelligence” (2010)
5. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML Microformat for Describing RESTful Web Services. In: Web Intelligence 2008, pp. 619–625 (2008)
6. Lambert, D., Domingue, J.: Grounding semantic web services with rules. In: Workshop: Semantic Web Applications and Perspectives, Rome, Italy (2008)
7. Steinmetz, N., Lausen, H., Brunner, M.: Web Service Search on Large Scale. In: IC-SOC/ServiceWave 2009, pp. 437–444 (2009)
8. Lathem, J., Gomadam, K., Sheth, A.P.: SA-REST and (S)mashups: Adding Semantics to RESTful Services. In: ICSC 2007, pp. 469–476 (2007)
9. García Rodríguez, M., Álvarez, J.M., Berrueta, D., Polo, L.: Declarative Data Grounding Using a Mapping Language. Communications of SIWN 6, 132–138 (2009)
10. Freitas Ferreira Filho, O., Grigas Varella Ferreira, M. A.: Semantic Web Services: A RESTful Approach. In: IADIS International Conference WWW/INTERNET 2009, Rome, Italy (2009)
11. Alowisheq, A., Millard, D.E., Tiropanis, T.: EXPRESS: EXPressing REStful Semantic Services Using Domain Ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 941–948. Springer, Heidelberg (2009)
12. Alarcon, R., Wilde, E.: Linking Data from RESTful Services. In: Linked Data on the Web, LDOW 2010 (2010)
13. Lerman, K., Plangprasopchok, A., Knoblock, C.A.: Semantic Labeling of Online Information Sources. Int. J. Semantic Web Inf. Syst. 3(3), 36–56 (2007)
14. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.A.: Automatically Constructing Semantic Web Services from Online Sources. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 17–32. Springer, Heidelberg (2009)
15. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: SIGMOD Conference 2001, pp. 509–520 (2001)
16. Doan, A., Domingos, P., Halevy, A.Y.: Learning to Match the Schemas of Data Sources: A Multistrategy Approach. Machine Learning 50(3), 279–301 (2003)
17. Heß, A., Kushmerick, N.: Learning to Attach Semantic Metadata to Web Services. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 424–440. Springer, Heidelberg (2009)
18. Rahm, E., Bernstein, P.: On matching schemas automatically. VLDB. Journal 10(4) (2001)
19. Battle, R., Benson, E.: Bridging the semantic web and web 2.0 with Representational State Transfer (REST). Web Semantics 6, 61–69 (2008)
20. Braga, D., Ceri, S., Martinenghi, D., Daniel, F.: Mashing Up Search Services. IEEE Internet Computing 12(5), 16–23 (2008)
21. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y.H., Simmen, D., Singh, A.: Damia: a data mashup fabric for intranet applications. In: VLDB 2007: Proceedings of the 33rd international conference on Very large data bases, pp. 1370–1373. VLDB Endowment (2007)
22. Resende, L.: Handling heterogeneous data sources in a SOA environment with service data objects (SDO). In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 895–897. ACM, New York (2007)

Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction

Carlos Rodríguez, Patrícia Silveira, Florian Daniel, and Fabio Casati

University of Trento, Via Sommarive 14

38123 Povo, Trento, Italy

{crodriguez,silveira,daniel,casati}@disi.unitn.it

Abstract. Automatically monitoring and enforcing compliance of service-based business processes with laws, regulations, standards, contracts, or policies is a hot issue in both industry and research. Little attention has however been paid to the problem of *understanding* non-compliance and *improving* business practices to prevent non-compliance in the future, a task that typically still requires human interpretation and intervention. Building upon work on automated detection of non-compliant situations, in this paper we propose a technique for the root-cause analysis of encountered problems and for the prediction of likely compliance states of running processes that leverages (i) on event-based service infrastructures, in order to collect execution evidence, and (ii) on the concept of key compliance indicator, in order to focus the analysis on the right data. We validate our ideas and algorithms on real data from an internal process of a hospital.

Keywords: Compliance, Decision Trees, SOA, Root-Cause Analysis.

1 Introduction

Compliance means conformance with laws, regulations, standards, contracts, policies, or similar sources of requirements on how to run business. Effective *compliance management*, i.e., the practice of assuring compliance, is an increasingly more important concern in today's companies, since the set of compliance requirements a company has to implement grows fast and their effect on the "traditional" business practices in a company may be considerable. Despite its increasing importance, compliance is however to a large extent still managed in rather ad-hoc ways and with little or no IT support. As a result, today it is very hard for any CFO or CIO to answer questions like: Which requirements does my company have to comply with? Which processes should obey which requirements? Which processes are following a given regulation? Where do violations occur? Which processes do we have under control? And so on.

While IT has been supporting (in more or less automated fashions) the execution of business processes for long time now, in the past the adoption of ad-hoc and monolithic software solutions did not provide the necessary insight into how processes were executed and into their runtime state, preventing the adoption of IT also for compliance assessment. The advent of workflow management systems and, especially today, of web service-based business interactions and the service-oriented architecture

(SOA) have changed this shortcoming, turning business processes into well-structured, modular, and distributed software artifacts that provide insight into their internals, e.g., in terms of execution events for tasks, service calls, exchanged SOAP messages, control flow decisions, or data flows. All these pieces of information can be used for online monitoring or enforcement of compliant process behaviors or they can be logged for later assessment. Unfortunately, however, the resulting amount of data may be huge (in large companies, hundreds of events may be generated per minute!), and – especially in terms of reporting and analysis – it is not trivial to understand which data to focus on and how to get useful information out of them.

Doing so is challenging and requires answering questions like how to collect and store evidence for compliance assessment in service-based business processes, how to report on the compliance state, and how to support the analysis of non-compliant situations. But more than these, the challenges this paper aims to solve are how to collect evidence in a way that is *as less intrusive as possible*, how to devise solutions that are *as useful as possible*, yet – at the same time – *as generic as possible* and independent of the particular IT system to be analyzed, and, finally, how to provide compliance experts with information that is *as useful and expressive as possible*. In light of these challenges, this paper provides the following contributions:

- A method for the definition and a dashboard for the visualization of so-called *Key Compliance Indicators* (KCI) for at-a-glance reporting on compliance;
- An algorithm and a tool for the *mining of decision trees* from process execution logs that particularly look at data from the perspective of compliance;
- An application of the algorithm mining approach to *real-world data* stemming from a typical business process running in a large Italian hospital.

In the next section we provide the necessary details about this process and highlight its compliance requirements, so as to derive the requirements for this paper in Section 3. In Section 4 and 5, we then discuss how to report on compliance and how to analyze non-compliance, respectively. In Section 6 we discuss some related works, and in Section 7 we conclude the paper.

2 Scenario: Drug Reimbursement in Hospitals

Let us consider the case of a drug reimbursement process in the healthcare domain. The process is the case study in one of our EU projects, where we cooperate with Hospital San Raffaele (Milan, Italy), which runs the process shown in Figure 1. The overall purpose of this process, from the hospital's point of view, is to obtain reimbursements from the Italian Health Authority for the drugs dispensed to outpatients (i.e., patients that are not hospitalized). In order to obtain the reimbursement, there are many compliance requirements imposed by the Health Authority, among which we mention privacy preservation in personal information processing, separation of duties, and the adherence of standard template of dispensation reports.

The core process that generates the information that needs to be sent to the Health Authority occurs inside the Ward. The process starts when a patient visits the hospital's ward to consult a doctor. After diagnosing the patient, the doctor prepares a drug prescription that is delivered to a nurse, who is in charge of dispensing the prescribed

drugs to the patient. If the amount of drugs is going below a certain threshold, the nurse issues a drug request to the central pharmacy of the hospital, which must replenish the ward's drug stock in no later than 48 hours. The execution of this process is fully supported by the ward's SOA-based information system, and all progress events generated during process executions are recorded in an event log for later inspection.

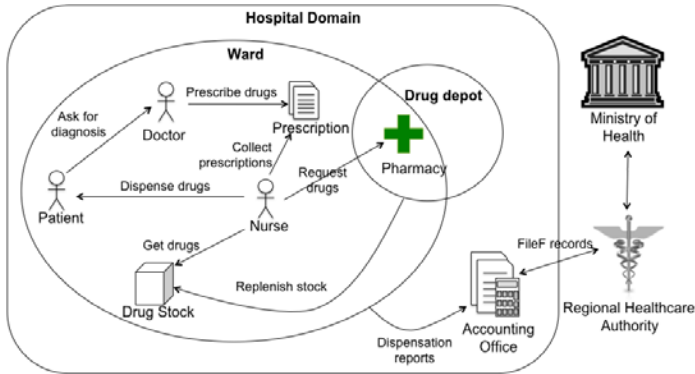


Fig. 1. Summary of the direct drug reimbursement process

While the process above is executed daily, the preparation of dispensation reports for drug reimbursement is a monthly task. That is, at the end of each month, the records of drug dispensations are collected from the various wards of the hospital and the corresponding dispensation reports to be sent to the Health Authority are created. These reports consist in simple text files (known as *FileF*) in which data about the dispensations are included. Examples of data included in these files are *hospital identification, patient, doctor, dispensed drug and quantity*, and *amount in Euros*. Whenever the report is ready it is sent to the Health Authority, which checks the quality of the report against some compliance requirements imposed on dispensation reports. For instance, one compliance requirement that decides whether a dispensation can be reimbursed or not regards the completeness and correctness of records: no *null* or incorrect data are tolerated in any field. If there are such problems in the report, the Health Authority sends a feedback to the hospital indicating the number and type of errors found for each record of the file, and, in turn, the hospital must correct them so as to get the reimbursement.

The complete reimbursement process is complex, and not complying with the applicable requirements can be costly. Therefore, in order to better control the compliance of the reimbursement process, the hospital wants to implement an early warning system that allows the hospital's compliance expert to have updated information on daily compliance issues, e.g., in form of indicators, reports, or predictions on the compliance of its processes. In addition, in case of repeated problems, it is important to understand why they happen and how they can be solved for the future. However, manually analyzing the data in the event log is time consuming and also error-prone but, still, the hospital wants to improve its compliance in order not to lose money for not reimbursed drug dispensations.

3 Service-Oriented Compliance Management: Requirements

The above scenario describes a service-based business process that is distributed over the hospital's ward and the drug depot and that asks for proper compliance management, that is, compliance assessment, reporting, and analysis.

As this paper has its roots in two EU FP7 research projects, i.e., *Compas* and *Master*, that both assist **compliance assessment** in the SOA, here we do not propose a new assessment technique and rather rely on the techniques proposed there: *Compas* (www.compas-ict.eu) strongly focuses on model-driven development of compliant processes and proposes a compliance checking approach that is based on (i) compliance requirements expressed in logical rules or process fragments and (ii) complex event processing (CEP) and business protocol monitoring to detect non-compliance with requirements. *Master* (www.master-fp7.eu), instead, specifically focuses on the security domain and proposes a two-layered approach to compliance assessment: first, it supports the CEP-based monitoring of running processes and the enforcement of individual rules; then, offline, it checks compliance of executed processes by assessing their conformance to a so-called ideal process model. Both approaches have in common the use of an instrumented service orchestration engine for the execution of business processes and the generation/logging of suitable execution events, starting from a signaling policy that specifies which events are necessary for compliance assessment.

Building on this background, **reporting on the state of compliance** requires being able to *store process execution and compliance data* and to develop a *reporting dashboard* on top, a task that we partly approached in [1]. But we also need to devise a method for the *easy specification* and, then, automated computation of *key compliance indicators* (KCIs), in order to visualize them in the dashboard. Next, the **analysis of root-causes for non-compliance** requires selecting a suitable *analysis algorithm* and – more importantly – understanding *which data* to look at, out of the huge amount of data that is available for this task, and to validate the algorithm in the context of the described scenario.

4 Reporting on Compliance

In order report on the compliance of business processes, the common approach is to visualize the compliance status at a high-level of abstraction, for instance, by means of KCIs that are graphically rendered in a compliance governance dashboard (CGD) [1]. KCIs support compliance experts with an overview of the compliance performance of business processes and can be seen as particular type of KPIs (key performance indicators) that specifically measures how compliant a process is with given requirements. A typical KCI may, for example, measure how many process instances, out of all the executed ones, satisfy a separation of duties requirement; but also a traditional QoS indicator (e.g., the average process execution time) can be seen as KCI, if we are subject to a compliance requirement regarding QoS (e.g., deriving from a contract with the customer). As we will see, KCIs also provide a starting point for finding the root-causes of non-compliance. This section explains how we store process execution data, specify and compute KCIs, and visualize them through effective visual metaphors.

4.1 Storing Process Execution and Compliance Data

The main sources of process execution and compliance data are the *event logs* generated by the execution of service-based business processes. Therefore, let us first conceptualize the key ingredients characterizing event logs, as we perceive them for our analysis. An *event* is a tuple $e = \langle t, s, ts, d, p_1, \dots, p_n, B \rangle$, where t is the type of the event (e.g., *ProcessStart*, *ActivityExecuted*, *Violation*), s is the source that generates the event, ts is a timestamp, p_1, \dots, p_n is a set of properties (e.g., event message header properties such as correlation data, process instance identifier or similar), and B is the body of the event message (e.g., containing business data needed for the computation of an indicator). Using this data, events can be grouped together by their process instance and ordered by timestamp, forming this way traces. A *trace* is a sequence of events $T_i = \langle e_{i1}, e_{i2}, \dots, e_{in} \rangle$, where i refers to a process instance identifier and n is the number of events that compose the process instance. This way, an *event log* can be expressed as a set of traces $L = \{T_1, T_2, \dots, T_k\}$, where k is the total number of traces.

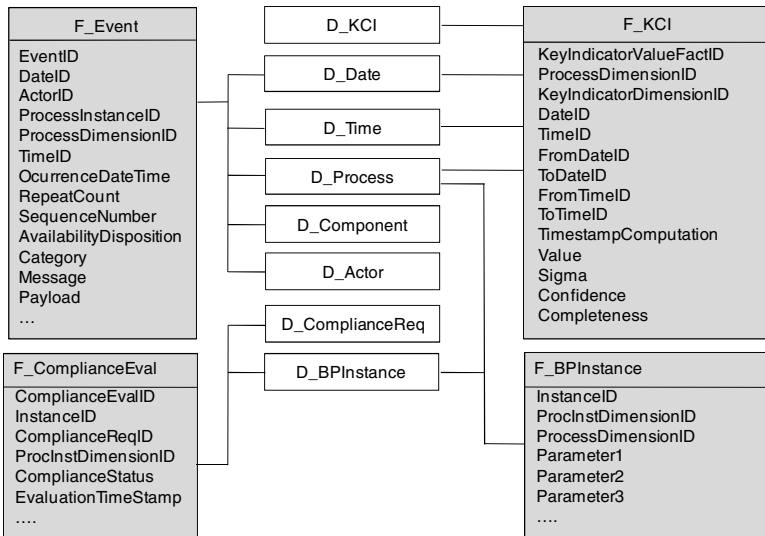


Fig. 2. Simplified schema of the data warehouse model

The events in the log are processed by Extract-Transform-Load (ETL) flows, in order to store them into a data warehouse (DW), which is modeled using a compliance-oriented dimensional data model. The reason for doing this is that we aim at leveraging the capability of dimensional models for keeping a conciliated view on the process execution and compliance data, and for supporting further analysis, e.g., by means of root-cause analysis algorithms or Online Analytical Processing (OLAP) tools. Figure 2 shows an excerpt of the schema of the DW. The tables in white are the *dimensional tables* that allow us to slice and dice through the *fact tables* (shaded gray). The fact table *F_Event* stores the events as they come from the event log, *F_KCI* stores the computed values of indicators, *F_BPInstance*, the instances of processes, and *F_ComplianceEval*, the compliance status of process instances as

computed, for instance, by the compliance checking algorithms adopted in the context of the Compas or Master projects.

The `F_BPInstance` table deserves a further explanation, as it constitutes an *abstraction* of the process execution data, and the basis for computing indicators and performing root-cause analysis. In our DW model, each business process *BP* has its own `F_BPInstance` table, or, as we call it, *process instance table* (e.g., in our scenario we have a `F_DrugDispensationInstance` table). In these tables, each row corresponds to an instance of the associated process, while columns (i.e., parameters of the process instance table) correspond to business data that are of interest for the analysis of each process. Table 1 shows a conceptual view on the process instance table for the drug dispensation process, where each row corresponds to a single drug dispensation. The *DrugType* column refers to the type of drug, *ErrPerData* indicates whether there was an error in the information about the patient, *ErrCompData* tells us if there was an error in any other complementary data, and *Compliant* tells us whether the dispensation was free of error. These parameters are obtained from the attributes of the events that are part of the event trace. Sometimes, the parameter values can be directly extracted from events without modifications (e.g., the *DrugType* parameter), while in other cases the values are obtained by performing aggregation/computations over a set of events and attributes of process instances (e.g., the *Compliant* parameter).

Table 1. Example of a process instance table for the drug dispensation process

InstanceID	DrugType	ErrPerData	ErrCompData	...	Compliant
38769	1	False	False	...	True
32537	6	True	False	...	False
27657	1	False	False	...	True
32547	2	False	True	...	False
35340	1	False	False	...	True
....

Finally, it is worth to mention that in order to populate the DW, the ETL usually needs to access other sources of data such as user management systems and human task managers, which are the main data providers for dimension tables, as opposed to event logs, which provide mostly the evidences of process executions.

4.2 Specifying and Computing Key Compliance Indicators

Generally, indicators are computed out of a variety of data and by means of different functions, ranging from the lowest business data granularity to the highest business goals. In the context of compliance assessment, a KCI is a measure (i.e., a numeric value) that quantifies compliance performance against compliance targets in a pre-determined time interval. For instance, one of the compliance requirements imposed by the Healthcare Authority is that of sending drug dispensation reports without errors in the data about dispensed drugs and patients. Whenever there is an erroneous record of drug dispensation, the corresponding drug is not reimbursed to the hospital, and, thus, it is important for the hospital to keep an eye on the accomplishment of this compliance requirement. KCIs are therefore useful means to assist this task.

KCIs can be easily specified by using the available information in Table 1. For example, a KCI may be defined as the percentage of non-compliant process instances out of all instances in the DW (and the reporting time interval). More precisely, we can use the *Compliant* column of a process instance table to compute KCIs, and we can express their respective formulas using standard SQL queries. SQL has been designed also as a language for computing aggregates and is well known, understood, and supported, so there was no reason to come up with another language. Yet, the ease with which we are able to express KCIs stems from the abstraction we made on the process execution data by using the so called process instance tables.

4.3 Compliance Governance Dashboard

Finally, KCIs are rendered to the compliance experts by means of a CGD, such as the one depicted in Figure 3 [1]. The CGD features a graphical representation of KCIs and serves as start point for further root-cause analysis. More specifically, the CGD creates an awareness of possible violations and concentrates the most important information to be evaluated at-a-glance. The interactive table (at the front in Figure 3) provides a drill-down and roll-up mechanism for the compliance status, for example, for the different drug dispensation locations controlled by the hospital (i.e., clinics, laboratories, dispensaries), according to two main analysis perspectives (compliance performance vs. process performance), down to the individual event level (e.g., the list of incomplete records associated to a drug (background of Figure 3).

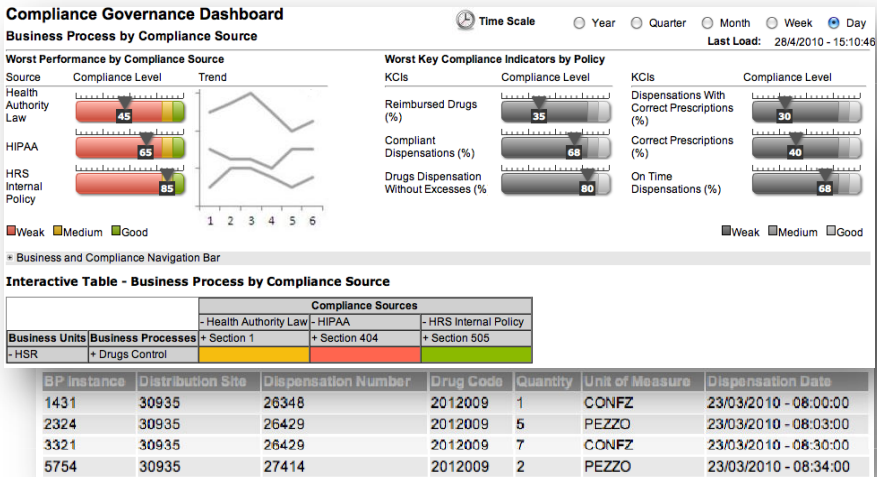


Fig. 3. CGD with KCIs and the interactive table for drill-down and roll-up [1]

5 Analyzing Non-compliance

While *checking* the compliance of business process instances means determining whether the process instances are compliant or not at the individual event trace level,

analyzing non-compliance of business process executions, i.e., understanding and explaining the underlying reasons of non-compliance, needs to be performed over a set of traces in order to be able to derive meaningful knowledge that can be used to improve processes for future executions.

Incidentally, labeling event traces as compliant or non-compliant, which is the main goal of compliance checking, is very similar to *classifying* data tuples, a data mining practice that is well-studied in literature [20]. There are several algorithms that can help in performing this analysis, among which we choose decision trees, as they are good for knowledge discovery where neither complex settings nor assumptions are required [20], and they are easy to interpret and analyze. In this section, we discuss how we address the issue of compliance analysis through decision trees, going from data preparation to the actual building and interpretation of the decision tree.

5.1 Preparing the Analysis

In Section 4.1, we introduced our DW model, which constitutes the basis for our CGD and the root-cause analysis. Preparing the analysis therefore means selecting which data, out of the huge amount of events stored in the DW, are suitable for identifying root-causes for non-compliance. In the same section, we also introduced the idea of having process instance tables, one per process, in which we store those process parameters that are used for computing indicators. Recall that each tuple in a process instance table represents a particular instantiation of the process under consideration and that each instance comes with its compliance label. Now, considering that we are interested in analyzing non-compliance problems for process instances, it is interesting to note that the process instance tables initially conceived for the computation of indicators also contain the data we are searching for. In fact, by defining a set of indicators for each process (and the events and data attributes that are necessary to compute them), the compliance expert implicitly performs a pre-selection of the data that are most likely to be related with compliance issues. The availability of the compliance label for each instance indicates that the best choice for the root-cause analysis is to use the process instance tables to feed the decision tree mining algorithm, as their data naturally fits the typical input format of these kinds of algorithms.

For instance, considering again the process instance table shown in Table 1, one way of building the training tuples for the decision tree is to use the *Compliant* column as the *class attribute* (leaf nodes) for the decision tree, while *ErrPerData* and *ErrCompData* can be used as the attributes on which the algorithm defines the split points (for internal nodes). This way, the training tuples can be represented as

$$\langle \text{ErrPerData}, \text{ErrCompData}, \text{Compliant} \rangle$$

The set of training tuples can be easily obtained through trivial SQL queries, and the retrieved result set can be used directly to feed the decision tree algorithm. Note that, as in the case of the specification and computation of the KCIs, the task of building the training tuples is greatly facilitated by the abstraction provided by the process instance tables.

5.2 Understanding Key Factors

The algorithm we use in our prototype implementation for building decision trees extends the C4.5 algorithm to handle uncertain data [21]. In this paper we do not

discuss the uncertainty aspect in mining data. However, our prototypes are equipped to handle uncertainty in the event logs we use for analyzing business process executions (for details on how uncertainty in event logs can be handled, see [6]). Instead, here we focus more on the aspect of discovering and understanding the key factors that affect the compliance of business executions.

As in any decision tree, the internal nodes contain the criteria used for classifying tuples. The leaf nodes, instead, contain the classes to which tuples are classified. For instance, if we choose the *Compliant* column of Table 1 as the class attribute, we will obtain a decision tree where the leaf nodes contain the *compliance outcomes* for the paths drawn from the root of the tree. However, nothing prevents us from choosing any other parameter of the process instance table as the class attribute when searching for the root-causes of non-compliant process executions.

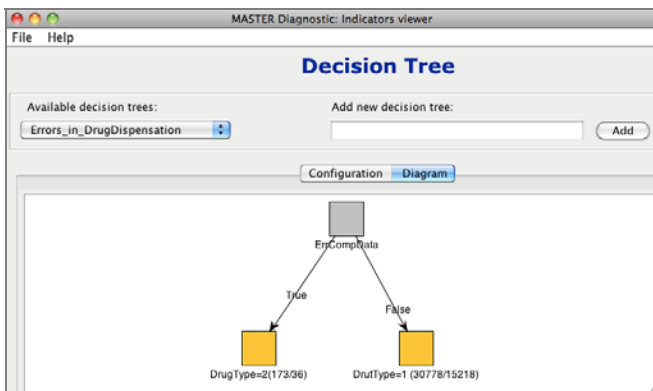


Fig. 4. Decision tree computed over non-compliant instances of the drug dispensation process

For instance, as part of the validation of this approach, we performed experiments on a dataset of more than 30000 drug dispensations performed between January and April of 2009 in the hospital described in the scenario (Section 2). To this end, a process instance table with around 25 relevant parameters was built for the drug dispensation process, among which the parameters shown in Table 1 were included. Since the dependence of the *Compliance* column on the *ErrPerData* and *ErrCompData* columns was fairly obvious (but still, proven with our tools), we narrowed our analysis by considering only those process instances that were not compliant. After exploring some combinations of parameters, we found out that there was a relation between the *ErrCompData* and *DrugType* parameters. More precisely, we found that 393 drugs dispensations out of around 30000 had some error, among which 173 had errors of the type *ErrCompData* and 220 errors of the type *ErrPerData*. While the decision tree was not able to tell us anything that was really significant about errors of the type *ErrPerData*, it was able to find something useful for the errors of the type *ErrCompData*, as shown in Figure 4. More precisely, the decision tree discovered that 137 out of 173 (79%) erroneous process instances corresponded to drugs of the type 2 (*DrugType*=2), which are drugs for ambulatory usage, while the rest (21%) corresponded to drugs of the type 6, 9 and 11.

Since the *ErrCompData* refers to error in the dispensation data (such as the drug code, quantity and unitary price), this may be an indication that, for example, this type of drugs is dispensed at ease, and thus, a better monitoring or compliance enforcement need to be carried out on the controls related to this compliance requirement.

5.3 Predicting Compliance States

While decision trees are generally perceived as simple classifiers, we however use them rather for discovering and understanding better the root-causes of undesirable behaviors. Furthermore, we advocate the use of decision trees also for predicting the potential outcomes of process instances that are still running. In fact, each decision point in a tree corresponds to an event (or better to an attribute of an event). So, if during process execution an event that corresponds to a decision point is generated, this allows performing predictions on the likely outcome (in terms of compliance) of the process instance: it suffices to inspect the path in the tree determined by the registered event to identify the instances' likely compliance label.

Thus, in the case of predictions of non-compliant behaviors, enforcement actions can be enacted in order to align process executions, whenever possible, to the corresponding compliance requirements. This is particularly useful in cases when the process has several tasks and long running times that span, e.g., over several hours. Also, the prediction is particularly useful in the case compliance is enforced manually, because it allows the compliance expert to better focus his effort on those process instances that are likely to be non-compliant, leaving out compliance ones.

6 Related Work

The major part of compliance management approaches focuses on the *business process modeling* aspect at design time [7-9]. Typically, they are based on formal languages to express compliance requirements (e.g., Business Property Specification Language, Linear Temporal Logic) and simulations to prevent errors at runtime (e.g., finite state machine, Petri nets). In this context, just few approaches address *compliance monitoring at runtime*. For instance, Trinh et. al. [10] monitor time constraints during the execution of process activities, using UML Timing Diagrams to specify constraints and Aspect Oriented Programming to control executions. Chung et. al. [11] check if the user-defined process is compliant to pre-defined ontology and a specific model, in which compliance requirements are described. An IBM research group [12] advocates the use of the REALM (Regulations Expressed As Logical Models) metamodel to define temporal compliance rules and the Active Correlation Technology to check them. That way, it can detect duplicate events or compute a user-definable function, which checks whether a function exceeds some threshold.

Concurrently, commercial *Business Activity Monitoring* (BAM) solutions have been developed to support compliance management (e.g., IBM Tivoli, HP Business Availability Center, Nimbus, Oracle Business Activity Monitoring). Although, such tools still do not have the capability to process and interpret *generic events* (e.g., user-defined business or compliance-related events). They only support the definition of thresholds for parameters or SLAs to be monitored. Also, the ability to *compare* monitored business process executions or, more in general, business patterns with expected execution behaviors is not supported.

Regarding reporting on compliance and KCIs, few works address this aspect and they do it partially. For example, [18] studies the representation of data through visual languages for risk and compliance management. In [19], the authors purpose a model-driven fashion approach to report on business performance and design dashboards.

To the best of our knowledge, no *mining approaches* have been specifically proposed to understand the root-cause of the compliance violations. However, few related approaches for the mining of business processes are in place [3-5][14-16]. Similar to our solution, they adopted log files and a consolidated warehouse containing business and process historical data, from where data subsets are extracted and used as input to mining algorithms in order to predict or understand the origin of undesired business process execution behaviors.

Finally, we can conclude that Compas and Master have been done significant contributions in all the fields mentioned in this section, since they provide solutions to manage, monitor and report on compliance based on generic events. For instance, [2][13] provide approaches to the management of the compliance monitoring at runtime, [17] states how to compute uncertain key indicators from uncertain data, [1] presents CGD to report on compliance and this paper presents root-cause analysis based on data mining techniques to understand non-compliant business processes.

7 Conclusions and Future Work

In this paper, we leverage on automated compliance checking techniques and complement them with a tactical perspective that targets compliance experts, which are accountable for assuring and improving compliance. We assist them by automating the analysis of the huge amount of data that is produced during process execution and specifically provide (i) a reporting dashboard with KCIs and KPIs to assess the state of compliance, (ii) a root-cause analysis technique to understand non-compliance. Our experiments with real data from a major Italian hospital show that the developed dashboard is effective in highlighting encountered problems and that the proposed abstractions and selection of data indeed allow us to identify also unexpected causes for non-compliant situations out of a large amounts of data.

It is important to note that, although in this paper we focused on the case of compliance, the ideas and solutions we propose are of general nature and can, for instance, easily be applied to the computation and analysis of KPIs. Similarly, we are not limited to process engine event as only source of information; events may also stem from web services, human task managers, or similar – if suitably instrumented.

Acknowledgements. This work was supported by funds from the European Commission (contract N° 216917 / MASTER and contract N° 215175 / COMPAS).

References

1. Silveira, P., Rodríguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, C.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: NFPSLAM-SOC 2009. Springer, Stockholm (2009)

2. Daniel, F., D'Andrea, V., Strauch, S., Schumm, D., Leymann, F., Mulo, E., Zdun, U., Dustdar, S., Sebahi, S., de Marchi, F., Hacid, M.: Business Compliance Governance in Service-Oriented Architectures. In: AINA 2009. IEEE Press, Los Alamitos (2009)
3. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in Business Processes. BETA Working Paper Series, WP 164, Eindhoven University of Technology, Eindhoven (2006)
4. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business Process Intelligence. *Computers in Industry Journal* 53(3), 321–343 (2004)
5. Seol, H., Choi, J., Park, G., Park, Y.: A framework for benchmarking service process using data envelopment analysis and decision tree. *ESA* 32, 432–440 (2007)
6. Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Toward Uncertain Business Intelligence: the Case of Key Indicators. *IEEE Internet Computing* 14(4) (2010)
7. Liu, Y., Mueller, S., Xu, K.: A Static Compliance Checking Framework for Business Process Models. *IBM Systems Journal* 46(2), 335–361 (2007)
8. Lu, R., Sadiq, S., Governatori, G.: Compliance Aware Business Process Design. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) *BPM Workshops 2007*. LNCS, vol. 4928, pp. 120–131. Springer, Heidelberg (2008)
9. Awad, A., Weske, M.: Visualization of Compliance Violation in Business Process Models. In: *BPI 2009*. LNCS, vol. 43, pp. 182–193. Springer, Heidelberg (2009)
10. Trinh, T., Do, T., Truong, N., Nguyen, V.: Checking the Compliance of Timing Constraints in Software Applications. In: *KSE 2009*, pp. 220–225 (2009)
11. Chung, P., Cheung, L., Machin, C.: Compliance Flow - Managing the compliance of dynamic and complex processes. *Knowledge-Based Systems* 21(4), 332–354 (2008)
12. Giblin, C., Müller, S., Pfitzmann, B.: From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. *IBM Research Report RZ 3662* (2006)
13. Mulo, E., Zdun, U., Dustdar, S.: Monitoring Web Service Event Trails for Business Compliance. In: *SOCA 2009*. IEEE Computer Society Press, Los Alamitos (2009)
14. Grigori, D., Casati, F., Dayal, U., Shan, M.: Improving business process quality through exception understanding, prediction, and prevention. In: *VLDB 2001*, pp. 159–168 (2001)
15. Apte, C., Bibelnicks, E., Natarajan, R., Pednault, E., Tipu, F., Campbell, D., Nelson, B.: Segmentation-Based Modeling for Advanced Targeted Marketing. In: *Knowledge Discovery in Databases and Data Mining*, pp. 408–413. ACM, New York (2001)
16. Bibelnicks, E., Campbell, D.: Mail Stream Streamlining. *Catalog Age* 17(12), 118–120 (2000)
17. Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Computing Uncertain Key Indicators From Uncertain Data. In: *ICIQ 2009* (2009)
18. Bellamy, R., Erickson, T., Fuller, B., Kellogg, W., Rosenbaum, R., Thomas, J., Wolf, T.: Seeing is believing: Designing visualizations for managing risk and compliance. *IBM Systems Journal* 46(2), 205–218 (2007)
19. Chowdhary, P., Palpanas, T., Pinel, F., Chen, S., Wu, F.: Model-driven Dashboards for Business Performance Reporting. In: *EDOC 2006*, pp. 374–386 (2006)
20. Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco (2006)
21. Tsang, S., Kao, B., Yip, K., Ho, W., Lee, S.: Decision Trees for Uncertain Data. In: *ICDE 2009*, pp. 441–444. IEEE, Los Alamitos (2009)

Trade-off between Complexity of Structured Tagging and Effectiveness of Web Service Retrieval

Maciej Gawinecki¹, Giacomo Cabri¹, Maria Ganzha^{2,3}, and Marcin Paprzycki²

¹ Department of Information Engineering,
University of Modena and Reggio Emilia, Italy
`name.surname@unimore.it`

² Systems Research Institute, Polish Academy of Sciences, Poland
`name.surname@ibspan.pl`

³ Institute of Informatics, University of Gdańsk, Poland

Abstract. Searching for services often starts from the exploration of the service space. Community generated tags can support such exploration. Researchers attracted by the community-available “free manpower” proposed more complex *tagging models*. Those models tag specific parts of the Web service definition: single operations, their inputs and outputs. However, there is no evidence whether the annotation effort is justified by the performance improvement of retrieval (based on those annotations). In this paper we apply similarity-based search to estimate the trade-off between retrieval effort and the annotation effort. Our experiments shows that exploiting similarity metrics for service ranking can compensate missing information and thus be more realistic solution than passing burden of tagging about more aspects on the community.

1 Introduction

Software developers are always short of time and thus need to find a relevant service quickly. A catalog in which services are described accurately, completely and in a concise way by some kind of metadata, may save developers’ retrieval effort significantly. No longer developers need to read complex documentation and to test each service candidate. However, building such a catalog requires service brokers to spend their limited funds on providing metadata for each service. It is then important for a service broker to choose such type of metadata that offers good trade-off between retrieval effectiveness and effort, they require to be defined.

Collaborative tagging provides a good trade-off because it moves the burden of describing service behavior to the community [1]. More complex tagging models encourage community to provide metadata also on parts of the Web service definition: single operations, and their input and output [2,3,4,5,6]. Adding more aspects to service metadata allows to find a service not only based on what it does, but also what information it consumes and what information it returns. Adding a structure allows to distinguish between those aspects explicitly during

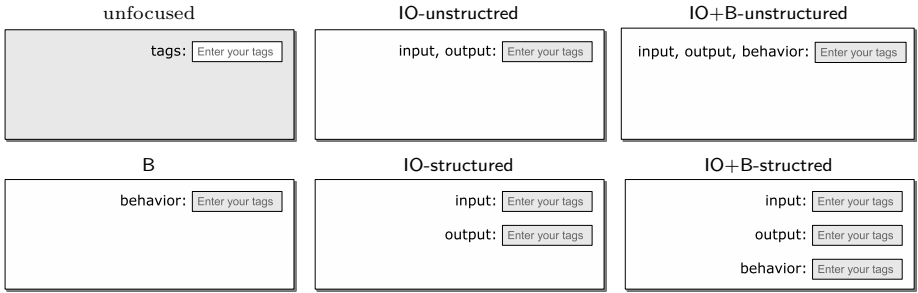


Fig. 1. User interface mock-ups for tagging models: traditional one (unfocused, not evaluated)—open to any aspect of a service—and 5 evaluated tagging models suggesting aspects to describe (Behavior, Input and Output) with and without separation

retrieval. The increasing complexity of the tagging models affects the design of user interfaces for tagging as shown in Figure 1. Unlike in traditional models, a user during tagging is encouraged to focus on selected aspects of a service. Depending on the model she is asked to describe one or more of them in the same or separated fields. Therefore, the user may ask whether effort spent on annotating complex structures is justified by the performance improvement. If not then the community may lose motivation to contribute to the catalog. However, few works on service retrieval using (structured) tags evaluate proposed approaches [3, 5], and none estimates the trade-off between the increased tagging complexity, and the retrieval effectiveness.

In this paper we estimate such trade-off to suggest a service broker which tagging model to choose when building the catalog of services. Particularly, we evaluate *whether tagging only about service behavior saves retrieval effort substantially enough or it is worth to encourage the community to describe services with other aspects?* And *whether these aspects should be separated by the user during tagging?* We limit our analysis to single-operation services. We estimate retrieval effort using similarity-based search that operates on service metadata in a uniform way across different tagging models. We estimate annotation effort for each model based on the number of aspects it describes, number of annotations and whether they are structured or not. By estimating those efforts on the corpus of 50 real services from geographic-domain annotated by 27 users we provide suggestions for choosing the right model. Main contributions of this paper are: (1) methodology for estimating annotation effort and retrieval effort for different tagging models; (2) suggestions for selecting the right model depending on the nature of underlying collection of services.

The paper is organized as follows. Section 2 discusses benefits and limitations of tagging for services; and also user incentives to provide tags. In Section 3 we identify five tagging models from the literature. In Section 4 and 5 we introduce measures for estimating amount of annotation effort and retrieval effort, respectively. Those measures are used in Section 6 to experimentally find a trade-off

between both types of effort. Based on our evaluation we provide suggestions for selecting a tagging model in Section 7.

2 Service Tagging Background

2.1 Benefits and Limitations of Tagging for Service Retrieval

In service retrieval the community tags play a number of roles: (i) they provide a network of links to browse services [1], (ii) they provide metadata for ranking-based search [4,3,5] and (iii) they facilitate quick understanding about a service, without reading a complex documentation. Application of collaborative tagging to services has been originally proposed by [1] as an alternative to authoritatively defined taxonomies that are tedious to browse and understand. The advantage of tags is they can capture aspects of a service that are important for the community but has not been encoded neither in the WSDL definitions from service providers, nor in the authoritative classification; moreover, they use vocabulary more relevant and intuitive to a developer [2]. Therefore, the distance between what the developer wants and how it is described in the repository can be minimized [7]. However, tags cannot be used alone for effective retrieval, but only as a complementary mechanism to traditional classification schemes like taxonomies and controlled vocabulary [7]. This goes along with observation of application of tags in other domains, like book search (e.g. Amazon). The possible answer to that problem, being of our concern in this paper, is to *focus* user tagging on particular aspects of a service. [3] proposed to tag input and output of a service separately to facilitate search based on interface matchmaking. In our previous work [5] we proposed to annotate also behavior of services, to match services that are functionally equivalent, but have incompatible interfaces. This approach has provided retrieval performance competitive to solutions based on Semantic Web Services (SWS), presumably thanks to fine-grained aspects of a service, that are easier to express by free-form tags than in formal languages for SWS.

2.2 User Motivations for Service Tagging

Since collaborative tagging relies on voluntary participation, a service broker has no control on which service will be annotated, how and when. Hence, important reasons must attract a critical mass of people to annotate services. In general, reasons to contribute in online communities differ from one community to another. In the following we focus on tagging incentives for communities of two real portals for Web service discovery and tagging. A reader interested in tagging motivations in general and incentives to contribute to collaborative online projects (like open source software or Wikipedia) is referred to works of [8,9,10].

For SeekDa.com Web service search engine the main target group consists of professional software developers who look for web services when they are in a hurry to finish a project and do not have time to develop their own application or address open source solutions [11]. They usually contribute with content (comments, ratings, tags, additional descriptions) during search, mainly to services

they found useful in their applications. Tags are provided only if existing ones have been found useful for service retrieval. In this way a user organizes services she found useful; she does not mind to share them with others.

ProgrammableWeb.com online catalog addresses needs of end-user programmers building mash-ups. Searching for a service is the most time-consuming part of application development in their case. It plays also inspirational role steering their projects in particular direction, because from existing mash-ups they can learn what applications can be built and from what services [12]. A user can tag only a service she submits to the community. Hence, tagging is a way to inform the community about own contribution.

Both mentioned portals do not limit aspects on which user may tag a service. Suggesting aspects, and particularly in clearly structured form, provides guidance to a tagger but it also limits types of tags she may enter [13]. It is an open question how such modification of tagging process affects users motivation to tag services.

3 Tagging Models for Web Services

Selecting the right tagging model boils down to choosing a user interface for tagging and deciding how annotations are stored. The interface encourages a user to tag on suggested aspects. It also defines whether the tags on different aspects can be entered into separate fields or into a single one. Annotations from taggers in traditional unfocused tagging model are stored within a folksonomy of services [1]:

Definition 1. A folksonomy $\mathbf{F} \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$ is a hypergraph $\mathbf{G}(\mathbf{F}) = (\mathbf{V}, \mathbf{E})$ with

- vertices $\mathbf{V} = \mathcal{A} \cup \mathcal{T} \cup \mathcal{S}$, where \mathcal{A} is the set of actors (users and the system), \mathcal{T} — the set of tags and \mathcal{S} — the set of services.
- hyperedges $\mathbf{E} = \{(\mathbf{a}, \mathbf{t}, \mathbf{s}) \mid (\mathbf{a}, \mathbf{t}, \mathbf{s}) \in \mathbf{F}\}$ connecting an actor \mathbf{a} who tagged a service \mathbf{s} with the tag \mathbf{t} .

Here, a single hyperedge is called an annotation. A single folksonomy may capture different aspects of a service, but does not allow to distinguish between them. To address this limitation, tags entered in different fields are put into separate folksonomies. Therefore, tagging models identified in the literature differ in the user interface (Figure 1) and number of folksonomies to store data:

- **Only the behavior aspect (B):** behavior of a service is described, as originally foreseen in [1]; only a single folksonomy F_b is necessary to model it;
- **Interface unstructured (IO-unstructured):** only input/output are described; only a single folksonomy F_{io} is necessary to model it;
- **Interface structured (IO-structured):** as previous, but aspects are explicitly structured, as defined in [3]; two folksonomies F_i and F_o for input and output, respectively, are modeled;

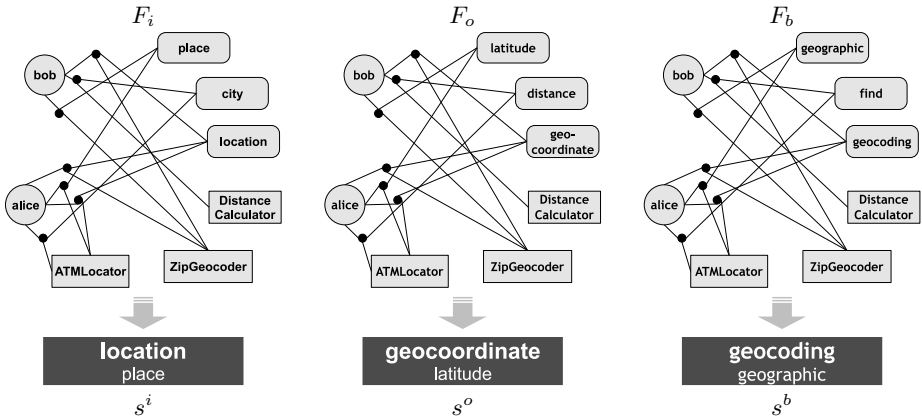


Fig. 2. Example folksonomies (F_i, F_o, F_b) for three service aspects in the B+IO-structured annotation model and corresponding descriptions (tag clouds) of the ZipGeocoder service extracted from them (s^i, s^o, s^b). A circle depicts a tagging user, rectangle—a tagged service and rounded box—a tag used. Each black spot defines a single annotation: which *user* annotated what *service* with what *tag*. A tag cloud is a visual depiction of user-generated tags for a given service part. The bigger the tag, the more users provided it.

- **All aspects without structure (B+IO-unstructured):** community describes behavior and input/output, but does not structure this information. Hence, all aspects can be stored in a single folksonomy F_{b+io} .
- **All aspects structured (B+IO-structured):** as previous, but aspects are explicitly structured to handle ambiguity between different types of tags (as in [5]); hence, three folksonomies F_i, F_o, F_b , are used for input, output and behavior of a service, respectively.

Figure 2 shows examples of folksonomies for B+IO-structured, the most complex model. For instance, F_b is defined by the following annotations: (bob, geographic, ZipGeocoder), (bob, geocoding, ZipGeocoder), (bob, find, DistanceCalculator), (alice, find, ATMLocator), (alice, location, ATMLocator), (alice, geographic, ATMLocator), (alice, geocoding, ZipGeocoder).

Obviously, the last model offers the most complex structure and covers all aspects of a service. The remaining models can be created by removing folksonomies that capture unused aspects and/or combining folksonomies to a single one. Such reduction always results in information loss.

4 Estimating Annotation Effort

To understand required user involvement for the given tagging model it is important to understand how much effort it requires from the community to annotate the whole collection of services in the catalog. The annotation process

for a single user means: (1) understanding what a service is about based on the provided information (documentation, interaction, monitored QoS, existing applications, other's people tags and comments), and (2) suggesting a combination of tags describing the service. Quantifying such effort is difficult because of many subjective aspects involved. Particularly, it varies heavily with the background knowledge and cognitive abilities of users, their familiarity with annotation tools, usability of such tools and of available information. This makes it difficult to capture the cost in a general way. Centralized annotation offers two measures that allow to capture objective aspects of the annotation effort: *absolute*, defined as a number of man-hours spent on a complete annotation of a whole collection of objects [14] and *relative*, defined as a fraction of possible annotations to collect [15]. Man-hours are not a good measure of effort of collaborative tagging, as it is almost impossible to measure them in the open (i.e. uncontrolled) distributed environment, with ad-hoc voluntary participants. The relative measure will not work because there is no criterion of completeness, as tags are added continuously [16]. Therefore, we propose to estimate annotation effort for each model based on: the number of aspects it describes, number of annotations its instance contains and whether they are structured or not. We assume the annotation required more effort from the community, if more aspects have been captured and more annotations have been provided. For simplicity, we assume that this is the same amount of effort for a human to provide 10 annotations as for ten people to provide a single annotation each.

5 Estimating Retrieval Effort

In ranking-based retrieval the less effective a solution is, the more results in the ranking a searcher needs to scan to find a perfect one. In other words, she needs to spend more retrieval effort. The normalized Discount Cumulative Gain (*nDCG*) [17] curve models retrieval effectiveness and allows to estimate amount of retrieval effort. Let us define first G_i , a gain value that the i -th returned service obtains for being relevant. We define

$$DCG_i = \begin{cases} G_1, & i = 1 \\ DCG_{i-1} + G_i / \log_2(i + 1), & i \geq 2 \end{cases} \quad (1)$$

The Discount Cumulative Gain (*DCG*) rewards relevant answers in the top of the ranking more than those in the bottom of the ranking. Calculated DCG_i is then normalized by the ideal possible DCG_i to make the comparison between different approaches possible. When plotting the *nDCG* curve, the X-axis represents a rank, and the Y-axis represents a *nDCG* value for a given rank. An ideal similarity approach has a horizontal curve with a high *nDCG* value. Hence, the size of the area above the actual *nDCG* curve tells how much more effort a user needs to spend on retrieval with respect to scanning the ideal ranking. The discount factor of $\log_2(i + 1)$ is relatively high, to model an impatient user who gets bored when she cannot find a relevant answer in the top of the ranking.

Retrieval effectiveness depends on the following variables: a user query formulation strategy, a ranking algorithm, an tagging model and its content. Since we are interested in attributing estimation of retrieval effort to the last two variables, we need to eliminate impact of the first two.

5.1 Eliminating Impact of Query Formulation Strategy

Users differs in how they formulate the same information need with query keywords and thus the choice of a user may impact retrieval effectiveness. Similarity-based search does not has this drawback, because a request is expressed as a service, for which services of similar functionality have to be returned. Similarity-based search ranks results with respect to they similarity to the request. Existing methods in the literature operates on different metadata describing a service: WSDL definition (e.g. [18]), keywords extracted from WSDL definitions (e.g. [19]), or semantic annotations (e.g. [20]). In our case metadata are defined by community annotations.

We define similarity measure for the tagging model, where all annotation aspects are explicitly structured (B+IO-structured). By s^i , s^o , s^b we denote aspects of the tagged service s , concerning input, output and behavior, respectively. They can be extracted automatically from the corresponding folksonomies F_i , F_o , F_b . For instance, Figure 2 shows descriptions extracted for the ZipGeocoder service. By λ we denote similarity metric for the tagged descriptions from a single folksonomy. Since different aspects are described in different folksonomies, we can measure similarities for each aspect of two services s_1 and s_2 separately and then combine similarities together:

$$\gamma^{b+io-structured}(s_1, s_2) = \lambda(s_1^i, s_2^i) + \lambda(s_1^o, s_2^o) + \lambda(s_1^b, s_2^b) \quad (2)$$

For unstructured annotations models, operating on a single folksonomy, we reduce similarity measure between s_1 and s_2 to $\lambda(s_1, s_2)$. For the IO-structured tagging model, using two folksonomies, we define it as: $\gamma^{io-structured}(s_1, s_2) = \lambda(s_1^i, s_2^i) + \lambda(s_1^o, s_2^o)$.

5.2 Limiting Impact of Ranking Algorithm

We want to define the similarity measure λ for any two resources (based on the information about them) in a *single* folksonomy. Depending on whether the tagging model is structured or not, a *resource* can be a single service (unstructured models) or a single aspect of a service (structured models). To avoid a bias towards a particular ranking algorithm we selected a number of similarity measures λ proposed by Markines et al. [21] for tagged resources. We selected two that provide relatively good accuracy, while are still simple to analyze: the projection-aggregated overlap similarity, and the distribution-aggregated cosine similarity. These measures differ in how they aggregate information about a resource from a folksonomy hypergraph, and how they compute similarity based on the aggregated information. *Projection-aggregated overlap similarity* considers two resources to be similar, if they share many tags. *Distribution-aggregated cosine similarity* considers not only the overlap in tags, but also how they are distributed among users and

other services. There are three variants of cosine similarity. A *term frequency* (TF) variant considers two resources similar, if tags they share have been proposed by many users. An *inverse document frequency* (IDF) variant considers two resources similar, if the tags they share are uncommon in the collection. A *combination* of those two (TF/IDF) considers two resources similar, if the tags they share have been proposed both: (a) for them but not for many other resources and (b) by many users. Only the TF variant has been evaluated in [21].

6 Experimental Evaluation

The goal of the preliminary evaluation was to check which tagging model offers the best trade-off between the annotation effort and the retrieval effort.

6.1 Test Data

Our test data consisted of the collection of services and their annotations. For the first part we used the Jena Geography Dataset (JGD50) [22] containing WSDL definitions of 50 real data-centric single-operation services from the geography domain. It is more representative for a class of domain-specific catalogs of homogeneous services (e.g. GeoNames.org) rather than for class of catalogs with a large number of diverse services (e.g. SeekDa.com). Moreover, for a single user the homogeneity of the collection means that she must provide more tags for a single service to differentiate it better from other similar services. In the simplest case, if each service was from a different functionality domain, then it would be sufficient to tag it with only its domain name.

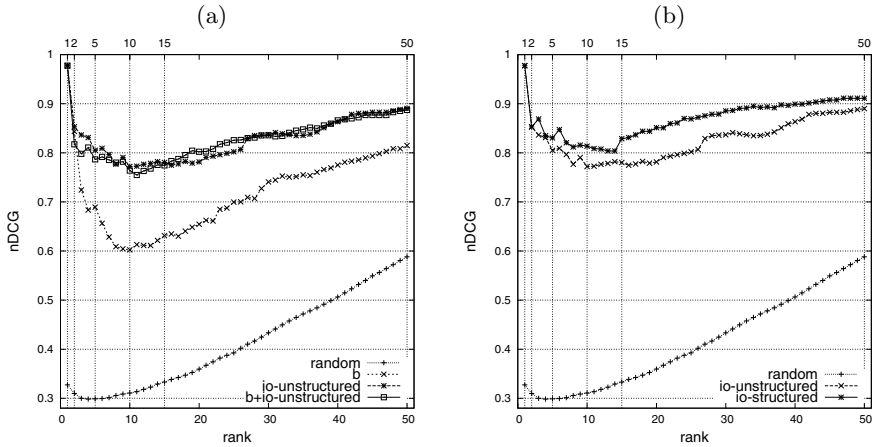
The data set contains also 9 service requests. We filtered those 7 of them, for which the set of relevant services (with respect to relevance judgments) included a large fraction of services of incompatible interfaces but still with equal or approximate functionality. Since original service requests were expressed in a natural language, and similarity search requires a service as a request, we replaced each original service request in the JGD50 with the most relevant service offer (based on the corresponding relevance judgments). To estimate relevance of results we used graded relevance judgments from the dataset. They fit well for similarity-based search where even partial match in functionality, can be somehow relevant to the user. We used the relevance setting that emphasized functional equivalence over interface compatibility, because it models well users that are interested in services providing similar functionality, even if they are not interface compatible; moreover, such users are not interested in interface compatible services that do not provide similar functionality.

At the time of the evaluation, there was no industrial corpus of structured tags for services placed in the test collection. Note that lack of such corpus is the most challenging obstacle in evaluation of retrieval based on collaborative tagging of Web services [1,2,4,6,3], and software components in general [7]. For the B+IO-structured model we reused the WSColab [4] artificial corpus, which contains 4008

¹ <http://fusion.cs.uni-jena.de/professur/jgdeval/JGD50-WSColab-Services.zip>

Table 1. Numbers of annotations for evaluated instances of tagging models

tagging model	#total	#input	#output	#behaviour
B	1496	0	0	1496
IO-*	2512	1437	1075	0
B+IO-*	4008	1437	1075	1496

**Fig. 3.** Effectiveness of overlap similarity when adding more: (a) aspects to unstructured models, (b) structure to IO-based model. Averaged over all queries.

annotations for JGD50 services. It consists of both annotations bootstrapped automatically (32%) and provided by the community (68%). The latter were collected in the open (non-laboratory) environment through the collaborative tagging portal from 27 users through the collaborative tagging portal based on WSDL documentation (see, also [23], for more details). For the remaining tagging models the tag corpora were created from the B+IO-structured model as described in Section 3. Numbers of resulting annotations for creating corpora are reported in Table 1.

6.2 Experimental Results

We compared similarity search results for the five tagging models identified in Section 3. Each model was evaluated using both similarity metrics: the overlap, and the cosine similarity (all variants). This resulted in 20 experimental cases. To put evaluation results into the context we plotted a curve for a random similarity search. The curve has been generated by averaging effectiveness over 50 random permutations of ranking. Generating another 50 permutations has shown that this is a sufficiently large base for stable results in this test collection. The curve is relatively high, confirming that services in the test collections are similar to each other and thus challenging to distinguish for taggers and a retrieval algorithm.

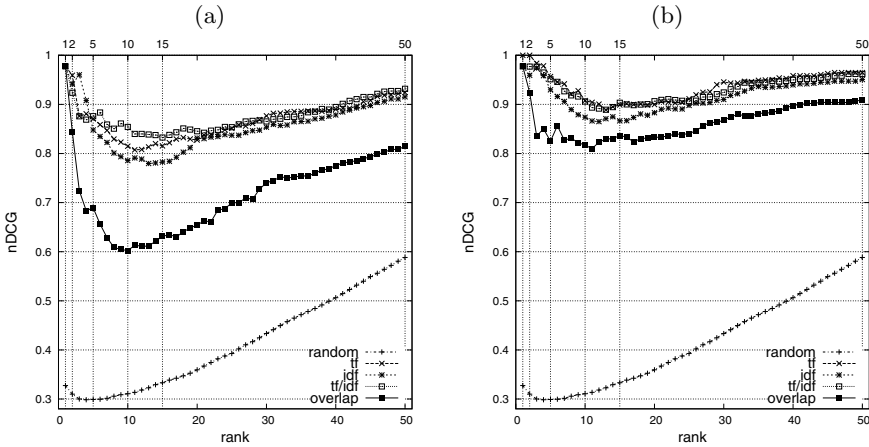


Fig. 4. Effectiveness of varying similarity metrics for: (a) the B model and (b) the B+IO-structured model. Averaged over all queries.

Firstly, we analyzed whether adding more aspects limits retrieval effort. Figure 3a shows nDCG curves for unstructured models with a different number of aspects. It demonstrates that, for the simplest similarity metric, IO-unstructured allows to save about 11% more retrieval effort than B, but combining those aspects together into IO+B-unstructured does not significantly save retrieval effort with respect to IO-unstructured: much below 1% (evaluation has shown that structured models behave in an analogous way). This is surprising because the large fraction of services in the corpus is functionally equivalent but not necessary interface compatible and adding behaviour tags should help improve effectiveness. A possible reason is that behaviour tags are more redundant: adding behaviour aspect introduces less than 14 new tags per a service (users often describe output of data-centric services as their behaviour), while adding input and output aspects to B introduces more than 20 new tags.

Secondly, we investigated whether adding structure limits retrieval effort. Adding a structure allows to disambiguate between tags describing different aspects, but may cause errors caused by structure gap. This is because the structure itself can be ambiguous for an annotator (we recall the example where output tags are found among the behavior tags). Figure 3b shows nDCG curves for IO-* models. It demonstrates that adding a structure saves only 4% of retrieval effort for the simplest similarity metric (evaluation has shown that IO+B-* models behave in the same way: about 4% of retrieval effort was saved). A possible reason is that only 12% of tags describing interface between input and output overlapped (on average for a service), which means that remaining tags provided reasonably good disambiguation comparing to the structure.

Thirdly, we investigated whether previous observation can be generalized to different similarity metrics. Across all tagging models, the projection overlap required more retrieval effort than any variation of the cosine similarity. This is consistent with results of Markines et al. [21], though in our case the additional information

about tag distribution in the folksonomy appears to have larger impact on the effectiveness of retrieval. For instance, Figure 4a shows nDCG curves of varying similarity metrics for the B model. The possible interpretation of such a large impact is that the projection overlap, in opposition to cosine similarity, does not distinguish between minor (e.g. *miles*), and major features (e.g. *geocoding*) and thus may incorrectly detect two services as highly similar based only on a minor feature. Major features are presumably those which are represented by tags on which more people have agreed. We have found also that among variants of cosine similarity there is no dominating winner. Moreover, we observed that using cosine similarity for more complex tagging models does not have such significant impact on retrieval effort increase. Comparing changes in retrieval effort between the B model (Figure 4a) and the IO+B-structure model (Figure 4b) demonstrates that fact. The conclusion is that selection of the right similarity metric can compensate lack of some aspects and structure in the tagging model.

Fourthly, having selected competitive similarity metric (TF/IDF cosine similarity), we related retrieval effort to the number of annotations (see Table 1) for the worst (B) and the most effective (IO-B-structured) model. Surprisingly, even if IO-B-structured allowed to take only 6% retrieval effort less than B, it still required significantly more annotation effort: it used about 2.7 times more annotations and three service aspects instead of one.

7 Conclusions and Suggestions for Service Brokers

Choosing the right tagging model is important for a service broker relying on the participation of the community. The performed experiments show that, for the considered corpus of service, the tagging model describing only behavior offered the best trade-off between retrieval effort and annotation effort. The amount of retrieval effort saved by tagging on more aspects is not impressive, especially when related to the annotation effort. Therefore, for *similarity search* in general, it makes sense to add more aspects to the tagging interface incrementally, only if existing ones do not allow to match or disambiguate services in the service catalog. Furthermore, exploiting similarity metrics for service ranking can compensate missing information and thus be more realistic solution than passing burden of tagging about more aspects on the community. Finally, using structure for retrieval did not limited retrieval effort substantially as other tags could play similar disambiguation role. However, the structure can be still for important for searches focused on selected aspects or for a tagger who needs clear separation between aspects during tagging.

Acknowledgments. We thank to the members of FUSION group from Jena University for discussion and comments that helped strengthen the paper.

References

1. Meyer, H., Weske, M.: Light-Weight Semantic Service Annotations through Tagging. In: Dan, A., Lamersdorf, W. (eds.) ICSC 2006. LNCS, vol. 4294, pp. 465–470. Springer, Heidelberg (2006)

2. Fernández, A., Hayes, C., Loutas, N., Peristeras, V., Polleres, A., Tarabanis, K.A.: Closing the Service Discovery Gap by Collaborative Tagging and Clustering Techniques. In: SMRR (2008)
3. Bouillet, E., Feblowitz, M., Feng, H., Liu, Z., Ranganathan, A., Riabov, A.: A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition. In: IEEE SCC, pp. 389–396 (2008)
4. Chukmol, U., Benharkat, A.N., Amghar, Y.: Enhancing Web Service Discovery by Using Collaborative Tagging System. In: NWESP, pp. 54–59 (2008)
5. Gawinecki, M., Cabri, G., Paprzycki, M., Ganzha, M.: WSCOLAB: Structured Collaborative Tagging for Web Service Matchmaking. In: WEBIST (2010)
6. Silva-Lepe, I., Subramanian, R., Rouvellou, I., Mikalsen, T., Diament, J., Iyengar, A.: SOAliveService Catalog: A Simplified Approach to Describing, Discovering and Composing Situational Enterprise Services. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSC 2008. LNCS, vol. 5364, pp. 422–437. Springer, Heidelberg (2008)
7. Vanderlei, T.A., Durao, F.A., Martins, A.C., Garcia, V.C., Almeida, E.S., de L. Meira, S.R.: A cooperative classification mechanism for search and retrieval software components. In: SAC, pp. 866–871. ACM, New York (2007)
8. Kuznetsov, S.: Motivations of contributors to wikipedia. SIGCAS Comput. Soc. 36(2), 1 (2006)
9. Marlow, C., Naaman, M., Boyd, D., Davis, M.: HT06, tagging paper, taxonomy, Flickr, academic article, to read. In: Hypertext, pp. 31–40 (2006)
10. Preece, J.: Sociability and usability in online communities: determining and measuring success. *Behaviour & Information Technology* 20(5), 347–356 (2001)
11. Cuel, R., Oksana, T.: SeekDa Case. Technical report, Insemtives Project (2010)
12. Hartmann, B., Doorley, S., Klemmer, S.R.: Hacking, mashing, gluing: Understanding opportunistic design. *IEEE Pervasive Computing* 7(3), 46–54 (2008)
13. Bar-Ilan, J., Shoham, S., Idan, A., Miller, Y., Shachak, A.: Structured vs. unstructured tagging a case study. In: Proc. of the Collaborative Web Tagging Workshop (WWW 2006) (May 2006)
14. Küngas, P., Dumas, M.: Cost-Effective Semantic Annotation of XML Schemas and Web Service Interfaces. In: IEEE SCC, pp. 372–379 (2009)
15. Gomadam, K., Ranabahu, A., Ramaswamy, L., Sheth, A.P., Verma, K.: Mediatability: Estimating the Degree of Human Involvement in XML Schema Mediation. In: ICSC, pp. 394–401 (2008)
16. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. In: WWW, pp. 211–220 (2007)
17. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20(4), 422–446 (2002)
18. Wang, Y., Stroulia, E.: Semantic Structure Matching for Assessing Web-Service Similarity. In: Orłowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSC 2003. LNCS, vol. 2910, pp. 194–207. Springer, Heidelberg (2003)
19. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: VLDB (2004)
20. Hau, J., Lee, W., Darlington, J.: A Semantic Similarity Measure for Semantic Web Services. In: Web Service Semantics Workshop at WWW (2005)
21. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating Similarity Measures for Emergent Semantics of Social Tagging. In: WWW, pp. 641–641 (2009)
22. JGD: Jena Geography Dataset, <http://fusion.cs.uni-jena.de/professur/jgd>
23. Gawinecki, M., Cabri, G., Paprzycki, M., Ganzha, M.: Evaluation of Structured Collaborative Tagging for Web Service Matchmaking (2010), <http://mars.ing.unimo.it/gawinecki/pubs/wscolab-manuscript.pdf>

Aspect-Oriented Checkpointing Approach of Composed Web Services

Soumaya Marzouk, Afef Jmal Maâlej, and Mohamed Jmaiel

University of Sfax, ReDCAD Laboratory
National School of Engineers of Sfax
BP 1173, 3038 Sfax, Tunisia

{Soumaya.Marzouk,Afef.Jmal}@redcad.org, Mohamed.Jmaiel@enis.rnu.tn

Abstract. This paper proposes a solution for strong mobility of composed Web services. In fact, strong mobility enables a running BPEL process to be migrated from a host to another and to be resumed on the destination host starting from a previous execution state called also checkpoint which avoids the high overhead of restarting the composed Web service in case of interruption of the BPEL process. The proposed solution makes use of Aspect-Oriented Programming (AOP) in order to enable dynamic capture and recovery of a BPEL process state. This will enable the choose, at runtime, of the instant of the checkpoint and the technique for enacting it. Thus, the proposed approach may be used for self-healing and self-adaptivity of composed Web services acting in case of failure or QoS violation. An experimentation has been performed on a Travel agency case study deployed on the AO4BPEL engine. It shows the efficiency and the usability of our approach.

Keywords: Strong mobility, Aspect-Oriented Programming, checkpoint, Composed Web services, BPEL.

1 Introduction

Self-adaptation of distributed applications becomes a necessity especially in highly dynamic environments. In fact, in such environments application components failure or QoS violation may frequently occur due to a hosting node crash or performance degradation. Accordingly, there is a need for mechanisms enabling the adjustment of the application execution at runtime. One of these mechanisms is strong mobility. The latter enables a running application component to be migrated from a host to another and to be resumed on the destination host starting from an intermediary execution state called also checkpoint. This mechanism has been used in many contexts such as self-healing, self-optimization and load balancing [1,2] of distributed applications. Many work dealt with checkpointing of distributed applications and offer several solutions to perform it. In fact, checkpointing may be enacted periodically [3], at migration time [1], or at natural synchronization barriers [4]. In addition, they may employ different techniques such as coordinated [1], or uncoordinated [2] checkpointing. Most existing

solutions are based on fixed checkpointing policies. Especially, checkpointing instants and techniques are statically fixed at design time. However, it should be beneficial if checkpointing policies would consider the execution context parameters. For instance, coordinated checkpointing is rentable only if the failure frequency is low. Otherwise, employing this mechanism may block the application progress and uncoordinated checkpointing with message logging becomes more adequate [5]. As parameters, like the failure frequency, may change during the application execution, we believe that strong mobility solutions should allow the dynamic change of the employed checkpointing technique at runtime.

Accordingly, we aim to propose a flexible solution for strong mobility allowing the change of the checkpointing position and techniques at runtime. Thus, we propose an approach based on source code transformation and Aspect-Oriented Programming (AOP). The main idea of our solution consists in instrumenting the source code of the application in order to prepare its execution state saving, and then putting the checkpoint/recovery code in Aspects which may be deployed at runtime. Thus, we will enable not only the dynamic selection of the checkpointing position, but also an adaptive selection of the appropriate checkpointing technique by dynamically deploying the suitable aspect.

In this paper, we deal with composed services and more precisely with orchestration process¹ strong mobility. In fact, interruptions² affecting the orchestration process is very critical for composed services as all running instances may be interrupted. For instance, if the orchestration process hosting node crashes, the whole composed Web service execution will be interrupted. Such a situation may frequently occur especially in large scale environment like Grids and Clouds where Web service based applications become more and more used for managing and executing user applications. Most existing work dealing with this issue focus on unavailability or performance degradation of composite Web services and use often substitution for recovering the composite Web service [6,7] or the reorganization of the whole orchestration process [8]. However, when a failure causes the interruption of the whole orchestration process and even the crash of the orchestration engine, all running instances will be re-executed which has a large overhead on the application execution time since composed Web services are often long running. In such a case, strong mobility of orchestration processes can adjust the situation and avoid the high cost of restarting the composed Web service execution. We implemented our solution on the AO4BPEL engine [9], and we evaluate it in order to prove its efficiency. Thus, our approach may be used as self-healing mechanism in case of failure of the orchestration process hosting node by migrating all interrupted instances towards a new server. Besides, in case of QoS violation a subset of the running instances may be migrated to a new server in order to decrease the initial host load. As a result, the proposed solution has many strength against existing ones. First, it is **efficient** since it

¹ We use the orchestration process term to refer to the workflow process coordinating the execution of composite Web services. This process may be described using BPEL.

² Interruptions may be due to failures occurring at the infrastructure, the engine, or the application level and may affect all or a sub set of the running instances.

reduces the reconfiguration overhead by avoiding the application restart. Second, it is **flexible** as checkpointing instants and techniques can be dynamically set at runtime. Third, it is **modular** thanks to Aspects which allow changing the checkpoint policy without modifying the orchestration process code. Finally, it allows a **simple filtering** of the target instances concerned by mobility through the filtering power of aspect pointcuts. All these points make our solution applicable for many purposes like self-adaptation and load balancing.

This paper is organized as follows. The second section will be devoted to the presentation of related work. We detail, in the third section, our approach for orchestration processes checkpointing including source code transformation rules and checkpoint, recovery and mobility Aspects. The fourth section will describe our evaluation environment and will show the results of the experimentations of our approach. Finally, we conclude and present future works.

2 Related Work

In this section we focus first on checkpointing mechanism for distributed applications in general, then we present works dealing with self-adaptivity of Web services. In the context of distributed applications, many works dealt with strong mobility and employ different checkpointing techniques targeting different application types such as MPI and RMI based. However, those solutions fix the checkpointing mechanism at design time.

For instance, in [3] checkpointing is done periodically in a coordinated way. In this approach, the application resumption necessitates the rollback to the last captured checkpoint and ensures the resumption after migration starting from it. However, this solution becomes not rentable when deployed on an environment having a high failure frequency [5]. Other solutions employ also coordinated checkpointing but only at migration time [1]. They reduce the checkpointing overhead and ensure the resumption starting from the interruption point. These solutions are efficient only in stable environments where failures are predictable. Checkpointing at natural synchronization barriers is employed by [4] and offers low checkpointing overhead but fixed checkpointing interval. However, this checkpointing interval should be less than the Mean Time Between Failures MTBF in order to ensure the application progress. Other solutions employ uncoordinated checkpoint with message logging [2]. Such solutions are the most efficient when deployed in environments with high failure frequency [5] but have the most overhead in terms of bandwidth and storage capacity. Thus, every checkpointing mechanism is appropriate to a particular execution context. This context is frequently changed in large scale environment. Accordingly, fixing the appropriate checkpoint policy at design time may be very difficult in particular for long running applications. In this paper, we overcome this issue by offering a flexible solution allowing the dynamic change of the checkpointing policy at runtime. Thus, an execution manager may decide to change the checkpointing policy whenever the execution context changes. To this end, we make use of aspect-oriented approach to enable the dynamic deployment of the suitable checkpointing code at runtime.

For service oriented applications, most existing work dealing with self-healing and self-adaptation of composed Web services focus on unavailability or performance degradation of partners Web services and use often substitution for recovering orchestrated services [6,7] or reorganization of the whole orchestration process [8]. However, few solutions deal with orchestration service failure or QoS violation and rely on restarting the orchestration process. Such a solution suffers from a high reconfiguration overhead caused by re-executing all partner services invocations at reconfiguration time for all running instances of the orchestration process. The solution provided here differs from the existing ones by making use of strong mobility of orchestration processes instances. In fact, using strong mobility has the strength of offering the possibility of resuming the interrupted instances execution starting from the last captured checkpoint instead of re-executing them. However, we cannot neglect some tentative to make BPEL based orchestration process persistent. This is the case of the BPEL Server Engine of GlassFish ESB [10]. It offers persistence and recovery of BPEL instances in case of failure or interruption of the BPEL process. This solution is implemented at the engine level and can be activated or deactivated when needed. However, this solution has the drawback of being non configurable. In fact, it does not offer the possibility of forcing mobility in case of QoS violation. Moreover, there is no possibility to manage the checkpointing process by selecting the checkpointing interval or positions. In other words, this persistence/recovery solution is internal to the engine and does not offer the possibility to the user to configure it. In addition, there are problems with persisting flow activities. In fact, the adopted solution relies on persisting every flow branch independently. This may lead to non deterministic behavior at recovery time. Other problem with this solution is the fact that one is limited by the engine and can not change it while using the persistence/recovery property. Other solutions employ VM migration in order to migrate the whole service. Such solution may be employed to orchestration processes but does not enable the individual migration of orchestration process instances which may be very useful for load balancing for example. Our solution overcomes those issues by offering a fully configurable solution, implemented at the application level, where checkpoint interval, position and method may be selected and changed at runtime. Moreover, mobility may be forced even if the orchestration process does not crash. And the mobility method may be also selected at runtime. In addition, in our solution there is no risk of non deterministic execution when recovering flow activities since we always save a unique state for each instance. This state is constructed by synchronizing all flow branches leading to save a consistent checkpoint.

3 Aspect-Oriented Checkpointing of Composed Web Services

Strong mobility of orchestration processes consists in interrupting all (or a subset of) their executing instances, migrating them to another node, and then resuming their executions starting from their last saved checkpoints. Therefore,

(1) we employ source code transformation rules in order to prepare the capture of the execution state, (2) we add external services (WSIM and WSCM) in order to manage the checkpoint and the mobility processes, and (3) we include the checkpointing, recovery and mobility code within Aspects. The latter enables the selection, at runtime, of the instant and the technique for enacting the checkpoint by deploying the appropriate aspect whenever a checkpoint is required. In the next sections, we will present the architecture of our checkpointing solution, the source code transformation rules, and the code of designed Aspects. Here, we apply our approach for BPEL and specifically for the BPEL version 1.1 [11] since it is the language employed by the AO4BPEL engine [9]. However, our approach is generic and may be applied for any workflow description language.

3.1 Architecture for Aspect-Oriented Checkpointing of Composed Web Services

Ensuring the strong mobility of an orchestration process relies on adding mainly two management services called the Web Service Invocation Manager (WSIM) and the Web Service Checkpoint Manager (WSCM) (see Figure 1).

The Web Service Invocation Manager (WSIM) is a Web service deployed in an intermediary tier between the client and the composed service. It routes messages between the orchestration process and its clients. This Web service is essential in case of orchestration process migration, since it ensures the routing of the response resulting from the new orchestration process host towards the initial client. In spite of this centralized architecture which makes all invocation pass through the WSIM, the latter does not constitute a bottleneck. In fact, the WSIM doesn't make any treatment apart of invoking the orchestration process, waiting for its response, and routing it to the client. Thus, if the number of invocation increases the orchestration process will be overloaded before the WSIM. Besides, the WSIM hosting node should be reliable since its interruption necessitates the explicit recall of the orchestration process by the client in order to resume the execution.

The Web Service Checkpoint Manager (WSCM) is a service which can be deployed on the same host of the orchestration process. It is responsible of the management of instances checkpoints. When the checkpoint aspect is deployed, every concerned instance calls the WSCM, through the checkpointing aspect code, for saving a copy of its state (checkpoint). Thus, the WSCM saves all captured checkpoints on a remote database in order to enable their possible use for resuming interrupted instances. In practice, in order to avoid the loss of checkpoints, the checkpoint database should be deployed on a reliable host or duplication strategies should be employed to enable their recovery.

As illustrated in Figure 1, all invocations of the orchestration process must pass through the WSIM (1). Generally, each invocation involves the creation and the execution of a new orchestration process instance. Each time a checkpoint is required the WSCM deploys the appropriate checkpoint aspect (2) which sends the current state, of each concerned instance, to the WSCM (3). The latter, stores these checkpoints on a remote database (4). When failure occurs or mobility

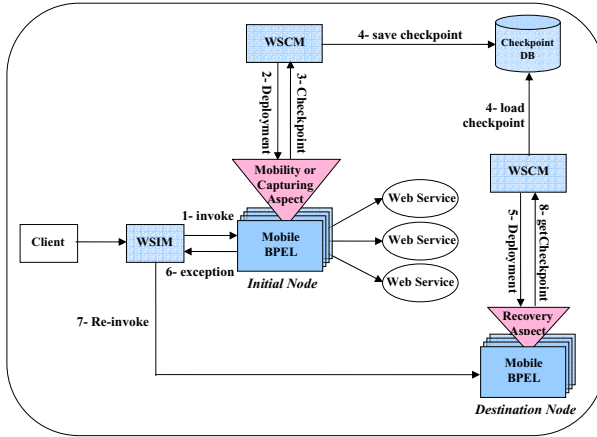


Fig. 1. Aspect-based Architecture for orchestration processes checkpointing

is launched through mobility aspect deployment, a copy of the orchestration process and the recovery aspect (5) will be deployed on a new server and the mobility aspect interrupts all (or a subset of) the orchestration process running instances. Then, the mobility aspect propagates an error (6) to the WSIM. The latter (7) re-sends all interrupted invocations to the new orchestration process in order to resume the stopped instances. Upon receiving these requests the new orchestration process creates, for each one, a new instance. And the recovery aspect contacts the WSCM to get the last captured checkpoint (8) stored in the remote data base (4), and integrates it within the instance execution which can then resume its execution starting from the loaded checkpoint.

3.2 Aspects and Transformation Rules for Strongly Mobile Composed Web Service

Ensuring strong mobility of a BPEL process consists in integrating the capability to capture its execution state (checkpoint), as well as the possibility of loading a checkpoint (re-establish) in order to resume the execution starting from it. In our approach, these functionalities are carried out thanks to code transformation of the initial BPEL process and aspect deployment. Indeed, the code of the process is instrumented in order to maintain perpetually the updated state of each running instance. Thus, Aspects may be deployed at any execution time in order to capture this state, recover it, or launch instance mobility. The transformation rules may be found in [12]. They correspond mainly to the addition of a set of variable reflecting the process state and the transformation of basic and structured activities in order to manage the artificial program counter. In the next subsections, we present the Aspects allowing checkpointing, mobility, and recovery of a BPEL orchestration process.

```

<aspect name="ForcedMobility">
  <pointcut condition="getVariableData('instanceID') == '123'">
    //process[@name="travelPackage"]//invoke |
    //process[@name="travelPackage"]//receive |
    //process[@name="travelPackage"]//assign |
    //process[@name="travelPackage"]//wait |
    //process[@name="travelPackage"]//switch |
    //process[@name="travelPackage"]//while
  </pointcut>
  <advise type="before">
    <bpws:sequence>
      <bpws:throw name="throw" faultName="bpws:forcedTermination"/>
    </bpws:sequence>
  </advise>
</aspect>

```

Fig. 2. Aspect implementing mobility

```

<aspect name="CheckpointAtSynchronizationBarriers">
  <partners>
    <partner name="wscm" serviceLinkType="tns:wscmSLT"/>
  </partners>
  <variables>
    <variable name="setrequest" messageType="tns:setStateRequest"/>
    <variable name="setresponse" messageType="tns:setStateResponse"/>
  </variables>
  <pointcut>
    //process[@name="travelPackage"]//assign[@name="ActivityCounterUpdateIn
    sequence"]
  </pointcut>
  <advise type="after">
    <bpws:assign>
      <!-- copy instance identifier to the setrequest variable -->
      <!-- copy activity counters identifier to the setrequest
      variable -->
      <!-- copy original variables to the setrequest variable -->
    </bpws:assign>
    <bpws:invoke name="invokeWSCMService" partner="wscm"
    portType="tns:WSCM" operation="setState"
    inputVariable="setrequest"
    outputVariable="setresponse">
    </bpws:invoke>
    <!-- End of the Checkpoint Bloc -->
  </advise>
</aspect>

```

Fig. 3. Aspect implementing checkpoint at natural synchronization barriers

```

<aspect name="ForcedCheckpoint">
  <partners>
    <partner name="wscm" serviceLinkType="tns:wscmSLT"/>
  </partners>
  <variables>
    <variable name="setrequest" messageType="tns:setStateRequest"/>
    <variable name="setresponse" messageType="tns:setStateResponse"/>
  </variables>
  <pointcut>
    //process[@name="travelPackage"]//assign[@name="ActivityCounterUpdate"]
  </pointcut>
  <advise type="after">
    <!-- Beginning of the forced checkpoint Bloc -->
    <bpws:assign>
      <!-- put checkpointIndicator to 1 -->
    </bpws:assign>
    <!-- if synchronization is done -->
    <bpws:switch>
      <bpws:case condition=
      "bpws:getVariableData('NbBranches') = bpws:getVariableData('NbSynch')">
        <bpws:assign>
          <!-- copy instance identifier to the setrequest variable -->
          <!-- copy NbBranches to the setrequest variable -->
          <!-- copy activity counters identifier to the setrequest
          variable -->
          <!-- copy original variables to the setrequest variable -->
        </bpws:assign>
        <bpws:invoke name="invokeWSCMService" partner="wscm"
        portType="tns:WSCM" operation="setState"
        inputVariable="setrequest"
        outputVariable="setresponse">
        </bpws:invoke>
        <bpws:assign>
          <!-- put the CheckpointIndicator to 0 -->
        </bpws:assign>
      </bpws:case>
      <!-- if synchronization is not yet done -->
      <bpws:otherwise>
        <!-- increment NbSynch -->
        <bpws:while condition="bpws:getVariableData('CheckpointIndicator')
        = 1">
          <empty>
        </while>
      </bpws:otherwise>
    </bpws:switch>
    <!-- End of the Checkpoint Bloc -->
  </advise>
</aspect>

```

Fig. 4. Aspect implementing forced checkpointing

Capturing and Re-establishing orchestration process States. The capturing Aspects may be deployed at any execution time and could enact different checkpointing techniques such as checkpointing at the next natural synchronization barrier, forced checkpoint, immediate checkpoint and migration. The choose of one of these Aspects to be deployed is out of the scope of this paper and depends on the current execution conditions and on the fixed objectives. As presented in Figure 3, checkpointing at natural synchronization barrier corresponds to waiting until reaching a sequential execution (i.e. execution with no parallel branches) to enact the checkpoint. In fact, at such position, capturing the checkpoint is simple and requires only sending the current state to the WSCM after the program counter update. In contrast, forced checkpointing (see Figure 4) involves synchronizing all parallel branches, before saving the execution state, in order to ensure a consistent checkpoint (see [12] for more details). The checkpoint at migration is similar to the forced checkpointing scenario, but it requires in addition forcing the migration by throwing a notification to the WSIM and stopping the orchestration process execution just after saving the state. Another checkpointing technique consists in using message logging to avoid rollback. Therefore, we may deploy with the forced checkpointing aspect an aspect ensuring the message logging functionality. In fact, this corresponds simply to create an aspect which is activated after *invoke* and *receive* activities and which saves messages in the WSCM.

The recovery aspect is deployed after the migration of an orchestration process. In such a case, the recovery aspect will be deployed on the destination server in order to load the last saved checkpoint of each interrupted instance, integrates it in the orchestration process instance and then resumes its execution. In case of uncoordinated checkpointing with message logging, the recovery aspect should replace *invoke* and *receive* activities by a call of the WSCM to get results.

For asynchronous communications, a problem may arise if a checkpoint takes place after an *invoke* and before its corresponding *receive*. In such a case, the invoked partner service will send the response to the initial orchestration node, and the current orchestration process will wait indefinitely to the response. For solving this problem, we employ, currently, the simple solution of re-invoking, after migration, all non-accomplished asynchronous invocations. This solution ensures the continuity and the correctness of the orchestration process execution after migration while avoiding restarting the whole orchestration.

Many mobility scenario involves the re-invocation of partner Web services like the case of rollback or asynchronous communications. The re-invocation of stateless Web service does not arise any problems. However, if we deal with re-executing stateful Web services or services persisting information in a database (e.g. payment), the re-execution may consistency problems. For avoiding such issues, we propose to deploy another WSIM between the orchestration process and any stateful partner. Thereby, after migration the orchestration process can get the response from the WSIM without re-invoking the stateful Web service.

Orchestration Process Mobility. Mobility of orchestrated process may be undertaken in case of hosting node failure or a QoS violation. In case of failure, the process is brutally interrupted, so the only solution corresponds to resuming all interrupted instances on the new host by re-invoking them through the WSIM.

In case of QoS violation, mobility corresponds to an external decision forced by deploying the mobility aspect which interrupts the instance execution and propagates exception to the WSIM. This aspect may have a general pointcut which allows its immediate execution for all running instances (i.e. without the condition attribute). Or, it may include an instance filtering pointcut which selects the target instances concerned with the mobility. In Figure 2, we employ a simple filter which selects one instance, but we may filter on other parameters such as the number of migrations or the passed execution time. In such a case, the used parameter should be added to the orchestration process code. Thus, the current solution allows immediate mobility by deploying the mobility aspect. Unlike the periodic checkpoint approach [12] which allows mobility only at the next checkpoint position.

4 Implementation and Tests

We evaluate our approach using a travel agency case study. Through this case study, we aim to estimate the overhead introduced by our solution. We evaluate

the benefit of the dynamic choice of the approach to a failure scenario where no checkpoint is employed and where the whole application is re-executed. checkpointing instant at runtime. Moreover, we compare our mobility

The travel agency application consists in a BPEL process orchestrating a set of Airline Company and Hotel services in order to get the shipper offer which satisfies a client request. The BPEL process invokes the airline services in parallel to get their offers then compares them in order to choose the shipper one and then it does the same operations with the hotel services. Finally, the BPEL process returns to the client the shipper offers. We instrument this application in order to add mobility support to the BPEL process. We deploy our strong mobility architecture, constituted of the client, WSIM, WSCM, initial BPEL host, and final BPEL host, using five machines having each 3GHz CPU and 500Mb of memory. The WSIM is implemented as a BPEL process and the WSCM corresponds to a stateless Web service communicating with a distant Data Base. For all mobility evaluations, we deploy the BPEL process in advance on the destination host. Thus, the overhead of code mobility and BPEL deployment is not considered in the experimentations. In this evaluation, the mobility of all running instances towards another host is initiated through mobility aspect deployment. this mobility may be needed in case of failure prediction or a serious performance degradation of the orchestration hosting node.

Figure 5 presents the evaluation of the execution time of one running instance of the BPEL process including checkpointing, migration, and recovery, while varying the checkpoint number. In this evaluation checkpointing is done using the forced checkpointing aspect. For each checkpoint number value, we show five evaluations. The first one (1) corresponds to the execution time of the original BPEL process. Then, we present three evaluations showing the execution time of the transformed and checkpointed BPEL process with migration and rollback when migration occurs just after a checkpoint (2), when migration occurs between two checkpoints (3), and when migration occurs just before a checkpoint (4). And (5) corresponds to the evaluation of mobility without checkpointing. Those evaluations shows that the nearest the migration position and the last captured checkpoint are, the lower the rollback overhead is. We notice also that when we employ only two checkpoints, the rollback overhead in the worst case is very high. This makes the overall execution time of the migrating BPEL process using our solution almost equals to the execution time necessary for BPEL process re-execution. Using four checkpoints gives best results while using six gives worst ones. This proves clearly that the selection of the checkpoint interval is fundamental to make a checkpointing solution efficient. This fact makes our solution very useful as it may be used to dynamically tune the checkpoint number according to mobility performances.

In Figure 6, we vary the number of instances which are executed in parallel by varying the number of clients. As the previous experimentation shows that the best checkpoint number value is four, we use the forced checkpointing and the checkpoint at natural synchronization barriers Aspects deployed alternatively four times during execution, and we deploy the mobility aspect for migrating

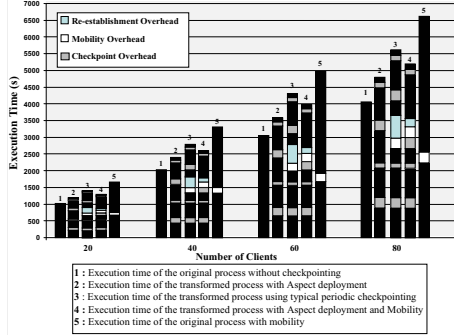
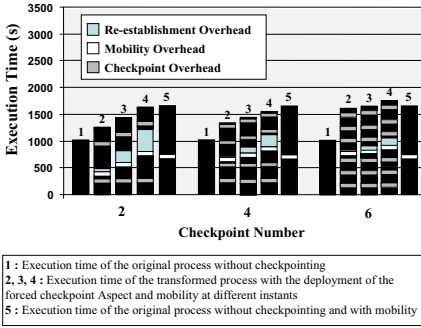


Fig. 5. Evaluation of the influence of the checkpoint number and the migration instant on strong mobility performance

Fig. 6. Comparison between the overheads of our solution, the overhead of periodic checkpointing and the overhead of orchestration process re-execution

all running instances. Thus, we experiment the use of different checkpointing techniques by dynamically deploying Aspects. In Figure 6, we present five evaluations for every client number value. The first evaluation (1) corresponds to the execution of the original process for which no transformation is done and no Aspects are deployed. The third one (3) corresponds to the execution time of the transformed process using typical periodic checkpointing and without aspect. We notice there the overhead of checkpoint, reestablishment and mobility. The second one (2) corresponds to the execution time of the transformed process with the deployment of the checkpoint Aspects four times. The overhead introduced by our transformation without aspect deployment does not exceed 17%, and with the deployment of Aspects (four times in this case) it reaches 20% of overhead compared to the original process. Those overheads are close to the typical checkpointing approach. We can easily notice the difference in the execution time of the checkpointing aspect for the same number of instances. In fact, the time needed for saving a checkpoint depends on the number of running branches at aspect execution time. The larger the number of branches is, the greater the synchronization is, and the more important the checkpoint time is. The fourth value in the evaluation results (4) corresponds to the evaluation of the mobility of all running instances. When mobility is initiated we deploy the checkpointing aspect to avoid the rollback overhead. The overhead of the mobility and recovery actions increases for larger instance number but does not exceed 2% compared to the original process execution time. In addition, compared to the typical periodic checkpointing approach the re-establishment overhead is smaller in our approach since no rollback is needed. The last value (5) corresponds to the mobility of the original orchestration process with no source code transformation and no checkpoint or recovery aspect but only the mobility aspect is deployed. Thus, after mobility, the whole process is restarted from the beginning. Our approach offers clearly better performances especially when mobility

occurs in an advanced stage of the orchestration execution. In this evaluation, we launch mobility at the middle of the process execution. We notice that the mean mobility overhead of our approach is about 22% of the original execution time. Besides, the mean mobility overhead of the original process reaches 60% of the original execution time. The overhead of the orchestration process strong mobility is almost constant whatever the mobility instant is. The mobility of the original process has a small overhead when mobility occurs at the process beginning, about 2% of the original execution time, but a very large overhead when mobility occurs at the process end which may reach 100% of the original execution time.

5 Conclusion

We presented in this paper a modular aspect-oriented approach for composed services checkpointing. Thanks to Aspects, our solution offers a flexible orchestration process checkpointing support allowing the dynamic choose of the instant and the technique of checkpointing. Thereby, depending on the execution context, such as the deployment environment state, the required QoS, etc, the need of checkpointing may be observed and immediately enacted by deploying the appropriate Aspects. Thus the whole orchestration process or a sub set of its running instances may be migrated to another hosting node and resumed starting from the last saved checkpoint. This solution may be employed for accomplishing several objectives like self-adaptation and load balancing.

We implemented and evaluated this approach using a travel agency case study deployed on the AO4BPEL engine, and we proved the usability and the efficiency of our solution compared to mobility solutions involving the re-execution of the whole application and also compared to periodic checkpointing approaches. In the future, we plan to enhance our solution by proposing an approach for automatically calculating the appropriate checkpointing interval. Moreover, we aim to complete the adaption process with monitoring, diagnostic, planning and executing phases.

References

1. Vadhiyar, S.S., Dongarra, J.J.: Self adaptivity in Grid computing: Research Articles. *Concurrency Computation: Practice and Experience* 17(2-4), 235–257 (2005)
2. Cappello, F., Djilali, S., Fedak, G., Herault, T., Magniette, F., Néri, V., Lodygensky, O.: Computing on large-scale distributed systems: Xtrem Web architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems* 21(3), 417–437 (2005)
3. Allen, G., Angulo, D., Foster, I., Lanfermann, G., Liu, C., Radke, T., Seidel, E., Shalf, J.: The Cactus Worm: Experiments with dynamic resource discovery and allocation in a Grid environment. *The International Journal of High Performance Computing Applications* 15(4), 345–358 (2001)
4. Camargo, R.Y.D., Goldchleger, A., Kon, F., Goldman, A.: Checkpointing BSP parallel applications on the InteGrade Grid middleware: Research Articles. *Concurrent Computing: Practice and Experience* 18(6), 567–579 (2006)

5. Lemarinié, P., Bouteiller, A., Krawezik, G., Cappello, F.: Coordinated checkpoint versus message log for fault tolerant mpi. *Int. J. High Perform. Comput. Netw.* 2(2-4), 146–155 (2004)
6. Ezenwoye, O., Sadjadi, S.M.: TRAP/BPEL: A framework for dynamic adaptation of composite services. In: *Proceedings of the International Conference on Web Information Systems and Technologies, WEBIST 2007* (2007)
7. Halima, R.B., Drira, K., Jmaiel, M.: A qos-oriented reconfigurable middleware for self-healing web services. In: *IEEE International Conference on Web Services, ICWS 2008, China*, pp. 104–111. IEEE Computer Society, Los Alamitos (2008)
8. Baresi, L., Ghezzi, C., Guinea, S.: Towards self-healing composition of services. In: Krämer, B.J., Halang, W.A. (eds.) *Contributions to Ubiquitous Computing. SCI*, vol. 42, pp. 27–46. Springer, Heidelberg (2007)
9. Charfi, A., Mezini, M.: Ao4bpel: An aspect-oriented extension to bpel. *World Wide Web* 10(3), 309–344 (2007)
10. Chandy, K.M., Lamport, L.: Distributed snapshots: determining global states of distributed systems. *ACM Transaction Computer Systems* 3(1), 63–75 (1985)
11. Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: *Business Process Execution Language for Web Services (BPEL4WS) Version 1.1* (May 2003)
12. Marzouk, S., Maâlej, A.J., Rodriguez, I.B., Jmaiel, M.: Periodic checkpointing for strong mobility of orchestrated web services. In: *Proceedings of the International Workshop on Self Healing Web Services (SHWS 2009) in conjunction of the 7th IEEE International Conference on Web Services (ICWS 2009)*, LosAngeles, California, USA. ACM, New York (2009)

A Runtime Performance Analysis for Web Service-Based Applications

Afef Mdhaffar, Soumaya Marzouk, Riadh Ben Halima, and Mohamed Jmaiel

University of Sfax, ReDCAD Laboratory, B.P. 1173, Sfax-Tunisia
{afef.mdhaffar,soumaya.marzouk}@redcad.org,
{riadh.benhalima,mohamed.jmaiel}@enis.rnu.tn

Abstract. Runtime performance evaluation is necessary for the continuous QoS (Quality of Service) management of Web service-based applications. The analysis is critical in service provisioning since it allows to detect QoS degradation and to identify its source. However, performance analysis in current applications is usually based on QoS reference values that are manually instrumented and pre-established independently from the execution context. It is inflexible to change. The paper extends our previous research on performance evaluation within a self-healing framework and proposes a novel analysis approach based on automatic generation of QoS reference values. These QoS reference values are generated whenever we start the application execution in a new context. This approach enables the detection of QoS degradation, the identification of its nature (like execution or communication level) and the localization of its cause (as scalability issue, node failure or network overload). The carried out experiments show that the dynamic generated QoS reference values are suitable and their associated analysis results are accurate.

Keywords: QoS reference values, Self-healing, Web services, Performance analysis.

1 Introduction

Web services become crucial and largely used in the design of distributed applications. To ensure their efficiency, Web services need to support self-healing properties in order to deal with dynamic execution context. To support such properties, a four phases process has been established [1]. The first phase consists in monitoring the Web service and its hosting environment in order to prepare the necessary measures to the analysis phase. The latter uses these metrics to detect possible failures or QoS degradation, identify the degradation nature and localize its cause. In such case, the third phase takes place. It looks for an equivalent Web service to replace the degraded one. Then, the fourth phase enforces repair actions. In a previous work [2], we performed the analysis based on pre-established QoS reference values. Therefore, we experimented each Web service on a large scale environment [3] to determine its QoS reference values. This is

hard to do because it requires human competences, specific tools and time. In addition, these reference values are related to the execution context. So, we need to repeat experiments whenever we (i) update the service implementation, (ii) change the hosting server, and (iii) modify the execution context.

Many existing works deal with the performance analysis of Web services. We notice that all studied works [4,5,6,7,8,9] rely on the use of a pre-established QoS reference values describing the expected performance of a Web service on a specific context. Such approaches are not suitable for dynamic context. In fact, it is very hard to manually generate QoS reference values for each Web service on every context. Also, we picked out that the majority of existing works [4,5,6,7,8,9] does not allow an accurate localization of the QoS degradation cause. It can lead to useless reconfiguration actions.

In this paper, we propose a novel approach of performance analysis based on automatic and dynamic generation of QoS reference values at runtime. This approach consists in (1) monitoring the variation of (i) QoS parameters values, (ii) node load and (iii) network load during a training period while using a previously developed QoS monitoring approach and specific tools to check the overload and (2) studying the variation of monitored values and the correlation between them. Through this study, we distinguish different reasoning scenarios in order to characterize the Web state and generate the QoS reference values from scratch. We implemented these scenarios as algorithms in order to automate the generation process. For instance, if the Web service state is acceptable¹, the average of the monitored values represents the QoS reference values. Otherwise, the algorithm alerts about the performance degradation, characterizes its cause and enforces recovery actions. Our approach regenerates the QoS reference values whenever the Web service changes its execution context.

To illustrate the proposed approach, we carried out experiments on the Conference Management System (CMS). It is a real WS-based application that we developed in order to review management in scientific conferences and applicable for industrial review processes. First, we compare the QoS reference values that we generated automatically to a manually generated QoS reference values. The evaluation shows that both values are so close and the difference between them is negligible. So, we conclude that the automatically generated QoS reference values are reasonable. Second, we demonstrate that the performance analysis based on the automatically generated QoS reference values is achieved rigorously. It detects the performance degradation of a Web service, characterizes its nature, and locates its cause. It can specify whether the degradation has been occurred due to a scalability issue, or due to the network performance degradation or due to the node performance degradation.

This paper is organized as follows. Section 2 describes our contribution and details the generation process of the QoS reference values. Section 3 illustrates our approach. Section 4 discusses related work. Last section concludes the paper.

¹ A Web service execution/communication state is acceptable, if no execution/communication time degradation is detected.

2 A Performance Analysis Approach Based on Automatic and Dynamic Generation of QoS Reference Values

We propose in this work a performance analysis approach for Web services. This approach enhances our previous QoS-Oriented Self-Healing middleware (*QOSH*) [2] while proposing a new analysis component. In our previous work [2], we used a pre-established QoS reference values for runtime analysis. This makes the used analysis approach [2] inflexible for many reasons. First, its application requires the elaboration of a large number of experiments to generate QoS reference values for each Web service. Second, we have to repeat experiments whenever the execution context changes. Also, our previous approach [2] does not allow an accurate localization of the degradation cause. So, we focus in this paper on (1) the automation of the QoS reference values generation for Web services and (2) the identification of the degradation cause. Thus, QoS (execution and communication time) reference values are generated and updated when the execution context changes without human intervention. Then, our new analysis component uses these values to detect degradation, and it accurately identifies its nature and its cause. This approach is detailed in the following.

2.1 Overview of Our Performance Analysis Approach

Our performance analysis approach shown in Fig. 1 starts with checking the Web service execution state while studying data monitored during the training

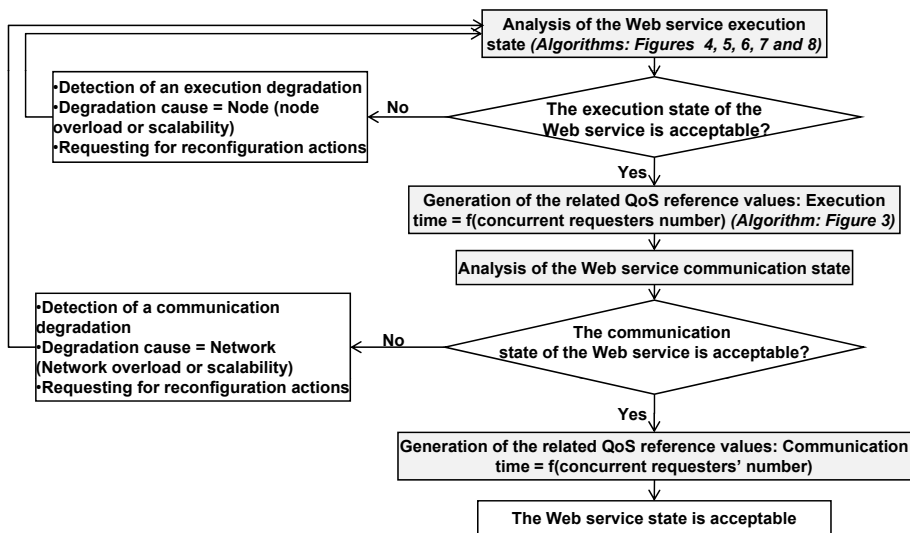


Fig. 1. Organigram of our performance analysis approach

period². In case of an acceptable Web service execution state, our approach generates related QoS (execution time: Fig. 3) reference values and checks the Web service communication state. Then, it generates its related QoS (communication time) reference values in case of an acceptable communication state. Otherwise, when our approach detects a communication or execution degradation, it deduces its cause, requests for reconfiguration actions and restarts our analysis approach. In case of mobile Web services, our approach regenerates the QoS reference values if the concerned Web service migrates to another node. During execution, the established QoS reference values will be updated periodically for consolidation purposes. Our performance analysis approach detects QoS degradation, deduces its nature (execution or communication) and localizes its cause (scalability issue, node or network overload).

2.2 A QoS Reference Values Generation Algorithm

Fig. 2 presents an algorithm that implements our performance analysis approach. It takes as inputs five parameters, which are collected during the monitoring phase. These parameters allow us to check the Web service state. They are detailed in Table 1. At the instant t_i , we measure the execution time ($T_{exec}(t_i)$), the request number ($NbReq(t_i)$) and the communication time ($T_{com}(t_i)$), while

```

1  Analysis : Begin
2     $WsState\_T_{exec} = \text{Analysis\_Execution\_State}(NbReq(t), T_{exec}(t), NodeLoad(t))$ 
3    If ( $IsOK(WsState\_T_{exec})$ ) Then
4       $Generate\_T_{exec\_Reference\_Values}(NbReq(t), T_{exec}(t), NodeLoad(t))$ 
5       $WsState\_T_{com} = \text{Analysis\_Communication\_State}(NbReq(t), T_{com}(t), NetworkLoad(t))$ 
6      If ( $IsOK(WsState\_T_{com})$ ) Then
7         $WS\_State = \text{"Acceptable"}$ 
8         $Generate\_T_{com\_Reference\_Values}(NbReq(t), T_{com}(t), NetworkLoad(t))$ 
9      Else
10        $WS\_State = \text{"Degraded"}$ 
11        $Degradation\_nature = \text{"Communication Degradation"}$ 
12        $Degradation\_cause = \text{"Bandwidth Overload"}$ 
13        $Trigger\_Alarms(Degradation\_nature, Degradation\_cause)$ 
14        $Request\_For\_Reconfiguration\_Actions()$ 
15        $Goto : Analysis$ 
16     EndIf
17   Else
18      $WS\_State = \text{"Degraded"}$ 
19      $Degradation\_nature = \text{"Execution Degradation"}$ 
20      $Degradation\_cause = \text{"Node Overload"}$ 
21      $Trigger\_Alarms(Degradation\_nature, Degradation\_cause)$ 
22      $Request\_For\_Reconfiguration\_Actions()$ 
23      $Goto : Analysis$ 
24   EndIf
25 End

```

Fig. 2. Performance analysis algorithm

² The training period is a time interval which gives us the possibility to assess the Web service performance using monitored QoS parameters, node and network load.

Table 1. The inputs of the performance analysis algorithm

Inputs	Description
NbReq(t)	It presents the variation (Real) of the successive request number during the training period T
Texec(t)	It presents the variation (Real) of the execution time during the training period T
Tcom(t)	It presents the variation (Real) of the communication time during the training period T
NodeLoad(t)	It presents the variation (Real) of the node load during the training period T
NetworkLoad(t)	It presents the variation (Real) of the network load during the training period

```

1 Generate_Texec_Reference_Values(NbReq(t), Texec(t), NodeLoad(t))
2 Begin
3   foreach (ti in T) do
4     If (isOk(Node)) Then
5       SaveOrUpdate (NbReq(ti), Texec(ti))
6 End

```

Fig. 3. The sub-algorithm: *Generate_Texec_Reference_Values*

adopting our monitoring approach [2]. To extract the network load value at instant t_i ($NetworkLoad(t_i)$), we used the *IPERF*³ tool which performs the measurement of the network bandwidth. Finally, to measure the node load at instant t_i ($NodeLoad(t_i)$), we have developed a specific *checking_node_load* service which evaluates the node load while checking the memory occupation variation and the response time variation of this node. The algorithm shown in Fig. 2 makes use of four sub-algorithms (emphasized in bold). The first one, *Analysis_Execution_State* (see Section 2.3), allows the execution time analysis of the Web service. The second one, *Generate_Texec_Reference_Values* (see Fig. 3), generates the execution time reference values. The third sub-algorithm, *Analysis_Communication_State* performs the analysis of the communication time. The fourth one, *Generate_Tcom_Reference_Values*, is very similar to the *Generate_Texec_Reference_Values* sub-algorithm and generates the communication time reference values. It consists on saving or updating ($NbReq(t_i)$, $Tcom(t_i)$) when the network is not overloaded. The execution time analysis and the communication time analysis sub-algorithms follow the same way of reasoning. So, we detail only the execution time analysis sub-algorithm.

2.3 Execution Time Analysis Sub-algorithm

The main idea of the execution time analysis sub-algorithm (*Analysis_Execution_State* ($NbReq(t)$, $Texec(t)$, $NodeLoad(t)$)) is to study the positive correlation between $NbReq(t)$ and $Texec(t)$. In fact, these two distributions should be correlated to reflect a normal behavior of the Web service. The correlation study is based on the computing of the correlation coefficient value using the equation (1) [10].

³ <http://iperf.sourceforge.net/>

$$\rho = \frac{\sum(x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2} * \sqrt{\sum(y_i - \bar{y})^2}} \quad \text{with } \bar{x} = \frac{1}{N} * \sum(x_i) \quad (1)$$

If the correlation coefficient value ρ is greater than 0.8 [10], we conclude that $NbReq(t)$ [x] and $Texec(t)$ [y] are positively correlated. Then, the execution time analysis sub-algorithm begins with checking the positive correlation between $NbReq(t)$ and $Texec(t)$. If they are positively correlated (Fig. 4 line 1), it concludes that the behavior of the concerned Web service is normal but it can't deduce that the execution state of this Web service is acceptable. Therefore, a continuous degradation of the concerned Web service performance can also lead to a positive correlation of these two distributions. As a result, we check the hosting node load to evaluate the state of the concerned Web service. If the hosting node is not overloaded (Fig. 4 line 2), this sub-algorithm concludes that the Web service execution state is acceptable (Fig. 4 line 3). Otherwise (Fig. 4 line 4), it deduces that the degradation comes from the execution time (Fig. 4 lines 5 and 6) and is related to a scalability issue (Fig. 4 line 7). In fact, the hosting node can't support this important number of simultaneous requests.

```

1  If (IsCorrelated(Texec, NbReq)) Then
2  If (not(IsOverloaded(node))) Then
3  WsStateExec = "Acceptable"
4  Else
5  WS_State = "Degraded"
6  Degradation_Nature = "Degradation of the execution time"
7  Degradation_Cause = "Node/Scalability"
8  EndIf

```

Fig. 4. Execution time analysis sub-algorithm - Part 1: The two distributions are positively correlated

If $NbReq(t)$ and $Texec(t)$ are not positively correlated, the execution time analysis algorithm checks the variation of $Texec(t)$ to examine the Web service state. It distinguishes between four main cases: (i) $Texec(t)$ is increasing (Fig. 5), (ii) $Texec(t)$ is decreasing (Fig. 6), (iii) $Texec(t)$ is constant (Fig. 7) and (iv) $Texec(t)$ follows a random variation (Fig. 8).

We detail the case when $Texec(t)$ is increasing and not positively correlated with $NbReq(t)$ (Fig. 5). The non correlation between these two distributions implies that the increase of the execution time value is not related to the simultaneous request number variation. Thus, it is related to a node performance decrease. In this case, we have to know if this node performance decrease is critical in order to characterize the concerned Web service state. This is achieved by checking the hosting node load. If the node is not overloaded (Fig. 5 line 3), our algorithm deduces that the execution state of the concerned Web service is acceptable (Fig. 5 line 4). Otherwise (Fig. 5 line 5), we deduce that an execution degradation has occurred (Fig. 5 lines 6 and 7). To locate the degradation cause, we analyze $NbReq(t)$. If it is constant or decreasing (Fig. 5 line 8), we conclude

```

1  If (not(IsCorrelated(Texec, NbReq))) Then
2    If (increasing(Texec)) Then
3      If (not(isOverloaded(node))) Then
4        WsStateExec = "acceptable"
5      Else
6        WsStateExec = "Degraded"
7        Degradation_Nature = "Degradation of the execution time"
8      If (decreasing(NbReq) or Constant(NbReq)) Then
9        Degradation_Cause = "Node"
10     Else
11       Degradation_Cause = "Node/scalability"
12     EndIf
13   EndIf

```

Fig. 5. Execution time analysis sub-algorithm - Part 2.1: Increase of the execution time during the training period

```

1  If (not(IsCorrelated(Texec, NbReq))) Then
2    If (decreasing(Texec)) Then
3      WsStateExec = "acceptable"
4    EndIf

```

Fig. 6. Execution time analysis sub-algorithm - Part 2.2: Decrease of the execution time during the training period

that the hosting node is the cause of the degradation (Fig. 5: line 9). Otherwise, (i.e. the variation is random) our algorithm deduces that the degradation would be related to a scalability issue or a node overload (Fig. 5: line 11).

The execution time analysis algorithm handles the case when $Texec(t)$ is decreasing and not correlated with $NbReq(t)$ in Fig. 6. It deduces that the Web service state is acceptable. In fact, the decrease of the execution time value implies that the hosting node offers a better QoS.

The algorithm shown in Fig. 7 deals with the case when $Texec(t)$ is constant and not correlated with $NbReq(t)$. It analyzes the hosting node load to characterize the Web service state. If the hosting node is not overloaded (Fig. 7: line 3), it concludes that the Web service execution state is acceptable (Fig. 7: line 4). In the opposite case (Fig. 7: line 5), we deduce that an execution time degradation has occurred (Fig. 7: lines 6 and 7). To locate the degradation cause, this sub-algorithm analyzes the variation of $NbReq(t)$. If it decreases (Fig. 7: line 8),

```

1  If (not(IsCorrelated(Texec, NbReq))) Then
2    If (Constant(Texec)) Then
3      If (not(isOverloaded(node))) Then
4        WsStateExec = "acceptable"
5      Else
6        WsStateExec = "Degraded"
7        Degradation_Nature = "Degradation of the execution time"
8      If (Decreasing(NbReq)) Then
9        Degradation_Cause = "Node"
10     Else if (Increasing(NbReq)) Then
11       Degradation_Cause = "Scalability"
12     Else
13       Degradation_Cause = "Node/Scalability"
14     EndIf
15   EndIf

```

Fig. 7. Execution time analysis sub-algorithm - Part 2.3: The execution time is constant during the training period

```

1  If(not(IsCorrelated(Texec, NbReq))) Then
2    If(random(Texec) then
3      If(not(isOverloaded(node))) Then
4        WsStateExec = "acceptable"
5      Else
6        WsStateExec = "Degraded"
7        Degradation_Nature = "Degradation of the execution time"
8      If(Decreasing(NbReq) Or Constant(NbReq)) Then
9        Degradation_Cause = "Node"
10     Else
11       Degradation_Cause = "Node/scalability"
12     EndIf
13   EndIf

```

Fig. 8. Execution time analysis sub-algorithm - Part 2.4: The execution time is random during the training period

it deduces that the degradation comes from the hosting node (Fig. 7 line 9). Otherwise, if the simultaneous request number is increasing (Fig. 7 line 10), our sub-algorithm deduces that the QoS degradation is related to a scalability issue (Fig. 7 line 11). Else, (i.e., $NbReq(t)$ is random), it concludes that the cause of the QoS degradation is related to a scalability issue and/or a hosting node overload (Fig. 7 line 13).

In Fig. 8, this sub-algorithm handles the case when $Texec(t)$ is random and not correlated with $NbReq(t)$. In order to evaluate the Web service state, it checks the hosting node load. If it is not overloaded (Fig. 8 line 3), our sub-algorithm concludes that the state of the concerned Web service is acceptable (Fig. 8 line 4). In the opposite case (Fig. 8 line 5), our sub-algorithm deduces that a degradation of the execution time has occurred (Fig. 8 lines 6 and 7). In order to locate the degradation cause, it checks the variation of the simultaneous request number. If it is decreasing or constant (Fig. 8 line 8), it deduces that the hosting node is the degradation cause (Fig. 8 line 9). Otherwise (Fig. 8 line 10), we conclude that the degradation is related to a scalability issue and/or to a hosting node overload (Fig. 8 line 11). In the following, we present carried out experiments illustrating our approach.

3 Illustration

To validate our approach, we carried out two kinds of experiments. In the first one, we compare the manually generated QoS reference values with the QoS reference values automatically generated within our approach. We note that these reference values are so close. In the second experiment, we show that the analysis using the automatically generated QoS reference values is achieved rigorously. In fact our analysis approach detects QoS degradation and allows an accurate identification of its nature and its cause.

The cooperative reviewing process starts by searching a suitable conference for authors. So, they send requests to the *ConfSearch* Web service looking for appropriate conferences (topics, publisher, etc.). In this paper, we focus on the

performance analysis of the *ConfSearch* Web service. The experiment platform is composed of three nodes. The first contains Intel Pentium Dual Core T2310 as CPU with 1,46 GHZ as frequency and operates under Windows XP. The two other nodes operate under Ubuntu Linux and have each other an Intel Pentium Dual Core as CPU with 3 GHZ as frequency. The application is composed of three main actors. The analyzer is deployed under the first node and implements our algorithm. This node contains also the monitoring database. The second actor is the *ConfSearch* Web service which is deployed under the second node. The Web service client represents the third actor, which is deployed under the third node. We use Apache Tomcat5.5 as Web server, Axis1.4 as SOAP engine, Java1.5 as programming language and MySQL5 as database management system.

3.1 Validation of the Automatically Generated QoS Reference Values

This experiment consists in comparing the QoS reference values, which are automatically generated by our algorithm, with the manually generated QoS reference values. It is composed of two phases. In the first one, we manually establish the QoS reference values of the *ConfSearch* Web service while running this Web service in optimal conditions. So, we run respectively 2-24, 32, 34 and 42 requests. For each request, we monitor QoS parameters, extract its values and measure network and node loads using specific tools (our monitoring middleware [2], IPERF and a specific *checking_node.load* service). This first phase is carried out 10 times. The graphs *Texec_manual_ref* (baby blue line) of the Fig. 9 and *Tcom_manual_ref* (orange line) of the Fig. 10 represent their average. The second phase corresponds to the training period. It consists in applying our approach in order to automatically generate the QoS reference values. During this phase, we have injected some transient violations in order to illustrate the capacity of our approach to eliminate the degraded QoS values from the generated reference values. We run respectively 2-24, 32, 34 and 42 requests. For each request, we monitor QoS parameters and measure node and network loads. For the execution time, we injected a delay on the *ConfSearch* Web service (while running several processes in the hosting node) in the 5th, 9th, 11th, 14th, 16th and 20th requests. For the communication time, we injected a network overload (while using the *HPING*⁴ tool) in the 5th, 9th, 18th, 20th, 21st and 22nd requests. This phase is also carried out 10 times. The graphs *Texec(with transient violations)* (blue line) of the Fig. 9 and *Tcom(with transient violations)* (red line) of the Fig. 10 show the experiment results that we conducted. At this phase end, our algorithm generates QoS reference values. They are shown in Fig. 9 (*Texec_ref_algo*) and 10 (*Tcom_ref_algo*). As presented in Fig. 9, our algorithm has correctly generated the execution time reference values (*Tcom_ref_algo*) (green line) which are very close to the manually generated one. In addition, Fig. 10 illustrates that the manual communication time reference values (*Tcom_manual_ref*) are very close to the communication time reference values generated by our algorithm

⁴ <http://www.hping.org/>

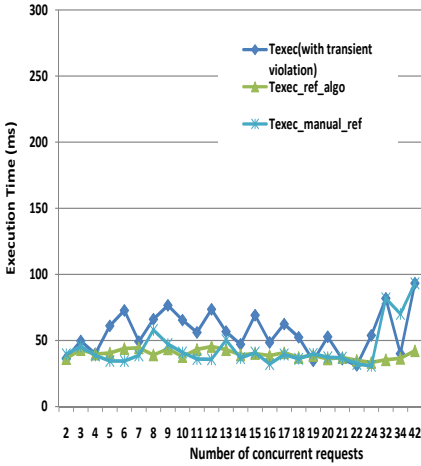


Fig. 9. Texec reference values

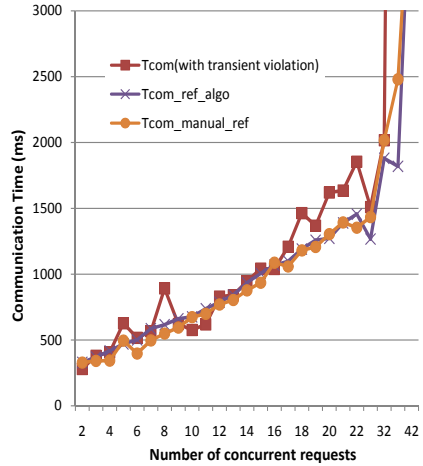


Fig. 10. Tcom reference values

(*Tcom_ref_algo*) (purple line). Fig. 9 and 10 illustrate that the degraded values (due to injected transient violations) are not selected for the generated QoS reference values. So, we conclude that the automation of QoS reference values generation gives an efficient result.

3.2 Efficiency of Our Approach

In order to illustrate the efficiency of our performance analysis approach, we injected three types of failures. This is detailed in the following.

-*First failure: Network overload.* This experiment consists in generating a flooding attack using the *HPING* tool while running our analyzer. We noticed that our algorithm detects a communication degradation and identifies its cause which is a network overload. This illustrates the efficiency of our approach.

-*Second failure: Node overload.* In this experiment, we generate a node failure and we follow the behavior of our analyzer. So, we run simultaneously a lot of processes on the *ConfSearch* Web service hosting node. The proposed algorithm detects an execution degradation and deduces its cause: node overload.

-*Third failure: Scalability issue.* In this experiment, we progressively increase the simultaneous request number to cause a scalability issue. We note that our analyzer detects the degradation and deduces its cause: a scalability issue.

This illustrates that our approach offers an accurate localization of the degradation cause. It checks the node load, the network load and the variation of the QoS parameters to allow this accurate identification of the degradation cause.

4 Related Work

Works dealing with the analysis of Web services may be classified in three categories. The first one is based on QoS parameters values comparison with

pre-established thresholds [8]. The approach of Tian et al. [8] monitors QoS parameters and checks the pre-defined thresholds. Thus, a degradation is detected when monitored values exceed thresholds. This approach performs the detection of the degradation while using a pre-established QoS reference values and does not allow neither the characterization of its nature nor the localization of its cause. However, our approach does not use a pre-established reference values and allows degradation detection, its nature characterization and its cause localization.

The second category is based on the model basis diagnosis technique [9,4,5,7]. It models the acceptable behavior of the monitored Web service. Then, the failure is detected when a deviation from the pre-established reference model [5] is detected. For instance, Yan et al. [9] use the BPEL specifications to model the normal behavior of the monitored Web service. When a deviation from the pre-established model is detected, a degradation is signaled. This solution enables the degradation detection and the faulty Web service localization. So, it does not allow an accurate localization of the degradation cause and uses a pre-established reference model. However, our approach does not use a pre-established reference model and allows an accurate localization of the degradation cause.

Finally, the third category [6] is based on the representation of the acceptable behavior of the monitored Web service by a set of policies. Degradation is assimilated to a deviation from the pre-established model. This work [6], contrary to our approach, relies on the use of pre-established QoS reference values and does not allow neither the characterization of the degradation nature, nor a precise localization of its cause.

We noticed that studied works proposed an analysis solution based on the use of a pre-established reference model (QoS reference values / reference model) and do not offer a precise localization of the degradation cause. All these works build manually their reference model before the real publication of the Web service which is used during the analysis phase. However, this reference model can not be the same for all contexts and it varies from a context to another. Especially, when we have to handle the analysis of mobile Web services. In this case, it is not judicious to use a single reference model for the analysis. In fact, the service mobility leads to a perpetual change of the hosting node which makes impossible the use of a pre-established reference model (The reference model is not the same for all nodes).

5 Conclusion

In this paper, we proposed an approach allowing the performance analysis at runtime. It automatically and dynamically generates the QoS reference values. These reference values enable the Web service state characterization. It detects performance degradation, identifies its nature and gives an accurate localization of its cause. Our approach regenerates the QoS reference values whenever the Web service changes its execution context. It differs from existing ones since it

⁵ A reference model describes the Web service acceptable state under a specific runtime context. In our work, it corresponds to the QoS reference values.

does not rely on the use of a pre-established QoS reference values and accurately identifies the degradation cause. Our analysis approach is merged within a self-healing framework developed in previous work [2]. Experiments are carried out to show the feasibility and the benefits of such approach. The QoS reference values that we generated automatically enables a rigorously performance analysis at runtime. In this work, we only deal with execution time and communication time as QoS parameters. Other QoS parameters (such as throughput, response time, availability and reliability) are out of the scope of this paper. In addition, during the training period, we do not know about the Web service state. We have to wait until the end of this period in order to check the Web service state.

As perspectives, we plan to extend our approach to include more QoS parameters while studying more Web service parameters. Also, we aim at experimenting our approach in a large scale environment like Grid'5000.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
2. Ben-Halima, R., Drira, K., Jmaiel, M.: A qos-oriented reconfigurable middleware for self-healing web services. In: *ICWS 2008: Proceedings of the 2008 IEEE International Conference on Web Services*, Beijing, China, pp. 104–111. IEEE Computer Society, Los Alamitos (2008)
3. Ben-Halima, R., Fki, E., Jmaiel, M., Drira, K.: Experiments results and large scale measurement data for web services performance assessment. In: *Proceedings of the 14th IEEE Symposium on Computers and Communications (ISCC 2009)*, Sousse, Tunisia, pp. 83–88. IEEE Computer Society, Los Alamitos (July 2009)
4. Ardissono, L., Console, L., Goy, A., Petrone, G., Picardi, C., Segnan, M., Dupré, D.T.: Towards self-diagnosing web services. In: *Proceedings of International Workshop on Self-Managed Systems and Services (SELFMAN 2005)*, Nice, France. IEEE Computer Society, Los Alamitos (2005)
5. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: Fault tolerant web service orchestration by means of diagnosis. In: Gruhn, V., Oquendo, F. (eds.) *EWSA 2006. LNCS*, vol. 4344, pp. 2–16. Springer, Heidelberg (2006)
6. Moga, A., Soos, J., Salomie, I., Dinsoreanu, M.: Adding self-healing behaviour to dynamic web service composition. In: *Proceedings of the 5th WSEAS International Conference on Data Networks, Communication and Computers*, Bucharest, Romania, pp. 206–211 (2006)
7. Pucel, X., Bocconi, S., Picardi, C., Daniele, Dupré, T., Travé-Massuyès, L.: Analyse de la diagnosticabilité des services Web. In: *Workshop Artificial Intelligence and Web Intelligence (IAWI)*, Grenoble, France (2007)
8. Tian, W., Zulkernine, F., Zebedee, J., Powley, W., Martin, P.: Architecture for an autonomic web services environment. In: *WSMDEIS*, Miami, pp. 32–44 (2005)
9. Yan, Y., Cordier, M.O., Pencole, Y., Grastien, A.: Monitoring web service networks in a model-based approach. In: *Third IEEE European Conference on Web Services, ECOWS* (2005)
10. Ross, S.M.: *Introduction to probability and statistics for engineers and scientists*. Elsevier Academic Press, Amsterdam (2004)

Business Process Compliance through Reusable Units of Compliant Processes

David Schumm¹, Oktay Turetken², Natallia Kokash³,
Amal Elgammal², Frank Leymann¹, and Willem-Jan van den Heuvel²

¹ Institute of Architecture of Application Systems (IAAS), University of Stuttgart,
Stuttgart, Germany

{Schumm,Leymann}@iaas.uni-stuttgart.de

² European Research Institute in Service Science (ERISS), Tilburg University,
Tilburg, Netherlands

{o.turetken,a.f.s.a.elgammal,w.j.a.m.vdnheuvel}@uvt.nl

³ Centrum Wiskunde & Informatica (CWI) Amsterdam, Netherlands

{Natallia.Kokash}@cwi.nl

Abstract. Compliance management is essential for ensuring that organizational business processes and supporting information systems are in accordance with a set of prescribed requirements originating from laws, regulations, and various legislative or technical documents such as Sarbanes-Oxley Act or ISO 17799. As the violation of such requirements may lead to significant punishment for an organization, compliance management should be supported at the very early stages of business process development. In this paper, we present an integrated approach to compliance management that helps process designers to adhere to compliance requirements relevant for their processes. Firstly, we introduce a conceptual model for specifying compliance requirements originating from various compliance sources. Secondly, we propose a framework for augmenting business processes with reusable fragments to ensure process compliance to certain requirements by design. Furthermore, we discuss the formalization of compliance requirements using mathematical logics and integrate the framework for process reuse with automated software verification tools.

Keywords: Compliance, Business Process Management, Process Fragment, Formal Modeling, Process Verification.

1 Introduction

In today's business environment, organizations have to cope with an increasing number of diverse and complex compliance requirements stemming from various laws, regulations, internal or external policies, business contracts etc. This increases the necessity and importance of a comprehensive compliance management solution, which must support compliance throughout all the stages of the business process life cycle. Compliance management ensures that business processes are in accordance with a set of prescribed requirements. It should be considered in three main stages: (i) compliance verification of business process models (static verification at design time), (ii) compliance monitoring of the running instances (dynamic verification at runtime),

and (iii) offline monitoring of the completed business process executions. We consider the static and dynamic verification phases as indispensable and complementary phases for ensuring and managing compliance. This is mainly because offline monitoring is a retrospective approach, which is based on the after-the-fact principle. A preventive focus is fundamentally required in order to achieve sustainability and effectiveness in compliance management.

In this paper we introduce a process-centric approach to compliance management focusing on the design time aspects where reusable units of compliant processes are utilized to augment a process with structures related to compliance. The basic idea is to combine the advantages of compliance checking based on logical formulas with a novel approach for business process reuse. Assume a reusable building block that implements a compliance requirement by means of activities and control dependency among them. We refer to such a building block as a process fragment for compliance, or compliance fragment for short. This fragment can be integrated into an existing process with the intention of making the process compliant to the corresponding compliance requirement. Thus, after the fragment has been integrated into a process, the process should actually comply with the requirement that the fragment implements. However, there is still a possibility that the process design violates the requirements as there is yet no evidence that the fragment has been integrated in the correct manner and in the correct place. The major reasons for an incorrect integration are wrong positioning, wrong concretization, and change of the original fragment design. Therefore, we propose involving rules that represent this compliance requirement in a formal manner. These rules can be checked against the modified process model using advanced methods for process verification to assure compliance.

The steps that have to be performed to provide the assurance of compliant process design are briefly described in the following: at first, a compliance expert defines and formalizes the requirements to which a particular process has to comply with. The resulting formal rules are either associated with existing compliance fragments or with new ones which are developed in cooperation of the compliance expert and a process designer. The compliance fragments which are associated with the rules are then integrated into the process by the process designer. The subsequent verification indicates if all rules could be verified, or if changes on the process are required.

The rest of this paper is organized as follows: In Section 2, we introduce a conceptual model for compliance management on which our work is based. In Section 3, we describe a common industrial scenario, which we use as a running example throughout this paper. Our approach to the development of business process models compliant by design is demonstrated in Section 4. In Section 5, we discuss related work. Finally, in Section 6, we conclude the paper and outline future work.

2 Conceptual Model

Most of the compliance requirements originate from rather generic compliance documents. Compliance requirements may emerge from different sources and can take various forms. They may originate from legislation and regulatory bodies (such as Sarbanes-Oxley and Basel II), standards and code of practices (such as: ISO 9001) and/or business partner contracts. These documents can be ambiguous and thus it is difficult to decide what exactly has to be changed in a business process in order to ensure its

compliance to these requirements. Therefore, an appropriate model for capturing and specifying compliance requirements is needed. In particular, since some parts of such documents may not be relevant for a given process, this model needs to describe compliance requirements and correlate them with business processes that must conform to them. Furthermore, since legislation and regulations tend to change over time, a link to the compliance source should be preserved. The conceptual model depicted in Fig. 1 provides the constructs to manage compliance in business processes.

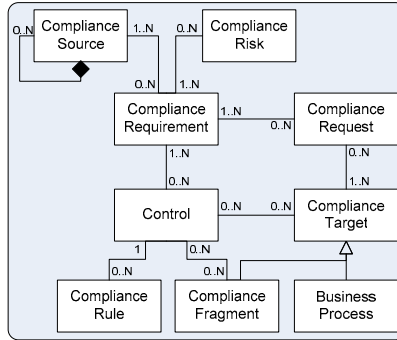


Fig. 1. Conceptual model for compliance management

A *Compliance requirement* is a constraint or assertion that results from the interpretation of the *compliance sources*, such as laws, regulations, policies, standards, contracts, etc. Failure to meet these requirements increases the likelihood of a *compliance risk* to materialize, which in turn might impair the organization's business model, reputation and financial condition. To mitigate these risks and ensure that compliance requirements are satisfied an organization defines *controls*. A control describes the restraining or directing influence to check, verify or enforce rules to satisfy one or more compliance requirements. A *Compliance rule* is an operative definition of a compliance requirement which formally describes a control. A *Compliance fragment* is a connected process structure that can be used as a reusable building block for ensuring a faster and more consistent specification and integration of compliance into a process. Compliance fragments can be used to implement a compliance rule in terms of activities and control structures. A *Compliance target* is a generic specification, such as a business process, or a compliance fragment, which is a target of compliance requirements. A user (compliance or business expert) can issue a *compliance request* to check whether a set of compliance targets conforms to a set of applicable compliance requirements. The purpose of a compliance request is to identify if and how a process can or should be changed to make it (more) compliant.

3 Running Scenario

In order to provide an illustration for the concepts introduced above and to demonstrate our approach we go over a motivating scenario. The general environment in which the scenario takes place is the e-business application domain, and particularly,

banking applications in which compliance to strict regulations and legislations is crucial. Fig. 2 depicts an excerpt from the process model for a “loan origination” process represented using the Business Process Modeling Notation (BPMN). The process starts with the customer submitting a loan request. Once the loan request is received, a credit broker checks if the customer’s banking privileges are suspended. Next, a loan threshold is calculated. If the threshold amount is less than 1M Euros, the post processing clerk checks the credit worthiness of the customer through a credit bureau service. If the threshold amount is greater than 1M Euros, the clerk supervisor is responsible for performing the same activities instead of the post processing clerk. Finally, the manager needs to approve the loan form and (in case of acceptance) send the signed form to the customer to sign it.

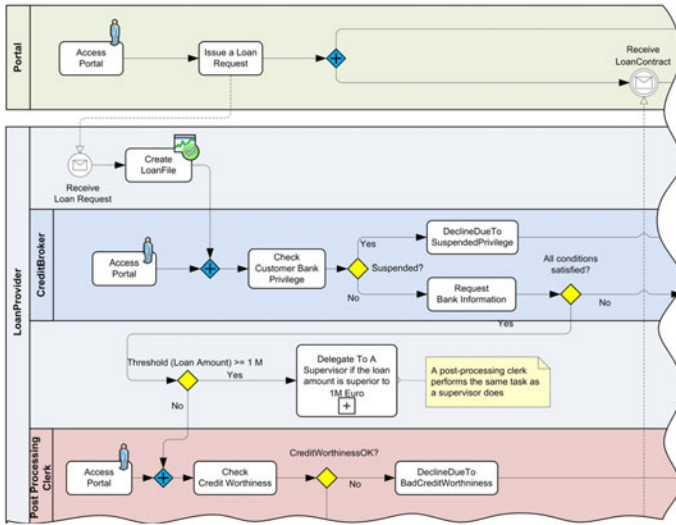


Fig. 2. An excerpt from the BPMN model of the running scenario

There are diverse compliance requirements relevant to this loan origination process, including access rights, temporal aspects, privacy and security. Table 1 gives an example of a compliance requirement regarding the appropriate segregation of duties on the loan origination process. The proposed approach will be discussed by going through this requirement and relevant controls.

Table 1. Compliance requirements relevant for the loan origination process

Control	Compliance Requirement	Comp. Risk	Comp. Source
1- Customer bank privilege check is segregated from credit worthiness check	Duties in Loan Processing should be adequately segregated	Loan granted with inadequate level of assurance	- Sarbanes-Oxley Sec. 404 - ISO 17799-10.1.3
2- If the loan request exceeds 1M Euros, the Clerk Supervisor checks the credit worthiness of the customer			
3- The branch office Manager checks whether risks are acceptable and makes the final approval of the request			

4 Ensuring Compliance of Business Processes

This section explains how compliance fragment reuse and static process verification can help us to achieve business process compliance by design.

As discussed in the previous section, a process designer is faced with the task of making a process compliant. We assume that an organization has a repository managed by compliance experts where all relevant requirements are stored in a format represented by the aforementioned conceptual model. As a proof of concept, we have implemented such a repository and call it Compliance Requirements Repository (CRR). The designer uses the CRR to find requirements that the particular process needs to adhere to. The ‘requirements search’ can be a simple keyword search done through all attributes of the requirement (including sources, risks and controls), or be based on an advanced query for expert users.

In response to the designer’s request, the CRR returns a list of all relevant requirements. If the discovered requirements have already been instantiated, i.e., formalized as discussed in Section 4.1 and available as concretized process fragments discussed in Section 4.2, they can be directly (re-)used. In this case, the concretized fragments are integrated into the process without the need to check them separately, as their compliant design has been proven before. The augmented process can then be checked against the formal rules by utilizing process verification tools for proving compliant process design (discussed in Section 4.3).

Formal rules can be associated to corresponding compliance requirements with the help of Compliance Request Language Tools (CRLT), discussed in more detail in Section 4.1. If there is one or more abstract fragment that corresponds to a particular rule, it can be concretized and customized by the process designer to fit a specific process. If such a fragment does not exist yet, it can be created and reused in the future. By integrating the fragment into the process, we ensure that the process adheres to the corresponding compliance rules. In our approach, we assume that entities (constructs) present in concrete fragments and compliance rules share unique identifiers in order to provide the correlation.

4.1 Defining and Formalizing Compliance Requirements

Compliance requirement specification language should be based on concepts derived from formal logics to enable automated verification of compliance targets against these requirements. Deontic logic (e.g. [19]) and temporal logic (e.g. [14]) families have been intensively discussed in the literature as a basis for such a specification language. In our framework, we mainly rely on temporal logic for representing compliance rules. Our choice is justified by the fact that system property specification using temporal logics is a mature field supported by efficient verification tools tested and applied in practice for over 20 years. Among the formalisms within temporal logic family, we prefer Linear Temporal Logic (LTL) [16] to Computational Tree Logic (CTL) mainly due to its simplicity, intuitiveness and compositionality of reasoning [22].

One of the main problems of the temporal logic family in general is that logical formulas are difficult to write and understand for users. The notion of *property specification patterns* (Dwyer's property patterns) was introduced in [6] as high-level abstractions of frequently used logical formulas. These patterns assist users in understanding and defining formal specifications. In addition to the original patterns introduced in [6], we have developed *Compliance Patterns* to capture recurring patterns in the compliance context. Table 2 shows such patterns applied to the running scenario. The first control is implemented using the newly introduced *SegregatedFrom* pattern that captures the typical compliance requirement which mandates segregation of duties among different roles and actors. In LTL, G , F , U correspond to the temporal operators 'always', 'eventually', and 'until' respectively. ' G ' denotes that formula f must be true in all the states of the business process model. ' F ' indicates that formula f will be true at some state in the future. ' U ' denotes that if at some state in the future the second formula g will be true, then the first formula f must be true in all the subsequent states. For example, the LTL representation of ' P LeadsTo Q ' is ' $G(P \rightarrow F(Q))$ ', which can be read as: If P is true, then in the future Q should occur.

Table 2. Compliance rules for the examples from the loan origination process

Control	Pattern	Comp. Rules in LTL
1- <i>Customer bank privilege check is segregated from credit worthiness check</i>	CheckCustomerBankPrivilege SegregatedFrom Check Credit Worthiness	$G((\text{CheckCustomerBankPrivilege.Role}(\text{Role1}) \rightarrow G!(\text{CheckCredit Worthiness.Role}(\text{Role1})))$
2 If the <i>loan request</i> exceeds 1M, the <i>Clerk Supervisor checks the credit worthiness</i> of the customer	$((\text{CreateLoanFile.Threshold} \geq 1\text{M}) \text{ LeadsTo CheckCredit Worthiness.Role("Supervisor")})$	$G((\text{CreateLoanFile.Threshold} \geq 1\text{M}) \rightarrow F(\text{CheckCredit Worthiness.Role}(\text{Supervisor})))$
3- The branch office <i>Manager</i> checks whether risks are acceptable and makes the final <i>approval</i> of the request.	$((\text{JudgeHighRiskLoan AND Approved} = \text{"Yes"}) \text{ Preceeds SignOfficiallyLoanContract.Role('Manager')})$ AND $((\text{JudgeHighRiskLoan AND Approved} = \text{"No"}) \text{ Preceeds DeclineDueToHighRisk('Manager')})$	$G((\text{JudgeHighRiskLoan } A \text{ Approved} = \text{"Yes"}) \vee (\neg \text{SignOfficiallyLoanContract.Role('Manager')} U (\text{JudgeHighRiskLoan } A \text{ Approved} = \text{"Yes"}))) A$ $G((\text{JudgeHighRiskLoan } A \text{ Approved} = \text{"No"}) \vee (\neg \text{DeclineDueToHighRisk.Role('Manager')} U (\text{JudgeHighRiskLoan } A \text{ Approved} = \text{"No"})))$

We are currently implementing an environment¹ as a part of a tool-suite for business process compliance management. The prototype is a web-based environment, which also incorporates stand-alone tools for building graphical representation of requirements using patterns. The ongoing integration with Process Verification toolkit (see Section 4.3) for process verification is achieved through a group of asynchronous web service calls. BPMN or BPEL representations of compliance targets (i.e. process models) and relevant formal compliance rules specified in LTL are transferred to the Process Verification toolkit. The toolkit returns the verification result, listing the rules that have been checked and whether they are satisfied or not. Fig. 3 presents one of the user interfaces from the implementation reflecting how the results of the compliance check are communicated to the business or compliance expert. The user interface exemplifies the case that the first control given in Table 2 is violated.

¹ CRLT: Compliance Request Language Tools, <http://eriss.uvt.nl/compas>

Compliance Check (UI-120)							
Compliance Target			Check Result	Degree of Compliance			
Loan Processing			Violations Exist! Please Check Details	84%			
Compliance Profile Details							
Control ID	C.Requirement	C.CONTROL	C.Risk	C.Source	Weight	Result	
1.1	Duties in Loan Processing are adequately segregated	Activity "Check Customer Bank Privilege" is segregated from "Check Credit Worthiness"	- Loan granted with inadequate level of assurance - Low probability of repayment	ISO 17799 - 10.1.3, Internal Policy 101, SOX	5 - Very High	Activities performed by the same role... Check Details	
ID	Description	Statement	Type	Design Time	Run Time	Satisfied?	Result Desc. / Remedy
1	Check if activity exists	F (CheckCustomerBankPrivilege)	LTL	✓		✓	
2	Check if activity exists	F (CheckCreditWorthiness)	LTL	✓		✓	
3	Check if activities follow each other	G((CheckCustomerBankPrivilege OR (CheckCreditWorthiness UNTIL CheckCustomerBankPrivilege)))	LTL	✓		✓	
4	Check if activities follow each other	G((CheckCustomerBankPrivilege THEN F (CheckCreditWorthiness)))	LTL	✓		✓	
5	Check if activities are assigned to different roles	G((CheckCustomerBankPrivilege.Role(Role1) THEN G((CheckCreditWorthiness.Role(Role1))))	LTL	✓		✗	Activities performed by the same role This rule can be checked when Runtime information is available
6	Check if activities are assigned to different actors	G((CheckCustomerBankPrivilege.Actor(Actor1) THEN G((CheckCreditWorthiness.Actor(Actor1))))	LTL	✓	✓	?	
1.2	Duties in Loan Processing are adequately segregated	If the loan request credit does not exceed 1 million EURO, the Post Processing Clerk checks the credit worthiness of the customer.	Customer initial credit worthiness check is segregated from post check, which is performed by supervisor role	- Loan granted with inadequate level of assurance - Low probability of repayment	3 - Moderate	✓	

Fig. 3. A user interface with compliance requirements identified for the running scenario

4.2 Compliance Fragments

Process fragments provide a lightweight approach for reusable process structures. In [20] we introduced process fragments for compliance (abbreviated as compliance fragments) as a means to realize compliance requirements within business processes (e.g., based on BPMN) and workflows (e.g., based on BPEL) respectively. In order to utilize this concept for a fast and consistent augmentation of processes with compliance a library of such reusable compliance fragments has to be built up by the bank or a consulting agency in our running scenario. This leads to the first phase in the management life cycle of compliance fragments, which is identification and design. In this phase either reusable process structures related to compliance are identified within an existing process and extracted there from, or they are designed from scratch. For instance, the fragment for approval shown in Fig. 4 could have been extracted from the bank’s quality assurance process. To ease reuse the extracted or designed fragment needs to be rendered somewhat abstractly, i.e. static values have to be parameterized, activities need to be generalized and process-specific parts have to be removed (Fig. 4a). A compliance fragment may have multiple points for integration into a process. We call those points fragment entries and fragment exits.

The next phase in the fragment life cycle is storage and retrieval. For this phase we are developing a fragment repository [7] that efficiently supports versioned storage and retrieval. In our example the process designer would query this repository and find (and retrieve) the abstracted fragment for approval. This fragment can then be integrated into the loan approval process in order to realize the compliance requirement. During integration the fragment has to be concretized, i.e. parameters have to be set and the fragment has to be customized for the particular process in which it is applied (see Fig. 4b). Therefore, checking an abstract fragment against concrete rules has little advantages, but it is possible (and useful) to check a concrete rule against a concrete fragment.

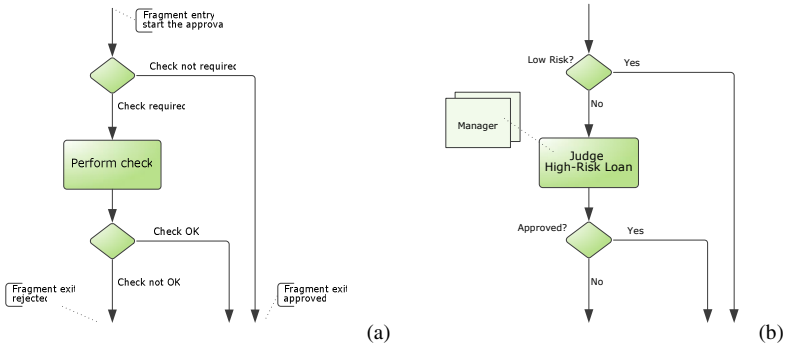


Fig. 4. (a) Abstracted process fragment for approval; (b) Concretized fragment

4.3 Process Verification

To achieve compliance-by-design, we aim at the detection of the violation of compliance rules in design and implementation of compliance fragments and business processes. To accomplish this goal, we automatically convert a compliance fragment or a business process (either in BPMN, BPEL or UML) to its formal representation in Reo [5]. Reo [2] is a graphical channel based coordination language that enables the modeling of complex behavioral protocols using a small set of channel types with predefined behavior. The application of Reo to business process modeling resembles that of Petri nets. Intuitively, an asynchronous FIFO channel with a buffer of capacity one in Reo corresponds to a place in a classical Petri net, while the notion of Petri net transition is generalized and can be composed of multiple synchronous channels. This enables the propagation of synchrony across Reo networks and helps us to model business processes in a more concise and compositional manner. Fig. 5 shows a Reo counterpart for the approval fragment. In this model, an abstract activity *Perform check* is represented as a buffer while conditional gateways correspond to nodes with outgoing filter channels.

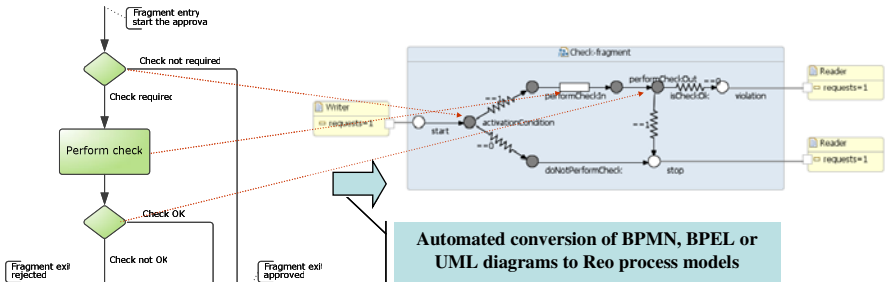


Fig. 5. Process formalization: Abstracted fragment for approval is converted to Reo

Eclipse Coordination Tools (ECT) [3], a supporting framework for behavioral service-based process modeling in Reo, consists of a set of integrated plug-ins that provide the functionality for converting, editing, animating, annotating, simulating and model checking formalized process models. Since high-level models often do not contain all the information necessary for the automated process verification, we assume that ECT is used by a technical specialist to refine the process models that are passed for compliance verification. The imported process models and fragments need to be refined and the compliance rules have to be transformed to a format which can be accepted by a specific model checking tool chosen to verify a given property.

Currently, three model checking tools are supported by ECT, namely Vereify [23], mCRL2 [15] and PRISM [18]. Vereify is a tool that can check properties specified in LTL and CTL-like logics and can be used for control flow analysis. Among its advantages are its compatibility with the compliance rule language discussed in Section 4.1 and the ability to visualize counterexamples in a user-friendly manner by showing them on Reo models using flash animations. Detailed examples of using this tool to process compliance analysis, in particular verification of temporal constraints on process control flow and segregation of duties, can be found in [11]. However, data specification supported by Vereify currently is not elaborate enough to enable the verification of data-dependent compliance rules. Such rules can be analyzed with the help of the mCRL2 toolset. The mCRL2 specification language and the corresponding toolset were developed by the University of Eindhoven and represent a powerful means for large-scale system verification. ECT includes a plug-in for automatic generation of mCRL2 specifications from Reo process models [12], annotated, if necessary, with data and time constraints. For example, Fig. 6 shows the model of the dataflow in a concretized process fragment, where the input data domain is described by a sort $el(activated: Bool, amount: Nat)$ which indicates whether the approval is activated and provides the requested loan amount. Data constraints in a format understandable by mCRL2 (e.g., $amount(e1(d)) > 1000000$) are used as annotations to graphical Reo models and specify process dataflow branching conditions.

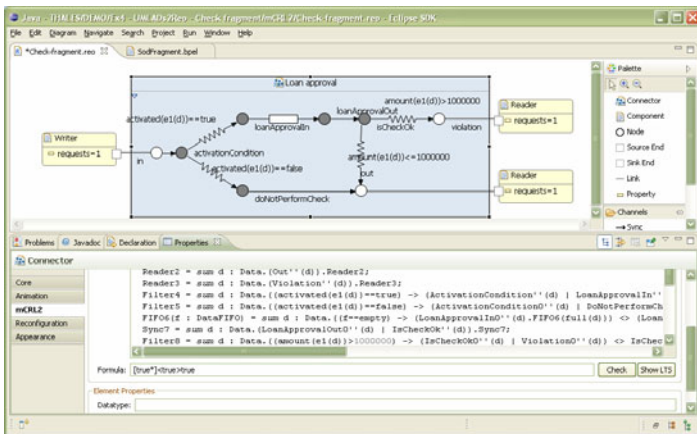


Fig. 6. Formal process model refinement: Concretized fragment for approval is annotated with information about input data domain and dataflow

Such a model can be used for explicit state space generation or model checking against properties specified in a variant of μ -calculus. This format subsumes temporal logics LTL and CTL and allows us to formally express compliance rules with time and data-aware conditions. For example, a compliance rule “*if a requested loan amount is higher than 1M, a manager authorization must be obtained*” corresponds to the following formula:

$$[\text{true} * \exists \text{amount} : \mathbb{N} . \text{LoanRequest}(\text{amount}) \wedge (\text{amount} > 1000000)] \mu X [\overline{\text{authorization}}] X$$

This formula literarily states that for a loan request with the amount exceeding 1M Euro the authorization activity is unavoidable. Finally, the PRISM model checker is used for the verification of probabilistic and quantitative properties of a Reo process model. More detailed study of the application of this tool to compliance analysis constitutes our future work. Apart from process model checking, formalized Reo process models can be used for model-based test generation [21]. In this case, generated tests may assure the compliance of an actual system implementation rather than just the designed model. For example, in the aforementioned scenario at least four test instances should be generated, with and without activated check conditions and loan requests with two amounts: one exceeding 1M, and one not exceeding 1M. Model-based test generation tools such as JTorX are compatible with the generated mCRL2 specifications and can be easily employed in our framework. After the verification, formalized models and model checking results are saved in a repository for further reuse and process reengineering. Counterexamples found by the model checking tools and generated tests that the system did not pass can help the designer to understand why the property violation occurs in the composed process (e.g., detect fragments that are implemented in a wrong way, point out where the wrong integration points or incorrect placements of fragments are).

5 Related Work

Temporal logic has been used intensively in the literature for the formal specification of compliance requirements, key work examples are: [1], [4], [8], [9], [14] and [10]. The authors of [14] proposed a static compliance-checking framework that includes various model transformations. Compliance requirements are modeled using the graphical Business Property Specification Language (BPSL) tool where graphically represented compliance requirements are automatically transformed to LTL formulas. Next, the NuSMV2 model checker is used to verify the compliance. The study in [1] utilized π -Logic to formally represent compliance requirements. In addition, a toolkit has been developed to implement the proposed approach (HAL toolkit).

On the other hand, business process models are abstractly modeled. If the abstract business process model is compliant, a BPEL process equivalent to the abstract representation can be automatically generated. The study in [4] utilized past LTL (PLTL) where properties about the past can be represented. However, sequential compliance requirements are just considered. On the other hand, the study in [16] has utilized the original pattern based system adapted in this paper. They considered only runtime

compliance monitoring though. The study in [10] employed the original pattern specification system used in this paper for the verification of service compositions. In addition, they have introduced the logical composition of patterns using Boolean logical operators. The correctness of pattern composition has also been proved. Composite patterns enable the definition of complex properties in terms of property patterns. Composite patterns can also be used for the specification of complex compliance requirements. Furthermore, authors in [9] have extended the original property pattern system to capture time-related property specifications, so that real-time requirements can be represented via patterns. E.g. activity *A* must always be followed by activity *B* within *k* time units.

Concerning reuse in business processes many concepts have been proposed so far. Besides the well-known approaches for reuse such as sub processes or business rules, more and more lightweight approaches are proposed. For instance, in decentralized process modeling multiple people are involved, each of them having local know-how. Each of the involved designers can model a particular aspect of a process as process fragment, i.e. as an incomplete but connected process structure. These fragments are later composed to a complete process model [13]. Although there is a significant number of works in each of these areas, there is, to the best of our knowledge, currently no approach that combines the advantages of formal languages and compliance checking based on logical formulas with an approach for business process reuse. Here we discussed a concept that demonstrates how these different fields can be combined to support compliance management in business processes.

6 Conclusion and Outlook

In this paper, we presented a framework for design-time business process compliance management. In particular, we introduced a conceptual model for specifying compliance requirements and discussed how these requirements can be stored and processed. The main contribution of this paper is an approach that combines the formalization of compliance requirements, their automated verification for a given process and a novel approach for process reuse. This combination enables a consistent augmentation of business processes with process structures that implement relevant compliance requirements and supports the development of compliant-by-design software applications. By going through a scenario, we briefly demonstrated the concepts and the proposed approach. We also demonstrated the core functionalities of the tools utilized, each representing a part of an ongoing effort on the development of a comprehensive tool-suite for business process compliance management.

Although the formal language introduced in this paper can be used to formalize compliance requirements of diverse types, only those requirements relevant to the control flow of the business processes can be tackled powerfully with compliance fragments. For instance, a locative requirement that demands a certain set of rules on data storage requires a different approach as it refers to database applications rather than activities and control structures. The approach presented in this paper can be seen as one piece of the puzzle in an overall solution to managing compliance.

Acknowledgements

This work is a part of the research project COMPAS (www.compas-ict.eu) which is funded by the European commission, contract no. FP7-215175. Many thanks go to Huy Tran for the development of the process model of the loan origination scenario.

References

1. Abouzaid, F., Mullins, J.: A Calculus for Generation, Verification, and Refinement of BPEL Specifications. In: Proc. of the WWV 2007, pp. 43–68 (2007)
2. Arbab, F.: Reo: A Channel-based Coordination Model for Component Composition. *Mathematical Structures in Computer Science* 14, 329–366 (2004)
3. Arbab, F., Koehler, C., Maraiakar, Z., Moon, Y., Proenca, J.: Modeling, Testing and Executing Reo Connectors with the Eclipse Coordination Tools. In: Tool Demo Session at FACS 2008 (2008)
4. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
5. Changizi, B., Kokash, N., Arbab, F.: A Unified Toolset for Business Process Model Formalization. In: Proc. of the Int. Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA 2010) (2010)
6. Dwyer, M., Avrunin, G., Corbett, J.: Property Specification Patterns for Finite-State Verification. In: Int. Workshop on Formal Methods on Software Practice, pp. 7–15 (1998)
7. Fragmento - Fragment-oriented Repository. Online Documentation (2010), <http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm>
8. Giblin, C., Liu, A., Muller, S., Pfitzmann, B., Zhou, X.: Regulations Expressed As Logical Models. In: Proc of the 18th Int. Annual Conf. on Legal Knowledge and Information Systems (2005)
9. Gruhn, V., Laue, R.: Specification Patterns for Time-Related Properties. In: 12th Int'l Symposium on Temporal Representation and Reasoning, USA, pp. 198–191 (2005)
10. Yu, J., Manh, T., Han, J., Jin, Y.: Pattern-Based Property Specification and Verification for Service Composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)
11. Kokash, N., Arbab, F.: Formal Behavioral Modeling and Compliance Analysis for Service-Oriented Systems. In: de Boer, F.S., Bonsangue, M.M., Madeline, E. (eds.) FMCO 2008. LNCS, vol. 5751, pp. 21–41. Springer, Heidelberg (2009)
12. Kokash, N., Krause, C., de Vink, E.: Data-aware design and verification of service composition with Reo and mCRL2. In: Proc. of the SAC 2010. ACM Press, New York (2010)
13. Eberle, H., Unger, T., Leymann, F.: Process Fragments. In: Proc. of the 17th Int. Conference on Cooperative Information Systems (CoopIS). Springer, Heidelberg (2009)
14. Liu, Y., Muller, S., Xu, K.: A Static Compliance-Checking Framework for Business Process Models. *IBM Systems Journal* 46 (2007)
15. mCRL2 toolset, <http://www.mcr12.org>
16. Namiri, K., Stojanovic, N.: Pattern-based Design and Validation of Business Process Compliance, pp. 59–76. Springer, Heidelberg (2007)
17. Pnueli, A.: The Temporal Logic of Programs, In: Proc. of the 18th IEEE Symposium on Foundations of Computer Science, Providence, pp. 46–57 (1977)

18. Probabilistic model checker, <http://www.prismodelchecker.org/>
19. Sadiq, S., Governatori, G., Naimiri, K.: Modeling Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
20. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. In: Proc. of the MKWI 2010, Universitätsverlag Göttingen (2010)
21. Tretmans, J.: Model Based Testing with Labelled Transition Systems. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST 2008. LNCS, vol. 4949, pp. 1–38. Springer, Heidelberg (2008)
22. Vardi, M.: Branching vs. Linear Time: Final Showdown. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 1–22. Springer, Heidelberg (2001)
23. Vereofy model checking tool, <http://www.vereofy.de/>

An Approach to Enable Replacement of SOAP Services and REST Services in Lightweight Processes

Teodoro De Giorgio, Gianluca Ripa, and Maurilio Zuccalà

CEFRIEL - ICT Institute Politecnico di Milano

Via Renato Fucini 2, 20133 Milano, Italy

{teodoro.degiorgio,gianluca.ripa,maurilio.zuccala}@cefriel.it

<http://www.cefriel.it>

Abstract. In the last few years, several Web APIs implementing the REST principles were created. Process modellers often need to use both SOAP and REST services within a single process. In this paper we present our experience in using MicroWSMO for supporting dynamic replacement of SOAP and REST services inside a service composition, even in presence of syntactic mismatches between service interfaces. We also present a running prototype we implemented in order to apply our approach to lightweight processes execution.

Keywords: Service Adaptation, SOAP Services, REST Services, Semantic Annotations, Service Mediation, Service Composition, Service-Oriented Architectures, Lightweight Processes.

1 Introduction

In a previous paper [1] we exploited SAWSDL [10] descriptions to support adaptation in the context of service composition. Some sort of adaptation is usually needed when a service to be invoked is not available and therefore should be replaced with an equivalent one: a number of syntactic mismatches can take place in this case, mainly at the interface level (i.e., the two services differ in the names and parameters of the operations exposed) or at the protocol level (i.e., the two services differ in the order in which operations must be invoked), thus preventing the further execution of the overall service composition. In order to address this problem, in [2] the authors defined a set of basic mapping functions. Since in real cases they often observe combination of mismatches, also the mapping functions can be combined to provide a solution to complex mismatches. These basic mapping functions are defined as follows:

- ParameterMapping: maps abstract service input data on concrete service input data.
- ReturnParameterMapping: maps abstract service output data on concrete service output data.

- OperationMapping: maps abstract service operations on concrete service operations.
- StateMapping: maps an abstract service state on a concrete service state.
- TransitionMapping: maps an abstract service transition on a concrete service transition.

Basic mapping functions can be combined in adaptation scripts, defined in a domain specific language. The language is composed of rules structured in two parts:

- A mismatch definition part that specifies the type of the mismatch to be solved by the rule, and contains two subelements: input, specifying the elements of the abstract service that show the given mismatch, and mapping, specifying the elements of the concrete service the input elements have to be mapped on.
- A mapping function part that contains the name of the function to be used to solve the mismatch.

For an in depth treatment of mismatches, mapping functions and mapping language see [2]. This solution usually requires manual definition of proper mapping functions at design time, thus encumbering the work of system integrators in charge of setting up service compositions. In [1] we showed that if service interfaces are semantically annotated, adaptive and dynamic replacement of services in a composition can be achieved in a more effective way. The idea behind the approach is that some ontology already exists and that adding semantic annotations to the service description is an easier task for the service provider than manually creating the mapping script for the service integrator. If two service providers use the same ontology for annotating their service descriptions, the definition of the mapping scripts can be automated.

We extended our approach described in [1] for supporting REST services. Indeed, in the last few years, several Web APIs implementing the REST principles were created as a lightweight alternative to SOAP. Since in our previous approach we exploited SAWSDL as a semantic description language for WSDL based services, we investigated some equivalent language in the context of REST services. In [7] MicroWSMO, a semantic annotation mechanism for RESTful Web services is described. It is based on a microformat called hRESTS (HTML for RESTful Services) for machine-readable descriptions of Web APIs, and it is backed by a simple service model which serves as a rough equivalent to WSDL. Our extended approach uses service descriptions written in MicroWSMO and SAWSDL format in order to enable for substituting a SOAP service with a REST one (and viceversa) as implementation of an activity of a process.

The approach was implemented in a prototype that is part of the Execution Engine developed in the SOA4All project [13]. SOA4All concentrates on empowering non-technical users to do simple IT modelling and development work in the area of service construction and composition. In SOA4All is introduced a Lightweight Process Modelling Language (LPML) for supporting users in the lightweight modelling based on three design principles:

1. abstraction of process models;
2. the use of semantic annotations;
3. context-awareness.

LPML reuses concepts mainly defined in BPMN [12] and BPEL [15] and add new modelling concepts for supporting these principles and to support users with advanced guidance during the modelling activities. The output generated by SOA4All tools starting from the business process graphically described by the users is a lightweight process described using BPEL [15] plus the following attributes extensions used to support the dynamic binding and service substitution at runtime:

- replacementCondition: this attribute represents an element in a taxonomy that defines the set of pre-defined replacement conditions. The replacement conditions are the situations in which the engine will try to substitute a service with another one.
- selectionCriteria: The selection criteria is the criteria applied by the engine for selecting a substitute service from a list of alternatives. The selection criteria are for example: concepts like the price of something, response time, service rating, and other.
- alternativeServiceList: this element lists the services that can be used as alternatives in the service substitution. is a list of alternativeService subelements:
- alternativeService: this element is an URI that points to the service description specification.

In this paper we focus on the execution of these lightweight processes. The Execution Engine component delegated to do this is able to analyze the attributes generated and to execute the process that contains in the alternative service list REST and SOAP services. The component is able to replace at runtime a SOAP service with a REST one, and vice versa. In this paper we illustrate the results of our research and the new solution developed for using SAWSDL and MicroWSMO for enabling the execution of self-adaptive processes.

The paper is organized as follows. Section 2 summarizes the problem. Section 3 presents our approach. Section 4 presents a comparison with respect to other work in the same field. Section 5 draws some conclusions.

2 The Problem

In many situations, process modellers need to use both SOAP and REST services within a single process. For the sake of clarity, in this section we refer to the specific sample process depicted in Fig. 1. This process is created by a modeller who needs to construct a weather forecast service that returns a weather forecast given the name of a city as input. The modeller, who can be supported by tools offered also from SOA4All project, searches for a weather forecast service.

She finds a WSDL/SOAP service that implements the required functionality and that is available for use with the required level of quality. The only problem

is that the service requires as input the geographical coordinates of a location and not the city name. Thus, the modeller searches for a service that, given a city name, returns its geographical coordinates. She finds such service implemented as a REST service. Combining these two services in a BPEL [15], LPML [14] or another process language, the modeller is able to fulfill the needed service.

The result is the sample process in Fig. 1, composed by two activities bounded at design time to two services, e.g., a REST service that gives access to a world-wide geographical database with a search function, and a SOAP service that provides weather forecasts. As depicted in Fig. 1, the sample process:

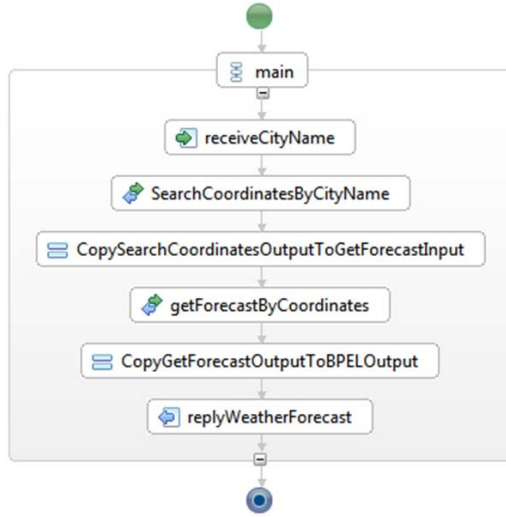


Fig. 1. Sample Process

1. Receives the city name string as input;
2. Invokes the search operation passing as input the name of the city, and obtaining as output the coordinates of the city, i.e., latitude and longitude;
3. The output of the previous activity is passed to the following activity as input;
4. Invokes the ForecastService, service passing as input the coordinates of the city, i.e., latitude and longitude, and obtaining as result the weather forecast for the city;
5. The output of the previous activity is passed to the following activity as input;
6. Returns the weather forecast.

During the modelling phase, the modeller discovers also some possible alternative services (that implement the same functionalities but with a lower level of quality) that can be used in order to successfully execute the composition in case a fault occurs during the invocation of one of the selected services. The Sample Process scenario proposed in this section requires to compose SOAP

and REST services in a single process definition. Aim of our approach is to allow the replacement of a failing SOAP service with a REST one or vice versa. Furthermore, such heterogeneous composition, in our scenario, can be built in a semi-automatic way without the need of heavy technical skills. Indeed we assume that the modeller is not an IT expert but a business domain expert.

3 Our Approach

The problem described in the previous section raises some questions:

- How to establish the semantic compatibility between REST services, and between REST services and SOAP services (given that we already solved the problem of the compatibility between SOAP services)?
- How to adapt REST services and SOAP services at runtime?

In order to support invocation of REST services and to enable service replacement and service message adaptation between SOAP and REST services, we use MicroWSMO service description for REST services. MicroWSMO, described in detail in [7], is a semantic annotation mechanism for REST services, based on a microformat called hRESTS (HTML for RESTful Services) for machine-readable descriptions of Web APIs. MicroWSMO adds SAWSDL-like annotations to hRESTS service descriptions. The SOA4All Studio provides SWEET [4], a tool used for annotating Web APIs in MicroWSMO. Starting from the SOA4All results and from the analysis of REST services and SOAP services we derived the information required for describing the characteristics of a REST service needed to allow the invocation and the replacement at runtime:

- The service description URL
- The URI template for each operation
- The name of the operations
- A model reference for each operation that refers to the corresponding service operation concept
- The name of the method (i.e., GET or POST)
- The list of the input parameters with the following information:
 - Parameter name
 - Parameter type (as XML Schema datatypes)
 - A model reference URI for each parameter
 - (If necessary) a reference to a lifting/lowering Schema for each parameter
- The output parameter with the following information:
 - MIME-type (default text/XML)
 - If the mime type is text/XML the annotated XML schema, otherwise the model reference URI
 - (If necessary) a reference to a lifting/lowering Schema

It is worth to be noted that the lifting and lowering schemas, that are part of both SAWSDL and MicroWSMO specifications, have to be used only in case of complex mapping logic, where some computation is necessary for translating

from one concept to another. Thus in our approach we rely mainly on the model-Reference attribute that we consider as a more lightweight means for annotating the service descriptions.

This minimum set of data is in line with the Lightweight RESTful Service Model described in [7] and [4].

This way of annotating can be used also for enhancing the discovery of a service by means of semantic search engines. In our experience, the opposite, i.e. using the annotation commonly created for enhancing the discovery for automating the service invocation/substitution at runtime, is very often not possible.

An example of a two fragments of service annotations for a REST and a SOAP service are given below. The following RDF fragment is the MicroWSMO description of the input parameter “latitude” of a service operation for a REST service.

```
<wsl:hasInputMessage>
  <wsl:Message>
    <sawsdl:modelReference
      rdf:resource="http://www.soa4all.eu/ontology/execution/parameters#Required"/>
    <sawsdl:modelReference
      rdf:resource="http://www.w3.org/TR/xmlschema-2/#double"/>
    <sawsdl:modelReference
      rdf:resource="http://www.soa4all.eu/ontology/execution/gps#Latitude"/>
    <rdfs:label xml:lang="en">lat</rdfs:label>
  </wsl:Message>
</wsl:hasInputMessage>
```

Below the same parameter is described in SAWSDL.

```
<xs:complexType>
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="latitude" type="xs:double"
      sawsdl:modelReference="http://www.soa4all.eu/ontology/execution/gps#Latitude"/>
    ...
  </xs:sequence>
</xs:complexType>
```

The parameters in the MicroWSMO and in the SAWSDL descriptions are compatible. Thus, the Execution Engine starting from a set of annotations of the two service descriptions similar to the one in the example above, is able to establish the semantic compatibility and to automatically provide the service substitution in case of necessity.

Now we describe what happens at runtime. We supposed that a process is deployed and all the artefacts required to execute a process and to provide the self-adaptation at runtime are generated. We assume also that the process of execution starts and the executor of processes sends the input required for the invocation of the services expected from the process.

During the execution of the process the role of the Execution Engine is to check the result of the service invocation and in case of necessity adapt the request/response. The Service Adapter is the component delegated to adapt the messages of the real services provided by third party. During the execution of the process, the Execution Engine is able to bind to the expected service suggested in input and invoke it. As shown in Fig. 2, it is possible that the response of the Service A invoked is a fault message. In this case, by the analysis of the selection criteria and based on the alternative service list, Service B is selected

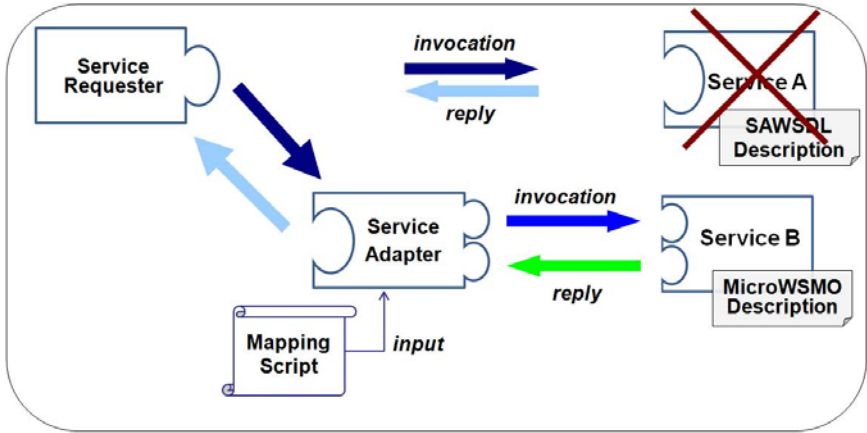


Fig. 2. Runtime Service Adaptation

as the new service to be invoked. In the case of the scenario in the sequence, the expected service is a WSDL/SOAP service and the alternative service is a REST service. The role of the Service Adapter in this second invocation is crucial for the success of the invocation. It consists of the following steps:

1. Receive from the Execution Engine the expected request and the reference to the alternative service;
2. Select the mapping script related to the adaptation from the expected service to the alternative service;
3. Execute the script and adapt the request to the alternative service using the mapping script selected;
4. Invoke the alternative service;
5. Receive the response from the alternative service;
6. Adapt the response to the expected service using the mapping script selected;
7. Send the adapted response to the Execution Engine.

Then, it is possible to continue the execution of the process, because the service replacement is transparent to the final user and it is internal to the Execution Engine.

4 Related Work

At the time of the writing of this paper, other initiatives are facing a similar problem. An example is Apache ODE [11] that proposes a REST variant of the BPEL invoke activity that replaces the attributes partnerLink/operation with resource and method. This new Apache ODE feature is not yet implemented and it is in the state of a proposal, subject to changes.

A similar attempt is the one described in [8] where the REST2SOAP framework is described. REST2SOAP aims at integrating SOAP services and RESTful

services in order to create a BPEL service that combines SOAP, REST services and user interfaces simultaneously. REST2SOAP leverages WADL specification, and can wrap RESTful services into SOAP services semi-automatically.

In [9] the author presents a hybrid approach that makes use of SOAP services and REST services. The system contains both a BPEL engine and a REST orchestration engine. A big workflow is divided into several sub-workflows according to their intrinsic architectural style. Then, each resulted sub-workflow is handled in a native way. SOAP-based orchestration is conducted in the BPEL engine, and REST services are orchestrated in the REST orchestration engine. Invocations help assemble the whole business process. Interactions between heterogeneous services are minimized.

All the above mentioned approaches enable the creation of orchestrations that combine SOAP and REST services but do not enable the dynamic substitution of each other at runtime.

In [3] an approach for replacement of services inside a service composition that focuses on conversational REST services is presented. The approach uses model checking techniques for automatically identifying the interaction protocols mapping. This approach can be combined with ours for enabling the mapping script generator to generate rules (i.e., state mapping and transition mapping rules) able to solve also protocol level mismatches.

In [5] an extension to the WS-BPEL standard process modeling language is proposed to support the native composition of RESTful services. The interaction primitives (GET, POST, PUT, and DELETE) stemming from the REST uniform interface principle can be directly used from within a BPEL process as new service invocation activities. The same authors, in [6], applied the notion of composition to RESTful services and derived a set of language features that are required by composition languages for RESTful services: dynamic late binding, dynamic typing, content-type negotiation, state inspection, and compliance with the uniform interface principle. They also presented a case-study using the JOpera visual composition language.

5 Conclusions

In this paper we described how to use SAWSDL and MicroWSMO for supporting the dynamic replacement of REST and SOAP services inside a service composition. The approach developed can help different groups of end users to build new services and processes according to their specific needs in a lightweight manner, since it hides a lot of technical details related to hybrid and heterogenous orchestration, and to self-adaptiveness. The approach heavily relies on semantic annotation approaches for the generation of runtime artefacts starting from a process description. There are many common scenarios for which this is applicable and semantic annotations already exists. In the examples proposed, this annotation is realized, using the SOA4All project tools.

The approach and the prototype support both SOAP services and REST services described by using SAWSDL and MicroWSMO, and it is able to replace at runtime

a SOAP services with a REST one, and vice versa. Furthermore, we are working for enabling the mapping script generator to generate rules (i.e., state mapping and transition mapping rules) able to solve also protocol level mismatches.

At the time of writing we are facing the problem of supporting services that work with content types that are not XML-based.

Acknowledgments

Parts of this work were sponsored by the European Commission in course of FP7 project SOA4All. The opinions expressed represent the authors' point of view and not necessarily the one of the projects participants or of the EC. We would like to thank Alessandro Marasco who is contributing to the implementation of the prototype.

References

1. Cavallaro, L., Ripa, G., Zuccalà, M.: Adapting Service Requests to Actual Service Interfaces through Semantic Annotations. In: PESOS Workshop. IEEE Computer Society Press, Vancouver (2009)
2. Cavallaro, L., Di Nitto, E.: An Approach to Adapt Service Requests to Actual Service Interfaces. In: SEAMS Workshops. ACM Press, Leipzig (2008)
3. 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.) ICSSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 159–174. Springer, Heidelberg (2009)
4. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services. In: OTM Workshops. Springer, Vilamoura (2009)
5. Pautasso, C.: RESTful Web service composition with BPEL for REST. *Data and Knowledge Engineering Journal* 68, 851–866 (2009)
6. Pautasso, C.: Composing RESTful services with JOpera. In: Bergel, A., Fabry, J. (eds.) *Software Composition*. LNCS, vol. 5634, pp. 142–159. Springer, Heidelberg (2009)
7. Kopecky, J., Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTfulWeb Services. In: *Web Intelligence*. IEEE, Los Alamitos (2008)
8. Peng, Y.-Y., Ma, S.-P., Lee, J.: REST2SOAP: A framework to integrate SOAP services and RESTful services. In: *Service-Oriented Computing and Applications SOCA* (2009)
9. He, K.: Integration and orchestration of heterogeneous services. In: *Joint Conferences on Pervasive Computing JCPC* (2009)
10. W3C: Semantic Annotations for WSDL and XML Schema. W3C Recommendation (2007), <http://www.w3.org/TR/sawSDL/>
11. Apache ODE: Project Website, <http://ode.apache.org>
12. Business Process Modeling Notation BPMN: Project Website, <http://www.bpmn.org>
13. SOA4All: Project Website, <http://www.soa4all.eu>
14. SOA4All project deliverable: D6.3.2 Advanced Specification of Lightweight, Context-aware Process Modelling Language, <http://www.soa4all.eu/file-upload.html?func=showdown&id=127>
15. OASIS Web Services Business Process Execution Engine (BPEL): TC Website, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

Context, Quality and Relevance: Dependencies and Impacts on RESTful Web Services Design^{*}

Hong-Linh Truong¹, Schahram Dustdar¹, Andrea Maurino², and Marco Comerio²

¹ Distributed Systems Group, Vienna University of Technology, Austria
{truong,dustdar}@infosys.tuwien.ac.at

² Department of Informatics, Systems and Communication
University of Milano - Bicocca, Italy
{maurino,comerio}@disco.unimib.it

Abstract. While several techniques have been introduced for specifying and acquiring context and quality information associated with Web services, they consider such information representing the whole Web services. However, accessing to context and quality of data resources provided by Web services is crucial. This is particularly relevant for data-intensive Web services of which the context and quality of data resources will strongly impact on the service development and composition. In this paper we contribute an analysis of relationships among context, quality, and relevance, as well as their impact on the design and composition of Web services, in particular at the data resource level. Then, we propose several techniques to incorporate context and quality descriptions into REST APIs and RESTful services publishing. By implementing these features, RESTful Web services could allow the consumer to specify and query context and quality information associated with services and data resources, thus fostering the provision of high relevant data resources.

1 Introduction

Web services (WS) have fostered the access of data resources in the Internet scale, e.g. several datasets are available in the UN DATA API project^[1] or Infochimps^[2]. When discovering, composing and executing Web services, we consider them and their provided data resources. However, currently non-functional descriptions of Web services mainly represent the service as a whole and they provide only marginal information at the data resource level. Several models and techniques [1][2] have been proposed to extend standard descriptions of a Web service with information regarding context (e.g., conditions for the service usage) and quality of service (e.g., response time, availability). Vice versa, there is not the same support for the specification of context and quality of data of data resources that can be accessed through the Web services, such as (i) under which situation the data can be used and (ii) the accuracy of the data. In particular, this problem is relevant for data-intensive RESTful Web services which rely on simple

^{*} The research in this paper is partially funded by the EU under the FP7 Commius project and by the SAS Institute srl (Grant Carlo Grandi).

¹ <http://www.undata-api.org/>

² <http://api.infochimps.com/>

principle and operation patterns and present several advantages over SOAP-based Web services [3]. RESTful Web services represent a practical way to access data resources on the Web but currently lack techniques for acquiring quality and context information associated with data resources and services, thus it is difficult to improve the relevance of the outcome of RESTful Web services.

The outcome of a service is considered relevant when the provided results match the consumer's purpose. Naturally, the concept of *relevance* is dependent on the consumer's need. To provide relevant results to the consumer, a service must be able to handle (i) context and quality information associated with the consumer's need and (ii) context and quality information associated with the provided resources. Based on that, any service composition should utilize context and quality descriptions from both sides - the provider and the consumer - in order to support the concept of relevance. This principle is not new, as researchers have investigated several ways to improve the relevance of output of different systems, such as information retrieval systems [4,5]. However, there is a lack in understanding the dependency among context, quality and relevance in data resource-oriented Web services. Furthermore, often the design of Web services lacks guidelines for supporting context and quality aspects at the data resource level. This problem strongly hinders the development of algorithms and techniques to improve the relevance of results provided by data resource-oriented services.

To tackle the above-mentioned issue, in this paper we focus on data-intensive services with which the relevance of the service output is judged mainly, besides the functionality of the data, on the context and quality in which the data can be utilized. We contribute an analysis of relationships among context, quality, and relevance, as well as their impact on the design and composition of Web services. Then, since the REST model is increasingly used for data-intensive services, we propose several techniques to incorporate context and quality descriptions into REST APIs and RESTful services publishing to allow the consumer to specify and query context and quality information associated with services and resources, thus fostering the evaluation of the relevance degrees of RESTful services and data resources.

The rest of this paper is organized as follows. Section 2 presents our motivating scenario. We present the role of context, quality, and relevance in Section 3. Current supports of RESTful services with respect to the context, quality, and relevance are discussed in Section 4. Techniques to enhance the support of quality and context in RESTful designs are presented in Section 5. Related work is discussed in Section 6, followed by a conclusion and future work in Section 7.

2 Motivating Scenario

To examine the importance of context and quality information related to Web services and data resources during service design, composition and execution, let us consider a simple composite service that supports the search of news and images. The composite service includes three RESTful Web services named Yahoo! Boss News

³ Since we focus on services providing data, in this paper the two terms "resource" and "data resource" are used interchangeably.

Search⁴, Google News Search⁵, and Flickr⁶. Our composition is written using Yahoo! Pipes⁷ which invokes these Web services using their REST APIs. Given a query of key words from the user, while the composition invokes services which return many data resources (e.g., figures and web news), many of them might not be relevant to the current user's context and expected quality.

Let us imagine that the user of the composite service is a doctor and she would like to perform a health research in Haiti due to the recent earthquake in January 2010⁸. In her research, she would like to find recent news and high-qualified images, but free-of charge. By entering "Haiti" into the composite service, the composite service is able to find news and images. Let us consider the first case in which the composite service does not understand the context of the search or does not understand how to specify context and quality parameters in service APIs. The corresponding composition is shown in Figure 11. In the second case, let us assume that the composite service understands context of the user or the composite service is able to obtain the user's context and quality requirements (this can be achieved based on specific ways of how the composition execution platform interacts with the user). Furthermore, the composite service knows how to specify context- and quality-related parameters in REST APIs of its composed services. Using the context and quality description expected by the user, the composite service can utilize these descriptions in the invocation of Web services and the processing of results to the user. Table 11 presents possible mappings from the user's requirements to parameters of REST APIs of corresponding services in the composition and Figures 2 shows the composition that utilizes appropriate context and quality parameters. We have observed that, in the first case, several results are not relevant to the user's request. Between two invocations, the results from Yahoo! Boss are similar as the quality parameter used does not affect the results of the five most relevant news. However, in the second invocation, results from Google News Search have a higher *irrelevance* as news are not related to Haiti and images from Flickr are more *relevant* based on understanding the *context* of the user and on the utilization of *quality* description of data resources provided by services.

The scenario shows the importance of understanding and utilizing context and quality information in order to provide relevant results to the consumer. However, this also presents many existing issues in current RESTful service design. First, how can the composition developer recognize context and quality description associated with services? For example, does Flickr service support data quality metrics or not? If not, maybe Picasa service⁹ should be used. Second, how context and quality parameters can be mapped and passed to a service via REST APIs (e.g., when asking services to provide high accuracy images)? Third, how to obtain context and quality description associated with services and data resources so that further activities can be made

⁴ http://developer.yahoo.com/search/boss/boss_guide/

⁵ http://code.google.com/apis/ajaxsearch/documentation/reference.html#_fonje_news

⁶ <http://www.flickr.com/services/api/>

⁷ <http://pipes.yahoo.com>

⁸ http://en.wikipedia.org/wiki/2010_Haiti_earthquake

⁹ <http://picasaweb.google.com/>

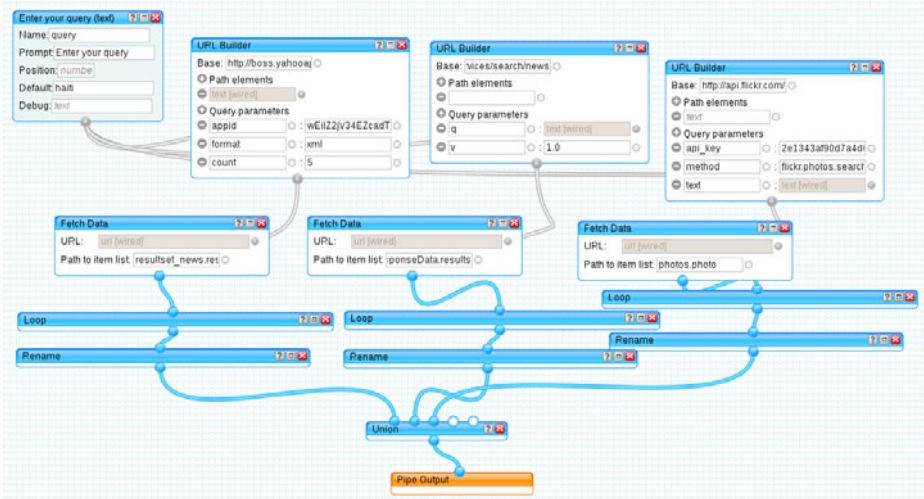


Fig. 1. A composite service for searching news and images

Table 1. Possible mappings of context and quality requirements to REST APIs. The mapping is not necessary suited to the provider’s definition but it is based on the view of a service composition developer.

Type	Yahoo!Boss News Search	Google News Search	Flickr
Context	age=2w (the news in the last two weeks)	contacts=all (search in user’s contacts);topic=m (search on health news)	licenses=1,2,3 (types of licenses are Attribution-NonCommercial-NoDerivs, Attribution-NonCommercial, and Attribution-NonCommercial-ShareAlike Licenses); tag=heath,medical (for medical topic), min_taken_date=2010-01-10 (for recent images)
Quality	age=2w (the timeliness of the news)	scoring (the higher the score is, the more relevant the news is)	accuracy=3 (images with the country level), option b for selecting high resolution images

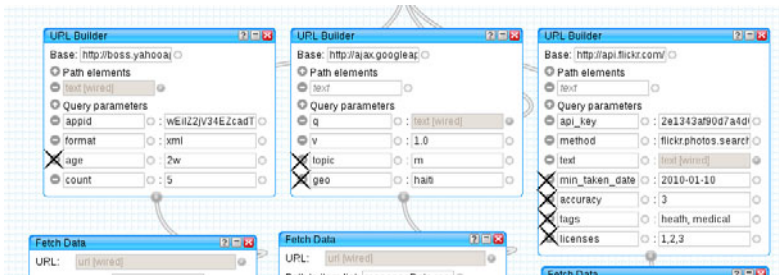


Fig. 2. Adding context and quality parameters (marked by X) during the service composition

in the service composition (e.g., filtering relevant images and news records based on their quality description)? When solutions for such questions are provided, the service composer can adapt and improve the quality of compositions substantially. However, this cannot be achieved without understanding the impact of having quality and context information at the resource level on that of services and composite services.

The two examples show that missing context and quality information leads to irrelevant results. Context and quality are not only associated with consumers but also with services and data resources. Being able to obtain context and quality information at the resource level will help to provide highly relevant results to consumers. However, so far, little attention has been spent to support this issue.

3 Context, Quality, and Relevance Dependencies

In principle, a composite service utilizes Web services that provide mechanisms to retrieve and manipulate resources. A consumer utilizes a composite service in order to retrieve the (most expected relevant) data resources. To be able to provide relevant data resources to the consumer, the composite service, Web services and data resources may implement interfaces to support the processing, publishing and discovery of possible context and quality information. In the execution model of the composite services, Web services and data resources, two flows of context and quality information exist. First, context and quality information can be required and defined based on the *consumer* \rightarrow *composite service* \rightarrow *Web service* \rightarrow *data resource* flow. Second, such information can be aggregated and changed based on the *data resource* \rightarrow *Web service* \rightarrow *composite service* \rightarrow *consumer* flow. Along these flows, several operations can be applied to context and quality information in order to support context- and quality-aware services and to provide high relevant results. We will concentrate on the second flow in this paper.

Figure 3 shows that context and quality are associated with composite services, Web services and data resources but they are originated from two different sources: the provider/integrator and the consumer. The consumer's context and quality descriptions define what an individual consumer need. The context and quality descriptions from

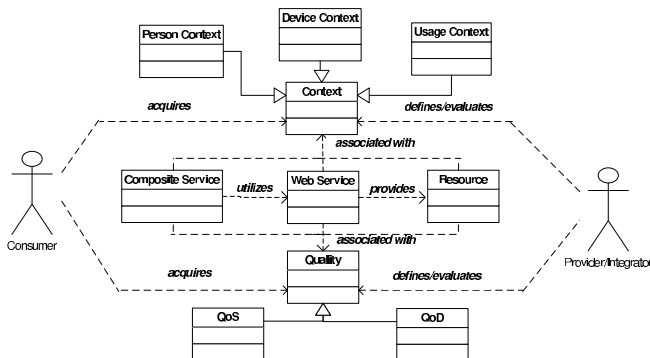


Fig. 3. Context and quality associated with composite services, Web services, and data resources

Table 2. The impact of the lack of context and quality information on the relevance of the (composite) service’s output

Type	Context	Quality	Impact on Relevance
Resource	not specified	not specified	it is difficult to select relevant data resources and to provide general description of context and quality for the service managing the resource. The service has to build its own quality and context description. This could be very difficult if the service developer is not the resource provider (e.g., the developer aggregates data resources in the Web).
Resource	specified	not specified	it is not clear if the resource will meet the consumer quality description. The service has to use its own knowledge to determine the quality of the resource.
Resource	not specified	specified	it will not be clear whether the data resource can be used. The service can only rely on its own knowledge in order to determine if a resource is suitable for a particular context.
Web Service	not specified	not specified	it is not sure if the Web service can be used. Even if the Web service can be used, it is not sure if the resource provided can be used. The composite service has to implement its own service selection mechanisms.
Web Service	specified	not specified	the Web service can be used but the composite service is not sure about the quality of the service and the resource. The composite service has to determine the quality of the resources at its side using its own knowledge.
Web Service	not specified	specified	it is not clear if the Web service is suitable for the context. The composite service has to determine the context of the service by its own knowledge.

the provider describe how services and data resources fit to generic requirements. As shown in Figure 3 many types of context and quality can be specified. The context can be specialized in *person context* (i.e., the context associated with consumer/provider), *device context* (i.e., the context of the device used to access the composite service, Web service or data resource) and *usage context* (i.e., the context in which the composite service, Web service or resource is supposed to be used). The quality can be specialized in *quality of data (QoD)* (e.g., the accuracy of data resources) and *quality of service (QoS)* (e.g., response time of the Web service).

Although several techniques have been introduced for specifying and acquiring context and quality information for services, they focus on the service level. Therefore, context and quality associated with data resources are also crucial in order to provide high relevant results. Table 2 describes how the lack of quality and context information at the resource level (and relevant to service level) impacts on the implementation of composite services able to provide high relevant results. Overall, this lack of information forces the service or the composite service to develop several mechanisms to compensate the missing information and these mechanisms might rely on only the knowledge of the service composition developer. This requires us to incorporate and develop models and APIs for handling context and quality information. The description, specification, and evaluation of context and quality information and the utilization of such information for improving the relevance of the service output require several research activities.

4 Quality and Context Support in Current RESTful WS Design

Having a detailed analysis of the dependencies among context, quality, and relevance, we examine how such information is coupled and supported in the current RESTful service design. The REST architectural model assumes that a resource can be created,

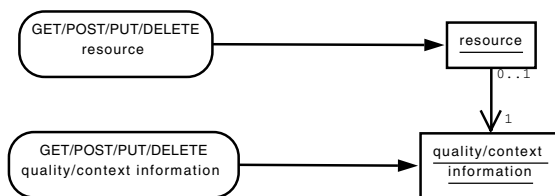


Fig. 4. Separate handling for resources and their context/quality information

updated, modified, and deleted by using four operations named *GET*, *POST*, *PUT*, and *DELETE*. Such operations are not designed to handle quality and context descriptions associated with resources. Of course, in principle, one can use these operations to handle context and quality associated with a data resource. However, so far we are not aware of this in the design of RESTful services. One approach is to consider the quality and context information as a new resource and access them separate from the resource they characterize (see Figure 4). However, in this case, the consumer will have difficulties to understand and manage the relationship between the resource and its quality/context description via separate resource paths in REST APIs. Therefore, to support the consumer to specify and obtain context and quality description of resources, similar operations on a resource should be used for handling the resource (such as *GET* the resource) and for handling context/quality information (such as *GET* the data quality).

Currently, at the service level, the context and quality information associated with RESTful services and parameters for specifying context and quality requirements cannot easily be distinguished from the REST APIs, if not impossible. In fact, many RESTful services do not provide such information. In RESTful service descriptions, such as based on WADL^[10] or ATOM^[11], there is no context and quality information. At the REST APIs level, there is no convention for specifying and obtaining context and quality information. Therefore, the RESTful service design should provide context and quality description exchange protocols among the interactions of service compositions, Web services and resources. In the following section, we present our approach in detail.

5 Enhancing Context and Quality Support in RESTful WS Design

In this section, we propose some extensions to the design and implementation of RESTful services in order to support quality and context aspects. We present our guidelines for this purpose by means of an experimental service which is a RESTful Web service to provide data from the Google Flu Trend^[12]. In our experimental service, named *googleFluTrend*, the list of countries is considered as a data resource and each country is also considered as a data resource^[13].

¹⁰ <https://wadl.dev.java.net/>

¹¹ <http://www.ietf.org/rfc/rfc5023.txt>

¹² <http://www.google.org/flutrends/>

¹³ Due to the space limit, we provide detailed supporting materials at

<http://www.infosys.tuwien.ac.at/prototyp/SOD1/quacore>

5.1 Representations for Context and Quality

Several works have proposed context and quality descriptions defined using, for example, XML, RDF and OWL [6,7,8]. Therefore, they can be reused to define representations for RESTful service. Since XML and JSON are the most popular way of describing the requests, responses and descriptions of RESTful services, we use them to represent the context and quality information.

Our quality and context representations are based on our work in context and concerns for data as a service [9]. Listings 1 presents an excerpt of a custom quality description in JSON (readers could refer to our supporting materials for examples of custom quality and context models). Given the representations of context and quality for RESTful services, we propose the following techniques for coupling context and quality information with RESTful Web services.

```

{
  "crq.qod": {
    "crq.uptodateness": "up to dateness",
    "crq.objectivity": "objectivity",
    "crq.freeoferror": "free of error",
    "crq.consistency": "consistency",
    "crq.dataelementcompleteness": "data element completeness",
    "crq.datasetcompleteness": "data set completeness",
    "crq.domainspecificqod": "URI specifying domain specific info"
  },
  "crq.qos":
  {
    "crq.responsetime": "response time",
    "crq.latency": "latency",
    "crq.capability": "capability",
    "crq.reliability": "reliability",
    "crq.availability": "availability"
  }
}

```

Listing 1. Simplified JSON-based quality representation

5.2 Coupling Context and Quality with RESTful Services

Given their representations, context and quality information need to be associated with resources and services and to be published so that the service composition can utilize the information. This association is performed by the service provider and the context and quality information reflect the conformity of the services and resources provided by the services. Obviously, such information can be published using existing approaches in Web service information management, e.g., using Web service registries. However, these approaches have not been designed for publishing information characterizing RESTful resources and services. To follow the principle of and the widely-accepted resource description for RESTful Web services, we illustrate techniques to publish context and quality information based on WADL.

WADL allows us to specify the following main elements: application, resource, method, request, response, and representation. Context and quality information can be associated with all these elements with the exception of

representation. From the service provider view, WADL elements can be associated with context and quality description for the following purposes: (i) publishing information for service discovery: this is related to `application` and `resource`, (ii) allowing service consumers to specifying inputs of context and quality in REST APIs: this applies to `request` and `method`, and (iii) allowing service consumers to obtain context and quality information associated with resources by using REST APIs: this applies to the `response` and `method`.

Publishing quality and context information for service discovery: In order to publish the context and quality description associated with a service, we need to provide two types of information (i.e., the schemas and the information according to the schemas) and to associate them with RESTful Web services. Because the context and quality descriptions are considered as extra documents about RESTful Web services, we should map these schemas and descriptions by using optional elements in WADL. To utilize existing constructs of WADL, we propose the following guidelines. The mapping of schemas of context and quality can be performed by using the `grammars` element of WADL. This element allows to include external schemas using a sub-element `include`. The `representation` element can be used to describe published, static quality and context descriptions. Furthermore, to distinguish different schemas and types of information, we can use the `doc` element. Table 3 describes how to associate quality and context information with the WADL of the service.

Table 3. Describing context and quality information in WADL

Element	Usage	Example
<code>grammars/include</code>	specify context and quality schemas	<code><include href="crq-quality.xsd"></code>
<code>grammars/include/doc</code>	specify the type of schema. We propose to use the title to describe the name of schema	<code><doc title="Quality"></code>
<code>representation</code>	specify the published, static quality and context information in service description and specify the representation of context and quality information associated with REST APIs (e.g., GET and POST)	<code><representation id="QoD" element="crqQuality:crd.qod" mediaType="application/json"></code>
<code>representation/doc</code>	specify the content of static, published quality or context information.	<code><doc title="QoDDescription">...</doc></code>

Based on the above-mentioned descriptions, several activities can be performed. For example, if a service consumer wants to search services based on quality and context descriptions, the consumer can utilize the information specified in the element `representation/doc` by filtering relevant `representation` elements based on the `title` element. Consumer-side code generation tools can utilize the `representation`, `request` and `response` elements to generate codes for handling quality and context information.

Example: A WADL description is available at <http://www.infosys.tuwien.ac.at/prototyp/SOD1/quaCore/examples/GoogleFluTrend-v0.2.wadl> for our experimental service. The `wadl2java` tool¹⁴ can be used to generate

¹⁴ <https://wadl.dev.java.net/>

functions `getAsQoD(String crqQod, String crqContext)`, and `getAsService Context(String crqQod, String crqContext)` to obtain context and quality information.

Specifying Context and Quality Parameters in REST APIs: The specification of context and quality descriptions in REST APIs can be done by using query parameters, which can be built based on the name described in the request of the WADL. By utilizing query parameters the form of `crq.metricName=value`, where `crq.metricName` is the name of context and quality metrics specified in context and quality representations, one can indicate context and quality aspects in REST APIs.

Example: To select resources with the minimum accuracy 0.5 in Europe, we can use the request `GET/resource?crq.accuracy="0.5"&crq.location="'Europe''`. Given the input request of context and quality information, the service can utilize these information to select the right resources. Optionally, the response can also include quality and context related information together with the requested resource.

Obtaining Context and Data Quality in REST APIs: Context and data quality descriptions should also be obtained for services and resources without obtaining the services and resources. To this end, we propose to use query parameters. For this purpose, context and quality parameters are specified but without any values. By following this convention, we can assume that the consumer requires only context and quality information. For example, a request like `GET/resource?crq.qod` would return only the quality of data of the requested resource. Using this way, the service consumer can query only the context and quality information before deciding which resources it should access. An advantage is that resources will not be accessed if their quality is not guaranteed. However, in cases of context- and quality-guaranteed resources, two invocations are needed to retrieve the resources.

Example: while the resource containing all countries can be obtained by using `curl http://.../resources/googleFluTrends`, to obtain the quality of this resource the parameter `crq.qod` without any value can be used as follows:

```
$curl http://localhost:8080/restfuldesign/resources/googleFluTrends?crq.qod
{"crq.qod": {
  "crq.datasetcompleteness": 0.10256410256410256,
  "crq.consistency": 1
}}
```

A request of `GET/googleFluTrends?crq.dataelementcompleteness=0.9` will return only resources that have the minimum completeness 0.9. For example, the data resource `Austria` is returned as its quality of data is:

```
{"crq.qod": {
  "crq.dataelementcompleteness": 0.9,
  "crq.consistency": 1
}}
```

6 Related Work

The majority of related work with respect to the role of context, quality, and relevance is in the focus of the information retrieval. Knight and Burn have proposed the relevance as a data/information quality metric and a contextual metric [10]. Batini and colleagues have also presented the fact that the relevance is also used as a data quality metric for data provided by databases [7]. However, these discussions are limited to database aspects. The relevance term defined in [7,10] can be used to indicate on the relevance of the resource from the provider view. Lachica and colleagues have proposed a framework that uses quality and relevance to rank information in information retrieval systems [11]. In their work, relevance is context dependent and is characterized by four types of context information. Thus, in some senses, they also presented the relationship among context, relevance and quality. However, their work is not focused on Web services and their relevance term is independent on quality information.

In Web services, user context and quality of service have been long considered as valuable source of information for supporting Web service design, discovery and composition [12,13]. Most related works using quality and context information in service-oriented computing can be divided into three classes: (i) approaches to enhanced Web service design, such as different ways to add non-functional parameters to Web services in [14]; (ii) techniques to increase the relevance of the result of Web service discovery and selection, such as the discovery of resources based on WS-Policy specifications using an external middleware in [15] and the NFP-based hybrid approach to Web service ranking in [16]; (iii) approaches to improve the relevance of information offered by the services using context information, such as users and their interaction with a service [12], user experiences [17], and context models for personalized Web services [13].

However, existing works in the above-mentioned classes mainly deal with QoS and context information at the whole service level and they do not cover the quality of the information of resources offered by the services. Furthermore, these works mainly support the service/resource discovery. Different from them, we focus on mechanisms to inquire combined context and quality metrics associated with resources. In [18], the quality of mashups and how to evaluate it are discussed. We believe that our approach is a complement work as we provide mechanisms for specifying and accessing quality of data associated with resources. Such mechanisms would simplify the retrieval of quality metrics of resources in order to evaluate data quality metrics for mashups.

7 Conclusion and Future Work

In this paper, we have analyzed the importance of context and quality support in service design, discovery, composition and execution, with a focus on the data resource level. We have discussed the impact of the lack of context and quality information on the guarantee of highly relevant response to the consumer. Although our study is generic, to prove our concepts, we have examined limitations of current RESTful Web service design w.r.t. context and quality information management. To overcome these limitations, we propose several steps in RESTful service design to allow data resources, Web services and composite services to specify and obtain context and quality information.

Our guidelines focus only on mechanisms for inquiring context and quality metrics associated with resources and for requesting and retrieving these resources using these associated metrics. Thus, semantic mapping of context and quality parameters within an individual (composite) service or consumer, is not addressed. The way to specify context and quality descriptions in REST APIs is just one method that will be compared with other ways, such as using HTTP headers.

References

1. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: SawSDL: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing* 11(6), 60–67 (2007)
2. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework. In: *WWW 2004: Proceedings of the 13th international conference on World Wide Web*, pp. 553–562. ACM, New York (2004)
3. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. “big” web services: making the right architectural decision. In: *WWW 2008: Proceeding of the 17th international conference on World Wide Web*, pp. 805–814. ACM, New York (2008)
4. Maglaughlin, K., Sonnenwald, D.: User perspectives on relevance criteria: a comparison among relevant, partially relevant, and not-relevant judgements. *J. Am. Soc. Inf. Sci. Technol.* 53(5), 327–342 (2002)
5. Greisdorf, H.: Relevance: An interdisciplinary and information science perspective. *Informing Science* 3, 67–71 (2000)
6. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. *SIGMOD Record* 36(4), 19–26 (2007)
7. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. *ACM Comput. Surv.* 41(3) (2009)
8. Bettini, C., Brdiczka, O., Henriksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* (2009) (article in press)
9. Truong, H.L., Dustdar, S.: On Analyzing and Specifying Concerns for Data as a Service. In: *Proceedings of The 4th IEEE Asia-Pacific Services Computing Conference, APSCC 2009, Singapore, December 7-11*. IEEE, Los Alamitos (2009)
10. Knight, S.A., Burn, J.: Developing a framework for assessing information quality on the world wide web. *Informing Science* 8, 159–172 (2005)
11. Lachica, R., Karabeg, D., Rudan, S.: Quality, relevance and importance in information retrieval with fuzzy semantic networks. In: *Proc. of the fourth international conference on Topic Maps Research and Applications, TMRA* (2008)
12. Pernici, B. (ed.): *Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility*. Springer, New York (2006)
13. Mamar, Z., Mostefaoui, S.K., Mahmoud, Q.H.: Context for personalized web services. In: *Proc. of the 38th Annual Hawaii International Conference on System Sciences (HICSS)*, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2005)
14. Ortiz, G., Núñez, J.H., Clemente, P.J.: How to deal with non-functional properties in web service development. In: Lowe, D.G., Gaedke, M. (eds.) *ICWE 2005*. LNCS, vol. 3579, pp. 98–103. Springer, Heidelberg (2005)
15. Mietzner, R., van Lessen, T., Wiese, A., Wieland, M., Karastoyanova, D., Leymann, F.: Virtualizing services and resources with probus: The ws-policy-aware service and resource bus. In: *IEEE International Conference on Web Services, ICWS 2009*, pp. 617–624 (2009)

16. Comerio, M., De Paoli, F., Palmonari, M.: Effective and flexible nfp-based ranking of web services. In: Proc. of ICSOC/ServiceWave 2009, Stockholm, Sweden, pp. 546–560 (2009)
17. Kokash, N., Birukou, A., D’Andrea, V.: Web service discovery based on past user experience. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 95–107. Springer, Heidelberg (2007)
18. Cappiello, C., Daniel, F., Matera, M.: A quality model for mashup components. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 236–250. Springer, Heidelberg (2009)

Quality-Based Recommendations for Mashup Composition

Matteo Picozzi, Marta Rodolfi, Cinzia Cappiello, and Maristella Matera

DEI - Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy
{matteo.picozzi,marta.rodolfi}@mail.polimi.it,
{cappiell,matera}@elet.polimi.it

Abstract. When composing mashups, the selection of suitable services is mainly based on functional requirements and does not consider the quality of the single services. In this paper, we show that the quality of component services can drive the production of recommendations that can help building quality mashups. We capitalize on a quality model for mashup services and discuss the concept of *mashability*, a multi-dimension quality property that expresses the capability of a component to maximize the quality of a mashup, and the concept of *role-based composition quality*, i.e., the quality of mashup compositions weighted according to specific roles that the composed services play within the mashup. We then show how such concepts can enable the production of quality-based recommendations for the mashup design.

1 Introduction

Web mashups are a new generation of applications that support the “composition” of applications starting from contents and services provided by third parties and made available on the Web [6,17]. Mashups were initially conceived in the context of the consumer Web, as a way for end-users to create their own applications. More recently, enterprise mashups started to emerge, as instruments for the average (i.e., not necessarily technically-skilled) users to easily and quickly create *situational* applications that can serve, even for a short period, to support their decisional processes [12]. Both the previous contexts are characterized by the involvement of end users. However, with very few exceptions, so far the research in the mashup field has focused on the technical issues, such as formats for data exchange, interoperability, etc., while very little attention has been devoted to easing the development process. In many cases, indeed, mashup composition still results into the manual programming of the service integration.

Recently, some approaches have been proposed to ease the mashup composition task. On the one hand, some tools try to reduce the efforts required to program the service composition, by proposing high-level abstractions, also sustained by visual easy-to-use notations [9,5]. On the other hand, some other approaches focus on the production of recommendations about the component services and the composition patterns to be adopted [8,16]. The approach we

discuss in this paper goes into the direction of easing the mashup composition by means of recommendations; its characterizing feature is that it takes into account quality, both component services quality and the overall mashup quality, as a key factor to drive the production of recommendations.

We capitalize on a previous model for the quality of mashup components [2], and identify how the quality of single components, expressed along several quality attributes, can be used to guide the selection of services and compositions. We in particular discuss the concept of *mashability*, a multi-dimension quality property that expresses the capability of a component to maximize the quality of the final mashup, and the concept of *role-based composition quality*, i.e., the quality of mashup compositions weighted according to specific roles that the composed services play within the mashup. We then show how these concepts can support the production of recommendations within a “quality-aware” mashup development.

The paper is organized as follows. In Section 2 we illustrate the typical scenario for mashup development, with the purpose of highlighting different quality issues, and the consequent need for quality-based recommendations. We then illustrate our model for mashup quality: we summarize our previous model [2] for the quality of mashup components (Section 3), and define the new perspectives of *mashability* (Section 4) and *role-based composition quality* (Section 5), which are the basis for the generation of quality-aware recommendations. Inspired by the recommendation model proposed in [8], in Section 6 we then show how our quality model can support the production of recommendations. We finally describe the most relevant related works (Section 7), and draw our conclusions (Section 8).

2 Quality Issues in the Mashup Development Process

There is a lack of proposals for the quality of mashups. In a sense, this is because, being Web applications, mashups can be mainly characterized by the external quality-in-use perspective, which is exhaustively covered by the huge research on Web application usability [13]. We however believe that, beyond quality in use, other issues must be considered, which are strictly related to the activities that characterize the mashup development process.

The typical scenario for mashup development spans from the production of single *mashup components* to the integration of selected components into a final *mashup composition*. Given the nature of mashups as applications integrating other resources, throughout the whole process the quality of the component resources and the adopted composition patterns play a fundamental role. We identify three stages in which the quality of component services comes into play.

The component developer creates component services for mashups. We assume that developers correctly implement the service functionality, taking into account well-known principles, best practices and methodologies for guaranteeing the internal quality of the code. However, when used in a mashup composition, component services can be selected by considering especially some external properties. Since our aim is to investigate the quality of the final mashups, this is the

perspective we are interested more, which is related to aspects such as the architectural style (e.g., SOAP services vs. RESTful services vs. widget APIs), the adopted programming language (e.g., client side such as JavaScript vs. server side such as Ruby), the data representation (e.g., XML vs. JSON), the component operability and interoperability (e.g., the multiplicity of APIs targeting different technologies). Such external aspects indeed affect the “appeal” of the component from the mashup composer perspective. The component developer should try to maximize them, and should also make these quality properties visible for the mashup composers, who can therefore base on them the choice of components. The component developer therefore builds the component having in mind the quality properties to be maximized. She can also document such quality properties by means of suitable component descriptors, possibly complementing a documentation for the component use (at least, ideally).

When composing a mashup, *mashup composers discover components*, directly from the Web or from repositories accessed from the adopted mashup tool, taking into account the fitness of each component for its purpose within the mashup (i.e., its functional requirements), but also the complexity of its technological properties (e.g., a simple programming API, languages and data formats enhancing operability and interoperability), as well as the richness and completeness of the provided data. The quality of single services can drive their selection, so enabling the composer to select components based on the provided quality.

At the end of the service selection phase, when a first committed composition is ready, quality assessment can take advantage of the defined composition model, to identify the role and importance that component services play within the composition. The quality of mashups can thus be revisited and quantified as an aggregation of the quality of the single components, weighted however on the basis of the component role and the adopted composition patterns. Therefore, once a composition model is available, some recommendations can be issued, about adding further components, or about alternative *similar* compositions able to increase the quality of the final mashup.

Addressing the previous quality concerns requires the definition of sound models, first of all focusing on the quality of components as stand-alone ingredients, but also on their attitude to generate quality mashups when combined with other components.

3 Quality of Mashup Components

Publishing mashup components through APIs or services hides their internal details and gives more importance to their external properties [18]. In line with this black-box view, in [2] we proposed a quality model for mashup components that privileges properties of the component APIs - this is indeed the perspective that is most relevant to the mashup composer or the mashup user. The model is based on both our own experience with the development of components and mashups [17], and on experimental evidence gathered by analyzing data from programmableweb.com [2]. It is organized along three main dimensions recalling

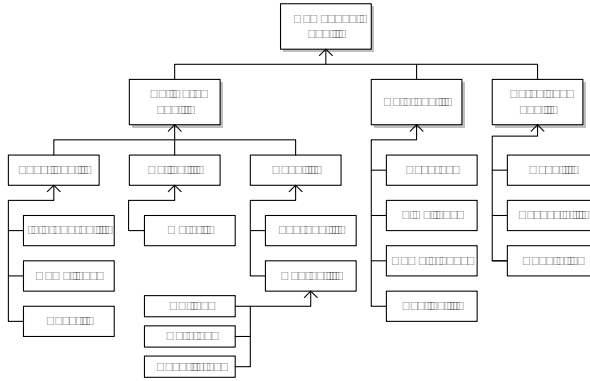


Fig. 1. The quality model for mashup components [2]

the traditional stratification of Web applications into data, application logic and presentation layer:

- *Data quality* focuses on the suitability of the data provided by the component in terms of accuracy, completeness, timeliness, and availability.
- *API quality* refers to software characteristics that can be evaluated directly on the component API. We split API quality into *functionality*, *reliability*, and *API usability*.
- *Presentation quality* addresses the user experience, with attributes such as *presentation usability*, *accessibility*, and *reputation*.

Figure 1 summarizes the quality dimensions, their attributes, and the finer-grained characteristics addressed by our model. For more details on the associated metrics the reader is deferred to [2].

4 Mashability

The previous model focuses on single components. *Mashability* is still a property of single components, but it expresses the measure in which the quality characteristics of a component would improve the mashup being created. It therefore puts the quality of each single component into the dynamic perspective of the mashup composition, and can be seen as the combination of *compatibility* and *aggregated quality*.

Compatibility estimates whether the combination of one component with those already included in a composition is possible, based on:

- *Technology compatibility*, to assess whether two or more components are compatible from the point of view of the adopted protocols, languages and data formats;
- *Syntactic compatibility*, to check the type compatibility between the input/output parameters exposed by the component services;

- *Semantic compatibility*, to check whether input/output parameters and involved operations belong to the same or similar semantic categories, assuming that the syntactic compatibility is satisfied.

Compatibility is not properly an expression of the quality of the final mashup; it is however a source of recommendations for the mashup composer about the selection of compatible component services, so as a correct composition is produced. The notion of *aggregated quality* is therefore needed, as an estimation of the final mashup quality achieved by aggregating the quality of individual component services. It is in this context that the quality of component services comes into play. As compatible services are selected and added into the mashup, their quality indicators are aggregated to estimate in which way they influence the quality of the final mashup.

In summary, mashability helps ranking the available components, based on their capability to be syntactically combined with previously selected components and to maximize the quality of the overall mashup. It is therefore a dynamic estimation of quality, which progressively takes into account the component services' attitude to be composed into quality mashups.

5 Role-Based Composition Quality

Mashup quality is not simply an aggregation of the quality of the individual components; therefore assessing and aggregating the quality of each component service is not enough. Mashup quality depends on the particular combination of components into a composite logic, layout and, hence, user experience. Component services, especially those provided with a user interface and therefore visible in the mashup, may play different roles that affect the perception of the quality of the final integration, and must therefore be carefully taken into account. By analyzing the most popular mashups published on programmableweb.com we have identified three typical roles [4]:

- *Master*: Even if a mashup integrates multiple components in one page, in most cases one component is more important than the others, being the service the user interacts with the most. It usually is the starting point of the user interaction that causes the other components to react and synchronize accordingly.
- *Slave*: The behavior of a slave component depends on another component; its state is mainly modified by events originating in another (master) component. Many mashups also allow the user to interact with slave components. However, the content items displayed by slave components are selected via the user's interaction with the master component, and by automatically propagating synchronization information from the master to the slaves.
- *Filter*: Filter components allow the users to specify conditions over the content shown by the other components. They provide (possibly hierarchical) access mechanisms that allow the users to incrementally select the contents they want to see. They also reduce the size of the data set shown by the



Fig. 2. Basic mashup development patterns [4]

other components, thus improving the mashup understandability and ease of use. In most cases, filter conditions are specified over the data set of master components, while slaves are synchronized and, hence, their content is automatically filtered by the integration logic.

Figure 2 highlights three basic patterns based on the previous roles of component services. The roles impact mashup quality. For example, in the slave-slave pattern represented in Figure 2(a), filters have minimal influence over the perceived quality of the components: being oftentimes developed by the mashup developer, they do not allow the specification of filter conditions that cannot be satisfied by the slave component. In other composition patterns, such as the master-slave and the master-master patterns depicted in Figure 2(b) and 2(c), the master is the major responsible of the final quality; it could even degrade the quality of the other components. The perceived quality of the final composition can be therefore achieved as an aggregate of the quality of individual component services, but weighted taking into account the services roles.

In order to describe the roles of component services, a mashup composition can be modelled as an oriented weighted graph $G = (V, E, W)$, where each vertex $v_i \in V$ represents a component service; each arc $e_{ij} \in E$ is associated with a binary value that represents that one or more bindings are defined between the two connected components to couple v_i output parameters with v_j input parameters; the arc weight, $w_{ij} \in W$, represents the actual number of mappings defined between the two connected components. Informally, the role, and thus the relevance, of a specific vertex v_i in a specific mashup can be estimated by considering all its direct and indirect bindings in all the paths that link v_i to a vertex v_k . In fact, we assume that the influence of one component on the total quality of the mashup depends on the number of its total bindings towards other components. More details about the identification of roles and the computation of quality weights can be found in [3].

6 The Model for Mashup Recommendation

Preliminaries. Based on the quality concepts introduced in the previous sections, in this section we propose a model for the production of recommendations able to support the mashup composer in the recognition of the most suitable components to use from a functional and non-functional point of view. Let us assume that the mashup composer could access a component registry C in which each component $c_i \in C$ is associated with a *component descriptor* and a *quality vector*. The component descriptor lists all the offered operations and related parameters and all the technical details needed to evaluate the quality dimensions described in Section 3. The quality vector $QD_i = [qd_{i1}, qd_{i2}, \dots, qd_{in}]$ contains the list of metrics associated with the quality dimensions. Starting from the quality vector, it is possible to define the value of the component quality cq_i as the aggregated quality index for the i -th component.

Quality assessment may play a fundamental role when selecting component services during the mashup development process. In fact, in case two components c_i and c_j are similar, that is functionally equivalent, quality can be a valuable discriminant factor. For this reason, it is worth to introduce the concept of *quality distance* between two mashup components as the absolute value of the difference between the quality of c_i and the quality of c_j :

$$Dist(c_i, c_j) = |cq_i - cq_j|$$

Besides the component registry, the mashup composer could also access a mashup registry M , in which previously composed “ready-to-use” mashups are stored. For each mashup $m_k \in M$, also assuming the availability of a composition descriptor, it is possible to access the list of the components $C_k \subset C$ used to create it. Additionally, given the description of the composition, it is possible to derive the composition graph and identify the pattern on which the composition is based, as described in Section 5. According to the approach presented in 8, each mashup could also be associated with an *Importance* index Imp_k that defines the mashup’s reputation. Importance is calculated taking into account the *mashup popularity* mp_k , that is the frequency of adoption by the community of users, and the *user rating* mr_k as proposed in 8. We refine the notion of Importance, also introducing the *mashup quality* mq_k ; therefore, given a mashup, its Importance is given by:

$$Imp_k = \alpha \cdot mq_k + \beta \cdot mp_k + \gamma \cdot mr_k$$

where α , β and γ are weights that express the relevance of each dimension, and $\alpha + \beta + \gamma = 1$.

The *mashup quality* is the characteristic feature of our approach. It is an aggregated quality measure that is calculated on the basis of the quality of the mashup components and of the patterns used to connect them 4.

The recommender algorithm. Considering the mashup development process described in Section 2, in this paper we aim at providing support to the mashup composer at the end of the component selection phase, or when a preliminary composition is available, i.e., when some components have been already included within the mashup. This allows us to identify the goal of the composition. In fact, our approach analyzes a mashup under construction m^* and provides recommendations about possible alternative compositions and/or extensions.

The main steps of the quality-driven recommender algorithm are four: (i) Discovery and evaluation of similar components; (ii) Discovery of similar mashups; (iii) Quality distance evaluation; (iv) Mashups ranking.

Discovery and evaluation of similar components. As soon as the mashup designer defines a composition m^* (final or intermediate), the first step of the quality-aware recommender algorithm has the goal to retrieve all the components that are similar to the components used in the mashups $C^* = [c_1^*, c_2^*, \dots, c_j^*]$. In detail, for each component c_j^* , the component registry C is analyzed in order to identify all the components that satisfy the mashability property, i.e. are syntactically and semantically compatible to c_j^* , and also improve the quality of the overall mashup. In this way, for each component c_j^* , an array of similar components $SC_j^* \subset C$ is created. Considering the set given by $c_j^* \cup SC_j^*$, it is possible to define the *ideal component* ic_j^* as the component that is associated with the maximum quality factor. The set of the ideal components IC_j^* then corresponds to the set of J components used in the *Ideal Mashup* im^* , that is the best mashup similar to the mashup m^* that the user can build according to his/her needs.

Discovery of similar mashups. The composition m^* is further analyzed for the retrieval of similar mashups. In details, the service registry M is analyzed in order to identify a set of mashups $SM^* \subset M$ in which each mashup $m_z \in SM^*$ contains at least J components equal or similar to the ones contained in m^* . A mashup $m_z \in SM^*$ could include also other components; it is however necessary that a similarity-based correspondence between the components $c_j^* \in m^*$ and the components $c_{zi} \in m_z$ is maintained.

Quality distance evaluation. As represented in Figure 3, for each mashup m_z in SM^* , the distance of a m_z from the ideal mashup is represented by the distance between a n -dimensional point p_{m_z} and the origin of axes, which is the ideal mashup. The point p_{m_z} describes the position of the particular mashup in the n -dimensional space and ranks the recommendations.

$$p_{m_z} = [1 - Imp(m_z), w_{z1} \cdot Dist(c_{z1}, ic_1^*), w_{z2} \cdot Dist(c_{z2}, ic_2^*), \dots, w_{zj} \cdot Dist(c_{zj}, ic_j^*)]$$

where $c_{zj} \in C$ are the components of m_z , ic_j^* are the ideal corresponding components, and w_{zj} are weights that express the following properties of components and composition patterns:

- the role of components in the mashup m_z as illustrated in Section 5;
- the similarity between c_{zj} , the m_z 's components, and c_j^* , the m^* components;

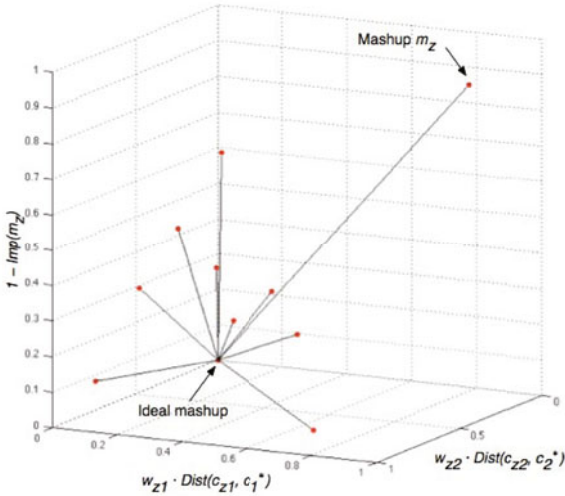


Fig. 3. An example of the vector space of similar mashups for a 2-component mashup. Segments represent the distances of the similar mashups from the ideal mashup.

- the position of each c_{zj} in the composition (e.g., let $c_1, c_2, c_3 \in C$, let \rightarrow the binding between two components, some possible compositions are $c_1 \rightarrow c_2 \rightarrow c_3, c_3 \rightarrow c_1 \rightarrow c_2, c_1 \rightarrow c_3 \rightarrow c_2$), so as to consider also different component bindings with respect to those defined in m^* , which however need to be downgraded through appropriate weights;
- the operations involved in the components binding in m^* with respect to the operation among components in m_z .

For the quality distance evaluation we need to consider: P_m the set of the p_{m_z} points, SC_j^* the set of the components similar to the m^* ones, and AC_z the set of the additional components which are the components present in m_z but not in m^* . We have introduced the two sets SC_z^* and AC_z to compute the weights w_{zj} , in order to give different importance to the components similar to the m^* ones, with respect to the others, because we want to penalize the components not in m^* , that do not match exactly the user’s needs. Differently from the approach presented in [8] we however think that super-compositions should be considered.

Mashups ranking Each m_z is associated with a point p_{m_z} in a multidimensional space where the origin of axes is the ideal mashup. The distances of each point from the origin can be calculated as the L2-norm $\|p_{m_z}\|_2$. The resulting ranking will provide information about the quality of all the mashups that the users could use to obtain the same functionality of the composition m^* .

7 Related Works

Several works have proposed quality models for traditional Web applications (see for example [14,13]). Few proposals also concentrate on modern Web 2.0 applications. For example, in [15], the authors extend the ISO 9126-1 standard, and discuss the internal quality, external quality, and quality in use of Web 2.0 applications. To our knowledge, there are no works explicitly addressing the quality of mashup component services and mashup compositions.

Our model for component quality is derived from the quality attributes defined by the ISO quality standard. We however add a specific perspective, which allows us to concentrate on the external quality of components, i.e., on the set of properties that affect the component's quality as perceived by the mashup composer (not necessarily the final mashup user). Other works focused on API quality in the more general SOA (Service-Oriented Architecture) domain, by specifically addressing the set of external factors that increase the ease of use of an API (the so-called *API usability*) [7,11]. Our approach capitalizes on these contributions but tries to go beyond, since it considers a broader set of external quality factors – not only usability –, all having impact on the success of mashup components.

Another novelty of our approach is that quality is the driving factor used for producing recommendations for mashup design. For service-based applications the literature already provides some approaches in which quality is the main driver for service selection and composition. For example, in [10] authors assess the overall quality of the final applications by aggregating the quality of the composing services. However, none of these approaches concentrates on the peculiarities of the mashup ecosystem.

Very few works have recently concentrated on *top-k* ranking of mashups (see for example [8,16]), to help designers in the choice of components and composition patterns. In particular, in [8] the authors propose a technique for ranking ready-to-use mashups, to support the development process through a novel auto-completion mechanism. The approach is based on the discovery of the common properties (i.e., components and compositions) that characterize the mashups developed by different users, and identifies classes of mashups that are similar to the one under development based on the “syntactic” inheritance of component services and composition patterns. Similarities is thus exploited to predict the most likely potential completions. Our approach is inspired by this work, but extends it in several directions. The main innovation is the promotion of the quality of components and compositions, which is adopted to identify the *ideal mashup*, and to compute the distance-based ranking of similar mashups. The concept of similarity is also extended by means of the semantic compatibility of components and composition - not only the syntactic compatibility.

8 Conclusion

In this paper we have discussed the need of addressing quality issues in the mashup development process. We have illustrated how a quality model for mashup

component services can drive the selection of quality components, and can also be used to assess the quality of the overall mashup by means of new concepts, such as *mashability* and *role-based composition quality*. We have then shown how the new concepts can be exploited to generate design recommendations.

The approach described in this paper is still at an early phase. In order to prove the feasibility of the most fundamental assumptions, we have partly implemented it as an extension of the MashArt platform [5]. We have in particular defined descriptions for mashup components that provide values for the quality metrics computation and, starting from them, we have implemented the method for ranking component services based on the mashability property¹ [3]. We have also implemented the graph-based algorithm to analyze the quality of the composition based on the component roles. Our current and future work is devoted to finalizing the implementation of the recommendation technique, conducting experiments (also involving real users, i.e., mashup designers) to assess the effectiveness and efficiency of the enriched development process, as well as measuring the performance of the proposed algorithms.

References

1. Calero, C., Ruiz, J., Piattini, M.: A Web Metrics Survey Using WQM. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 147–160. Springer, Heidelberg (2004)
2. Cappiello, C., Daniel, F., Matera, M.: A quality model for mashup components. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 236–250. Springer, Heidelberg (2009)
3. Cappiello, C., Daniel, F., Matera, M.: Assessing mashup quality by looking at composition models and patterns. Technical report, Politecnico di Milano (2010)
4. Cappiello, C., Daniel, F., Matera, M., Pautasso, C.: Information quality in mashups. *IEEE Internet Computing* (in print, 2010)
5. Daniel, F., Casati, F., Benatallah, B., Shan, M.-C.: Hosted universal composition: Models, languages and infrastructure in mashart. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009)
6. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding ui integration: A survey of problems, technologies, and opportunities. *IEEE Internet Computing* 11(3), 59–66 (2007)
7. Ellis, B., Stylos, J., Myers, B.A.: The Factory Pattern in API Design: A Usability Evaluation. In: ICSE, pp. 302–312. IEEE Computer Society, Los Alamitos (2007)
8. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for mashups. *PVLDB* 2(1), 538–549 (2009)
9. IBM. IBM Mashup center, www.ibm.com/software/info/mashup-center/.
10. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: Qos aggregation in web service compositions. In: *EEE*, pp. 181–185. IEEE Computer Society, Los Alamitos (2005)

¹ Semantic similarity assessment is based on the use of the Wordnet 3.0 dictionary and the Pellet 2.1.0 ontology reasoner.

11. Jeong, S.Y., Xie, Y., Beaton, J., Myers, B., Stylos, J., Ehret, R., Karstens, J., Efeoglu, A., Busse, D.K.: Improving Documentation for eSOA APIs through User Studies. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 86–105. Springer, Heidelberg (2009)
12. Jhingran, A.: Enterprise information mashups: Integrating information, simply. In: Dayal, U., Whang, K.-Y., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y.-K. (eds.) VLDB, pp. 3–4. ACM, New York (2006)
13. Matera, M., Rizzo, F., Carughi, G.T.: Web Usability: Principles and Evaluation Methods. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 109–142. Springer, Heidelberg (2005)
14. Olsina, L., Covella, G., Rossi, G.: Web Quality. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 109–142. Springer, Heidelberg (2005)
15. Olsina, L., Sassano, R., Mich, L.: Specifying Quality Requirements for the Web 2.0 Applications. In: Proc. of IWOST 2008, pp. 56–62 (2008)
16. Soliman, M.A., Saleeb, M., Ilyas, I.F.: Mashrank: Towards uncertainty-aware and rank-aware mashups. In: ICDE, pp. 1137–1140. IEEE, Los Alamitos (2010)
17. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) WWW, pp. 923–932. ACM, New York (2007)
18. Yu, S., Woodard, C.J.: Innovation in the programmable web: Characterizing the mashup ecosystem. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC Workshops. LNCS, vol. 5472, pp. 136–147. Springer, Heidelberg (2008)

Partial Information Extraction Approach to Lightweight Integration on the Web

Junxia Guo¹, Prach Chaisatien¹, Hao Han^{1,2},
Tomoya Noro¹, and Takehiro Tokuda¹

¹ Department of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1-W8-71, Meguro, Tokyo 152-8552, Japan
`{guo,prach,han,noro,tokuda}@tt.cs.titech.ac.jp`

² Digital Content and Media Sciences Research Division, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan
`han@nii.ac.jp`

Abstract. We present partial information extraction approach to lightweight integration on the Web. Our approach allows us to extract dynamic contents created by scripts as well as static HTML contents. Our approach has three application areas: automatic generation of Web services from Web applications, automatic integration of Web applications with Web services on desktop computers, and automatic integration of mobile phone applications with Web applications and Web services on mobile phones.

Keywords: partial information extraction, lightweight integration, desktop computers, mobile phones.

1 Introduction

The purpose of this paper is to present partial information extraction approach to lightweight integration on the Web and show that there are three important application areas of our approach in the lightweight integration.

The traditional way of lightweight integration on the Web is to integrate Web services by writing a script or a program invoking those Web services. This approach is natural but not directly applicable to the integration of the Web applications which do not have Web service APIs.

Some of the other existing approaches to lightweight integration on the Web are as follows. The first approach is to integrate the Web sources in predefined library using GUI interface as in Yahoo Pipes [24]. Users do not need to do programming, but they cannot add new Web sources into the library. The second approach is to use static Web contents extracted from Web applications as in Dapp Factory [11]. Users can use information extracted from Web applications and RSS to do lightweight integration, but they cannot extract dynamic Web contents such as the clock part of Localtimes.info [18] created by a script.

Our partial information extraction approach is a technique to extract partial contents from Web pages in one Web site according to GUI-based definition of

partial contents of a sample Web page in the Web site. Our technique is able to extract both static and dynamic contents created by scripts using the method called hide-and-display method.

Thanks to the hide-and-display method, our approach allows us to construct Web service functions from Web applications which do not have Web services. Our approach allows us to integrate both Web applications and Web services on desktop computers using descriptions. Our approach also allows us to integrate Web applications with Web services and mobile phone applications on mobile phones using descriptions.

The organization of the rest of this paper is as follows. In Section 2, we explain the detail of our partial information extraction approach. Section 3 explains the methods that can generate Web services from Web applications automatically. In Section 4, we explain our method to integrate Web applications with Web services on desktop computers. We present the method that integrate Web applications with Web services and mobile phone applications on mobile phones in Section 5. We evaluate our approach in Section 6. Finally, in Section 7, we give our conclusion.

2 Partial Information Extraction Approach

Classical partial information extraction methods allow us to extract parts of Web pages which are static data such as texts and images. We refer to these methods as first-generation methods. First-generation methods can extract titles and contents from news site articles, for example. More and more Web sites generate Web pages containing client-side scripts such as JavaScript and Flash instead of ordinary static HTML pages. Our partial information extraction method allows us to extract parts of Web pages dynamically created by client-side scripts.

We use the term static Web content to represent the content shown on the Web page directly in the HTML source file, for example texts and images. And we use the term dynamic Web content to represent the content created by client-side scripts dynamically, which is shown on the Web page but cannot be found directly in the HTML source file.

Our partial information extraction approach is as follows.

For static Web content, we supply two kinds of extracting methods. One method is to extract the target part in text format excluding the tags of the HTML document. For example, the extracted information of a picture is the value of attribute "src" of node , and the extracted information of a link is the value of attribute "href" of node <a>. The details of the extracting algorithm can be found in [3]. Another method is to use the same method as dynamic Web content, if the users want to keep the same layout of target parts with the original Web page. Users can choose the second method by setting the type of extracting Web content as "dynamic".

For the dynamic Web content, we use hide-and-display method instead of using the extracting method of static Web content to keep the functionality of dynamic Web content. This is because the scripts usually use DOM operation

to control the dynamic parts of Web pages and sometimes access to the elements outside of the target parts such as the hidden values. Therefore, if we just extract the target parts, the original executing environment of scripts may be broken. The hide-and-display method works as follows. First, we extract the whole HTML document of the target Web page. Then, we hide all HTML nodes in the extracted HTML document of the target Web page. Finally, we show the target Web contents and do the necessary settings. The details of the hide-and-display method can be found in [4].

In order to describe the necessary information for the extraction and emulation, we use XML-based notation called Web Application Contents Description Language (WACDL). The WACDL file includes the information about the target Web contents, such as the start page URL of the Web application, the path of target Web content, the type of the Web content and so on. The detail about the definition and format of the WACDL file can be found in [2].

By the Web contents extraction method and the hide-and-display method, we can get any parts from any Web applications, and maintain the functionalities of dynamic Web contents. Based on this description-based partial information extraction method, we can generate Web services from general Web applications automatically. And we can integrate both Web applications with Web services on desktop computers. Also we can integrate Web applications with Web services and mobile phone applications on mobile phones based on this method. We explain the three kinds of applications in the following sections.

3 Automatic Generation of Web Service Functions

Our partial information extraction approach allows us to generate Web service functions in mainly two ways: static construction and dynamic construction. In static construction, expected input and output information is extracted from Web pages of one Web site and construct a table. This table offers a Web service function returning output information for given input information. In dynamic construction, input information is sent to expected Web application and output information is extracted from the resulting Web page.

An ordinary Web service responds to the requests from users by returning the data from server-side information resources, usually a database. For our Web service construction, the information resources are the Web applications. Thus, our method needs to extract the information from the Web applications to generate the resulting tables. Based on our partial-information-extraction-method and resulting-table-query-method, users can generate Web services as described in the following steps: Firstly, users select the target parts from Web pages to generate the extraction patterns, which consist of the names and data types for the selected parts. Secondly, users run the constructed Web services at a proxy server, and configure Web service interfaces for them. Thirdly, the Web services use the extraction patterns to extract the partial information statically or dynamically according to the requests of users to create the resulting tables. Finally, the Web services search for the desired information from the resulting tables and respond to the users with the corresponding field values.

We refer users to run our constructed Web services at a proxy server. In the proxy server, the extraction patterns are used to extract the partial information and the resulting tables are generated. By using designed interface, the users send the requests to proxy server and get the responses from the resulting tables.

Usually, for the users of an ordinary Web application that is suitable for Web service construction, there are two basic types of methods to send their requests. The first type is to enter a keyword into an input field by keyboard and click the submit button by mouse to send the query. The second type is to click a link or an option in drop-down list in Web page by mouse to view a new Web page. For the first type, the URLs of target Web pages are changed by the requests of users. We call them Dynamic URLs. For the second type, the URLs are not changed by the requests of users. We call them Static URLs. In order to get the URL from all kinds of Web applications, we use HtmlUnit [15] to emulate the submitting operation.

In the Web applications, some information is static or changed periodically. We do not need to extract them dynamically, especially the static information with static URLs, after the users send the requests every time. We define three kinds of extraction frequencies for Web services: dynamic extraction, periodic extraction and static extraction.

1. Dynamic Extraction: The Web services extract the partial information dynamically after the users send the requests. It is suitable for the information extraction from real time system or the search result Web pages with dynamic URLs such as CNN news search result pages.
2. Periodic Extraction: The Web services update the resulting table by extracting the partial information at a regular interval such as one hour, one day or one week. For example, the weekly ranking of hot items is updated once a week at Rakuten [19].
3. Static Extraction: The extracted information is unchanged in a long period such as the basic information of a country/region. For example, the capital city and area information of most countries are unchanged and the population and life expectancy information remain unchangeable in a long period at BBC Country Profiles. Users can set the extraction frequencies for the constructed Web services according to the actual update frequencies of target Web applications.

Web services are self-contained and self-describing. The most-used style architectures of Web services are SOAP and REST. SOAP stands for Simple Object Access Protocol. Google implements their Web services to use SOAP, and we can find SOAP Web services in a number of enterprises' software. REST stands for Representational State Transfer. A number of new web services are implemented using a REST style architecture these days rather than a SOAP one, such as the Web services of Yahoo, Flickr [13] and del.icio.us [12]. Compared with SOAP, REST is lightweight and easy to build, and provides the readable results.

Figure 1 shows a simple example that uses the automatic generated Web service to get the top ten news' title, link URL and update time from CNN News [10].

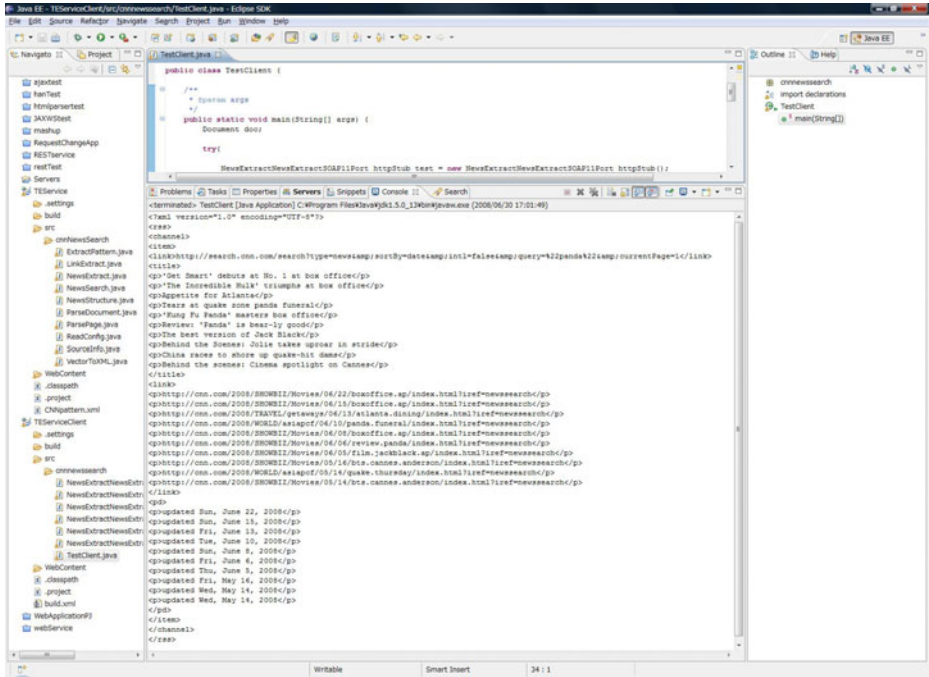


Fig. 1. Response of an Automatic Generated Web Service

4 Lightweight Integration on Desktop Computers

Our partial information extraction approach allows us to integrate Web applications with Web services. Web applications usually return resulting Web pages of much information. Our partial information extraction can select an expected part of the resulting Web pages.

To use parts of Web applications as mashup components, a normal method is that we use partial information extraction method to extract certain parts from Web applications and use them as mashup components. Most existing research work can only extract static Web content to generate mashup components. But based on our approach, we can extract not only static content but also dynamic content to generate mashup components. Furthermore, it allows us to deliver information between the generated mashup components.

Based on our approach, we build an example mashup application which integrates four components extracted from four Web applications: Localtimes.info [18], BBC Country Profiles [9], Weatherbonk.com [21] and ABC News [6]. After giving a country name, for example "Japan", and clicking the "Search" button, we can get a resulting Web page as shown in Figure 2, where the following components can be found.

Component 1. The local clock of the given country's capital city from the Localtimes.info Web site. The local clock is created by client-side script and

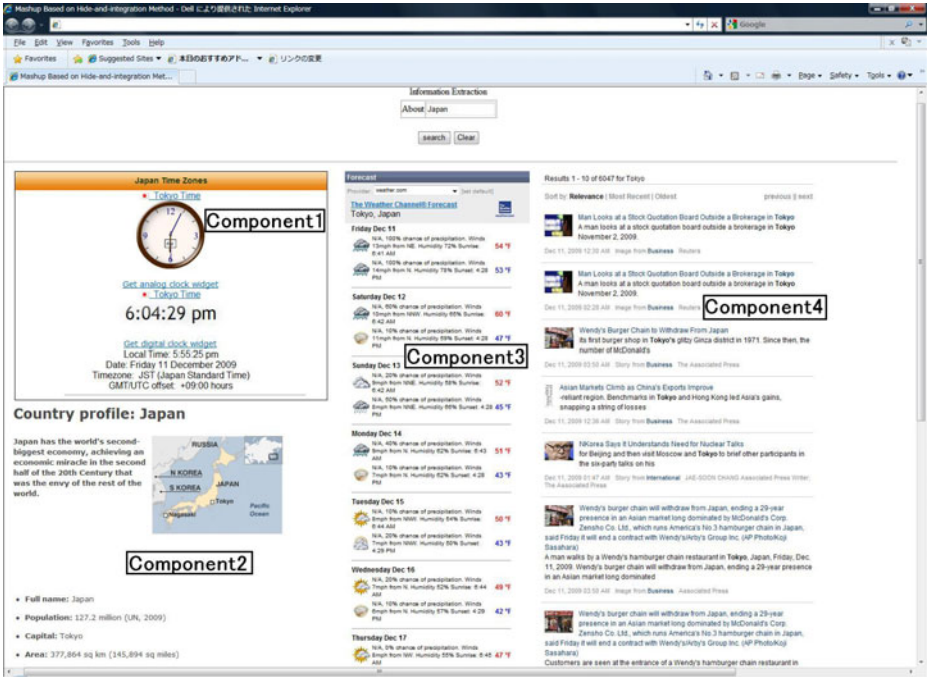


Fig. 2. Example of Mashup Web Application

the image changes every second. This component receives input parameter from the input-box of the user-interface.

Component 2. Introduction, location map and basic information of the given country name from BBC Country Profiles. The capital city's name extracted here will deliver to component3 and component4 as searching keyword. This component also receives input parameter from the input-box of the user-interface.

Component 3. The capital city's weather information from Weatherbonk.com. This component receives input parameter-capital city's name from "Component 2".

Component 4. The latest news articles from ABC News about the capital city. This component receives input parameter-capital city's name from "Component 2".

We create the mashup components one by one according to the recorded order in the WACDL file. As a result, the components can only use the information extracted from the component recorded before itself as the searching keyword. We support two kinds of basic information transfer styles with the above restriction. One style is that transfer information from one component to other all components. The other one style is that transfer information one by one sequentially. We also support the mix of these two kinds of information transfer styles in one

mashup application. We do not support the information transfer from multiple components to one component by now. We do not have technical problem on addressing this kind of usage with our approach, but it is different to create the rule of the combination which is extracted from multiple Web applications. We currently only support string type of information to be transferred between mashup components. Because there are many potential problems for other types of information. For example, the numerical value, although we can treat it as string when we extract or transfer the data. But we must make sure that the numerical value's format of the receiving component is same as the extracted data.

5 Lightweight Integration on Mobile Phones

Traditionally integration of Web applications with Web services on mobile phones uses construction of integrated pages rather than construction of integrated mobile phone applications. Our partial information extraction approach allows us to construct mobile phone applications by integrating Web applications with not only Web services but also mobile phone applications on mobile phones. Mobile phone applications deliver unique capabilities such as GPS location service, voice recognition and camera/image processing applications. Based on our partial information extraction approach, we present a description-based approach for the integration of mobile mashup applications combining partial extracted Web applications and mobile phone applications.

First, we present the image of a mobile mashup application example shown in Figure 3 to demonstrate how the description-based approach is used in composing a mashup application. The example consists of the integration of a mobile phone's user input interface with external Web resources. The mashup application aids users by shortening the interaction sequences. These sequences may conventionally require users to copy and paste one output from one section for using in another section. Furthermore, the partial extraction method reduces the amount of data and programming efforts using the native device programming code (such as network connection and HTML tags/text processing).

The objective is to integrate a mashup application which translates a product's bar-code into a product name, search for related information via Web applications based on our information extraction approach, and then display the output on the mobile phone. The following must be considered prior to the configuration process:

- Active components such as the Bar-code scanner component and the Web display component. These components require user interaction.
- Web application components which can use parameters generated from the Web extraction tool (i.e., Amazon.com [7] and Goodreads.com [14]).

First, users use the Web extraction tool shown in Figure 4 to generate parameters for their integrations, and then begin to configure each component in the

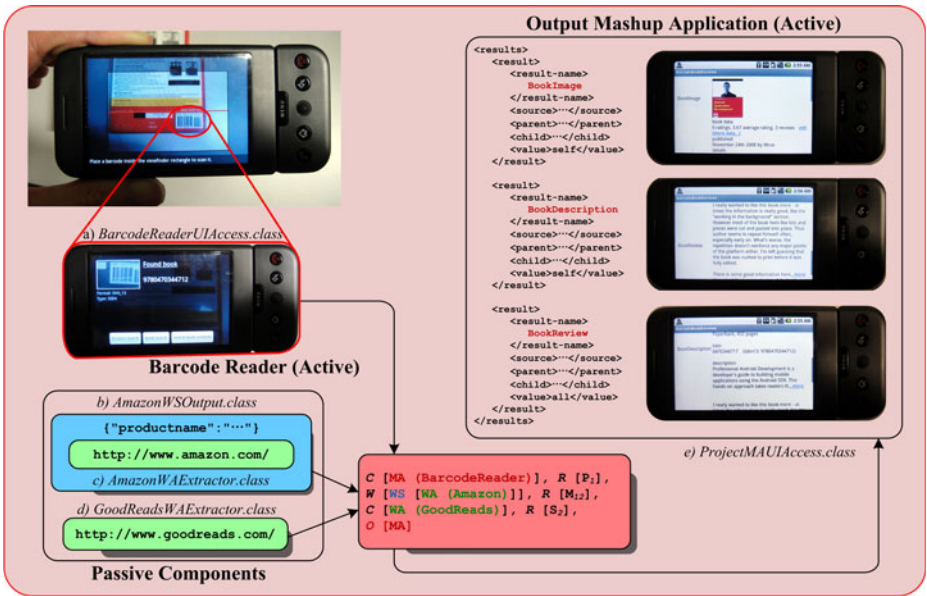


Fig. 3. The Image of a Book Reviews Bar-code Reader Mashup on Mobile Phone



Fig. 4. Sample use of the Web extraction tool

integration. Users are required to install the bar-code reader application manually and need to know the Android's Intent configurations of the bar-code reader application.

In the process for generating this sort of mashup application, the Android's Android Application Package (APK) file [8] contains two class files concerning the access to user interface while the i-jetty's Web Application Archives (WAR) file [20] contains three class files. Each generated class file performs a specific task as follows:

- a) BarcodeReaderUIAccess.class - accesses the bar-code reader interface via Intent.
- b) AmazonWAExtractor.class - extracts the information from Amazon.com [7] using parameters from a).
- c) AmazonWSOutput.class - generates JSON message from b).
- d) GoodReadsWAExtractor.class - extracts information from Goodreads.com [14] using parameters from c).
- e) ProjectMAUIAccess.class - accesses the user interface on the mobile phone to display output from d).

In intended use, after users manually install the APK file and the WAR to the mobile phone, user can access the bar-code scanner interface from the home screen.

6 Evaluation

This section evaluates our approach and application examples. Our work aims to support the flexible reuse of general Web applications in the lightweight integration on the Web. We build several mashup applications with various parts from different kinds of Web applications, and prove that our approach is applicable to general Web applications. We evaluate our approach from three aspects. First, we evaluate the extracting accuracy, which is the base of generating mashup components. Then we evaluate the efficiency of our approach for lightweight integration on desktop computers by comparing with other existing research works. Finally, we evaluate the efficiency of our approach for lightweight integration on mobile phones.

We choose several Web applications to generate components and build mashup applications. The experimental results about extracting accuracy are given in Table [1].

As the experimental results show, the average extracting accuracy is more than 90%. We checked the extracting results manually and counted the parts that had been extracted correctly. Our approach is based on the assumption that the searching result pages of the same request function are the same or similar within a Web application. But some of the searching result pages may be very different from other pages. As a result, among the all 201 countries and regions that we have checked in the BBC Country Profiles, we got correct results from 194 web pages. At present we support one path to describe the location

Table 1. Experimental Results of Extracting Accuracy

Web Application	Input Type for Search	Input Parameter	Num. of Checked Web Pages	Num. of Extracting Parts	Num. of Correct Pages	Percent of Correct Pages
abc News	Input Box	city name (Beijing, Tokyo ...)	194	1 (text, link)	194	100%
BBC Country Profiles	Option List	country name (Japan, Italy ...)	201	4 (text, image, bullet list)	194	96%
Localtimes	Link List	country name (Canada, Russia ...)	72	1 (dynamic content)	72	100%
Weatherbonk	Input Box	city name (Beijing, Tokyo ...)	194	1 (dynamic content)	189	97%
Yahoo Finance [22]	Input Box	stock name (AAPL, GOOG ...)	36	1 (dynamic content)	36	100%
Infoplease [17]	Link List	country name (Canada, Russia ...)	204	5 (image, text)	186	91%
Yahoo News [23]	Input Box	city name (Beijing, Tokyo ...)	189	1 (text, link)	189	100%
Total	—	—	1090	—	1060	97%

of "targetContent". We would like to support multiple paths to describe the location of "targetContent" and allow the users to decide how many paths they would like to use according to the accuracy they want.

We delivered the name of the capital city extracted from BBC Country Profiles to the ABC News and Weatherbonk. We got 194 (100%) correct results from the ABC News, and 189 (97%) correct results from the Weatherbonk. The cause of error is that some country/region names are the same as their capital city names. But in the Weatherbonk Web application, when we input the city's name, it is treated as a country name and we get a different page.

As we mentioned, there are mainly two kinds of existing approaches of lightweight integration on the Web. The first kind of approach is to integrate the Web sources in predefined library using GUI interface as Yahoo Pipes, iGoogle [16], Mixup [5] and so on. The users do not need programming, but they cannot add new Web sources into the library. The second kind of approach is to use static contents extracted from Web applications, as Dapp Factory [11], C3W [1] and so on. Users can use information extracted from Web applications and other Web sources to do lightweight integration, but they cannot extract dynamic contents. Based on our partial information extraction approach, we can extract not only static parts but also dynamic parts to do lightweight integration with other

Web sources. Furthermore, we can deliver information between the extracted parts.

Our partial information extraction method can be used with mobile application to do lightweight integration on mobile phones. In the configuration process of the mashup application presented in the example shown in Figure 3, users are required to use our Web extraction tool. We use the tool to specify and generate parameters to point to one result from a resulting list. To show more results, more parameters have to be specified which causes users to repeatedly use the Web extraction tool to generate more parameters. In general programming, programmers can write extraction loops; however, extraction loops is not presented in our method. In the case where many results are preferred, it is recommended that wrap the information extracted from the Web application using the Web Service Interface Wrapper so that users can apply the wrapper's outputs with typical programming.

To conclude, our method to perform Web information extraction on mobile phones using description-based configurations still requires manual works. The solution is to make the data flows diverge to parse the DOM tree remotely with other Web servers. In this case, we have to redesign the system architecture to be able to connect to external Web servers to do the configured tasks. Mobile phone hardware constraints and network latency also yields time lag when performing Web extraction. It is better to reduce the complexity of the DOM tree before the parsing process begins.

7 Conclusion

We have presented partial information extraction approach to lightweight integration on the Web. Our approach allows us to extract dynamic contents created by scripts as well as static HTML contents. We presented three application areas of our approach. Automatic generation of Web services from Web applications, automatic integration of Web applications with Web services on desktop computers, and automatic integration of Web applications with Web services and mobile phone applications on mobile phones.

References

1. Fujima, J., Lunzer, A., Hornbak, K., Tanaka, Y.: C3W: Clipping, Connecting and Cloning for the Web. In: Proceeding of the 13th International Conference on World Wide Web (2004)
2. Guo, J., Han, H., Tokuda, T.: A New Partial Information Extraction Method for Personal Mashup Construction. In: Proceeding of the 19th European-Japanese Conference on Information Modelling and Knowledge Bases (2009)
3. Han, H., Tokuda, T.: WIKE: A Web Information/Knowledge Extraction System for Web Service Generation. In: Proceeding of the 8th International Conference on Web Engineering (2008)

4. Han, H., Guo, J., Tokuda, T.: Towards Flexible Integration of Any Parts from Any Web Applications for Personal Use. In: First International Workshop on Lightweight Integration on the Web (Composable Web 2009) (2009)
5. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F.: A Framework for Rapid Integration of Presentation Components. In: Proceeding of the 16th International Conference on World Wide Web (2007)
6. ABC News, <http://abcnews.go.com/>
7. Amazon.com, <http://www.amazon.com/>
8. APK file, <http://developer.android.com/guide/appendix/glossary.html>
9. BBC Country Profiles, http://news.bbc.co.uk/2/hi/country_profiles/default.stm
10. CNN News, <http://www.cnn.com/>
11. Dapp Factory, <http://www.dapper.net/>
12. Del.icio.us, <http://delicious.com/>
13. Flickr, <http://www.flickr.com/>
14. Goodreads.com, <http://www.goodreads.com/>
15. HtmlUnit, <http://htmlunit.sourceforge.net>
16. iGoogle, <http://www.google.com/ig/>
17. Infoplease, <http://www.infoplease.com/countries.html>
18. Localtimes.info, <http://localtimes.info/>
19. Rakuten, <http://www.rakuten.co.jp/>
20. WAR file, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WCC3.html
21. Weatherbonk.com, <http://www.weatherbonk.com/>
22. Yahoo Finance, <http://finance.yahoo.com>
23. Yahoo News, <http://news.yahoo.com/>
24. Yahoo Pipes, <http://pipes.yahoo.com/pipes/>

Domain-Specific Mashups: From All to All You Need

Stefano Soi and Marcos Baez

Dipartimento di Ingegneria e Scienza dell'Informazione,
University of Trento,
Via Sommarive, 14 38123,
Trento, Italy
{soi,baez}@disi.unitn.it

Abstract. Last years, aside the proliferation of Web 2.0, we assisted to the drastic growth of the mashup market. An increasing number of different mashup solutions and platforms emerged, some focusing on data integration (a la Yahoo! Pipes), others on user interface (UI) integration and some trying to integrate both UI and data. Most of proposed solutions have a common characteristic: they aim at providing non-programmers with a flexible and intuitive general-purpose development environment. While these generic environments could be useful for web users to develop simple applications, they are often too generic to address domain-specific needs and to allow users to develop real-life complex applications. In particular, proposed mashup mechanisms do not reflect those specific concepts that are proper of a given domain, which domain-experts are familiar with and could autonomously manage. We argue the need for domain-specific mashup architectures, also going beyond today's enterprise platforms, in which standard mashup mechanisms and components are driven by an underlying domain-specific layer. This layer will provide a service and component ecosystem built upon a shared and uniform conceptual model specific for the given domain. This way, domain experts will be provided with mashup components and mechanisms, following those well-known concepts and rules proper of the domain they belong to, that they are able to understand, use and, finally, profitably compose. In this paper, we will show the necessity of such an architecture through a real-life use case in the context of scientific publications and journals.

Keywords: domain specific mashups, vertical mashups, end-user centric mashups.

1 Introduction

During last decade a vast amount of functionalities have been made available as online services, in form of Web Services, APIs, RSS/Atom feeds and so on. While these services can also be used independently from one other, putting them together to create a value-adding combination could lead to much more fruitful results, as described in [1]. This is exactly what mashup solutions try to achieve. In addition, most of available mashup platforms aim at giving the possibility to develop such composite applications to domain experts, i.e. users with very limited programming skills but

deep knowledge of the domain being the context of the problem to be solved. A number of studies (e.g., [2], [3]) discuss about benefits of moving the development of this kind of composite applications from IT-experts to non-programmers. This would be a radical paradigm shift bringing two main advantages, that is, first, avoiding requirement transfer from domain-experts to IT-experts and, moreover, allowing to face the development of situational applications¹, that is applications addressing transient or very specific needs for which the standard development lifecycle is not adequate since it would not be time- and cost-effective.

Current mashup building tools have the - non trivial - target of enabling domain-experts to develop such mashed up applications without - almost - any programming skill or any intervention of expert developers. This is often the main claim of available mashup platforms but, from our studies and experience in the mashup field, we found that this claim is only partially fulfilled. In particular, available mashup solutions provide easy mechanisms, suited for non-programmers, allowing them to produce very simple applications (often just "toy applications") or covering only some aspects of the integration needs of the users [4]. When users' needs go beyond this complexity level, available solutions show up their limitations.

In our opinion, main reasons underlying difficulties in overcoming these complexity limits are related to the fact that current mashup platforms (like [5],[6] and similar) have the ambitious goal of "integrating all the Web". In other words, they are not targeted at a specific domain but aim to give the possibility to make interacting user interfaces (UIs) and services coming from completely different domains and producers. For the time being, we think that "integrating all the Web" is a too ambitious goal. The lack of widely adopted - official or de facto - standards in this area make the integration of highly heterogeneous components a complex task that, at the end, is not sufficiently supported with mechanisms well-suited for non-programmers.

Starting from these considerations, we argue that there is the need for domain-specific mashup solutions. In particular, we propose to place the mashup system upon a layer defining concepts and policies of the given domain. We will see that this layer should include all the knowledge about the specific domain that could be then used to ease the mashup development, making actually possible to move the mashup application development from IT-experts to domain-experts.

Summarizing, the main contributions of this work are:

- stating the need for moving from horizontal general-purpose mashup solutions to vertical domain-specific ones, allowing domain-experts to autonomously compose their applications playing in their well-know playground
- proposing a modular architecture that makes a net separation between mashup layer and – pluggable – domain layer
- giving a first characterization of the “domain” concept, in terms of domain entities and rules, and first proposals on how the mashup platform should adapt to these.

To make our proposal clearer, we will explain the proposed solutions with the help of a use case, taken from the scientific publications context. In particular, we will make

¹ For details and references about the concept of situational application we refer to the Wikipedia page: http://en.wikipedia.org/wiki/Situational_application.

reference to a project our group is working on, called *LiquidPub* (<http://liquidpub.org/>), and we will show how the solution we propose could be profitably applied in that context. This will be the domain we will try to characterize and on which verticalize. We will base our example on a mashup tool coming from another project of our group, called *mashArt* (<http://mashart.org/>), which provides a complete mashup platform allowing for *Universal Integration* [7], that is seamless integration of data, services and user interfaces (UIs), targeted to non-programmer web users.

The rest of this paper is structured as follows. Section 2 will introduce and describe the motivating scenario, with particular reference to the LiquidPub project. In Section 3 we will see in more detail how generic mashup platforms work and what are their limitations. Section 4 will propose an architecture trying to overcome the issues presented in the previous section. In Section 6 will be presented the final conclusions and future work.

2 Motivating Scenario: Knowledge Dissemination

The Web has pushed forward technological and social changes in different areas and the scientific domain has not been the exception. It has opened a brand new world of possibilities for how the scientific knowledge can be consumed, produced, shared and disseminated. This has motivated an extensive research on how to exploit these opportunities, leading to novel *models*, new forms of *scientific contributions*, *metrics*, *services* and *sources of information*. Having a virtually infinite number of possibilities also implies that there could be different ways of consuming /disseminating /evaluating the scientific research work. The selection of the right configuration in terms of type of content (peer-reviewed papers, preprints, blogs, datasets...), the metrics (h-index, citation count, pagerank,...), sources (reputed publishers, open archives or the whole web) and the actual process will probably depend on the final usage scenario and the beliefs of the community. Implementing a particular dissemination model would normally require programming knowledge to produce the required code (e.g., in java, perl, ruby...), following a particular development process. Considering that everybody in the scientific domain has different thoughts on how to do this, it will be unnecessarily limiting to restrict this to programmers. It is clearly something end-users, or domain experts, should be able to do, and not only programmers.

Mashups provide the foundations for supporting such a scenario. However, current mashup platforms provide rather generic components, and so, at a level the scientist (non-programmer) cannot manage. For example, if a scientist strongly believes that i) blogs and open archives are valid dissemination venues, ii) sharing and consumption should be the main goal of a dissemination model, and therefore iii) sharing data should be used as base to evaluate researchers; presenting her a component that connects to a web service as primary tool does not help her in the composition of the dissemination model she believes in. Configuring and wiring components would become extremely complex (e.g., setting up a web service connection, or even selecting the right web service), and the user would be required to provide the mapping in terms of I/O among heterogeneous web services (when the whole concept of “mapping” is probably obscure to her), to reflect a flow and a process at such low level. Maintaining and reasoning over such a composition would also be a complex task, not to mention that there is no way to ensure that the final outcome is actually a dissemination model. Hence, in this context,

defining the desired dissemination model would not only be complex but it would require programming skills to specify the mashup, regardless how fancy the user interface might be.

Thus, domain concepts and processes (e.g., notions of publication, review, paper, venue,...) should be exploited in order to really assist her in the composition. Taking this as reference scenario, we discuss the limitations and required extensions to current mashup platforms, in the following sections.

3 Understanding Current Mashups and Their Limitations

As introduced in Section 1, the mashup approach should provide domain-experts with no programming skills with suitable mechanisms allowing them to develop autonomously their situational applications, leading to the above mentioned advantages in terms of responsiveness and effectiveness of solutions.

What makes possible moving application development from IT-experts to domain-experts, is probably the complete separation of roles, and thus of required skills, among component and composition developer, as depicted in Figure 1.

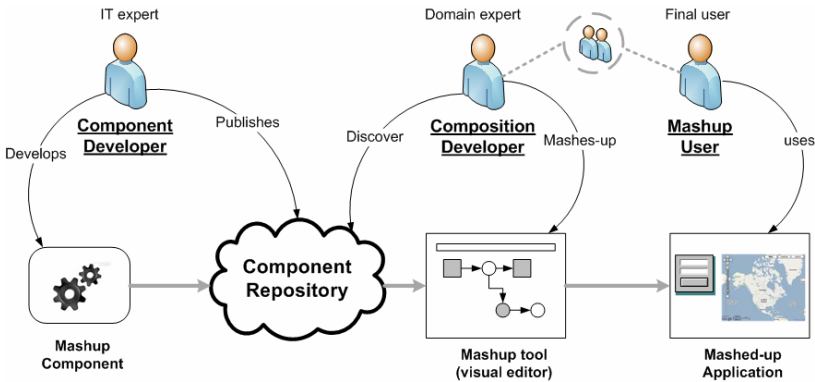


Fig. 1. Separation of roles among component and composition developer. Dotted representation among composition developer and mashup user, indicates that both roles can be covered by the same person, as typically happens in the context of situational applications. The figure is an adaptation of the one presented in [15].

The former is responsible to create and publish the building blocks that will be glued together to realize the final composite application. These components will implement or wrap some services or will represent a user interface. Complexity is primarily pushed into components, leaving compositions simpler and lightweight. It is clear that component development requires specific programming skills, in particular in the web-programming field, since mashup platforms are typically offered as web applications, following the Software as a Service (SaaS) approach [8]. To this end, a number of web tools have been proposed to help and simplify the creation of services and components (e.g., data extraction from web pages, web clipping) [9]. Some examples are OpenKapow and Dapper. Both this tools provide simple mechanisms to grab contents from web pages

and expose extracted data as web services or RSS feeds. However, this kind of tools still requires a not negligible knowledge of programming concepts to be effectively used.

Assuming that components have been developed and made available, composition developers, now, only need to define the business logic addressing their needs, connecting available components, usually through simple visual mechanism (e.g., drag and drop). This operation should not need any particular programming knowledge or complex operation and should be tackled by advanced web users that have a deep knowledge in their domain but no skills in programming. Typically, in the context of situational applications development, the domain-expert plays both the mashup developer and mashup consumer role (as indicated by dotted representation in Figure 1), since she develops compositions to automate processes covering her situational needs.

Providing autonomy in mashups composition to advanced web users is the big claim of most mashup platforms, but our experience and studies showed that it is not actually fulfilled in general. Proposed solutions allow domain-experts to compose their applications without the need to write programming code, but this does not mean that the composition process is easily and intuitively affordable by non-programmers [10]. In the example of Section 2, this would be the case for the scientist trying to select publication venues but that find herself with a web service connector. Analyzing available mashup tools, we concluded that they are often confusing for domain-experts, starting from the components selection that, given the vast amount of available possibilities, could be time-consuming and error prone. For example, if we analyze two popular visual environments for "Consumer mashup" composition, Yahoo! Pipes and Microsoft PopFly², we can see that they provide users with about 50 components for Yahoo! solution and more than 300 for Microsoft one. Moreover, when the needs require more complex mashup solutions many tools either are no more sufficient or start requiring to the composition developer (domain expert) a deeper and deeper understanding of programming concepts. In fact, a significant part of offered components provide functionalities that can be exploited only by those users that have good programming knowledge (e.g., regular expressions, loops). Another important lack of currently available solutions is that almost none of them provide *universal integration*, that is, as discussed in [7], the seamless integration of data, application, and user interface (UI) components, characteristic that we consider necessary to actually enable end-users to develop their situational applications. For instance, Yahoo! Pipes is mainly oriented toward data integration while Intel Mash-Maker mainly focuses on UI integration, but to build real-life complex applications both ingredients are needed. In the field of the "Enterprise mashup" tools many efforts is being done to address some business-critical issues, like security, privacy, reliability and accountability. From the point of view of domain-experts usability, enterprise solutions suffer of the same problems of consumer ones. In particular, although there exist powerful and complete mashup solutions, they are usually targeted at programming-skilled users. A noticeable example is the Tibco³ suite, providing users with a vast amount of different components and mechanisms, covering every possible need, but often strictly related to programming concept that domain-experts could completely ignore or, however, difficultly manage.

² Microsoft PopFly was discontinued on August 2009, but still remain an important mashup platform example that attracted thousands of developers.

³ <http://www.tibco.com>

All the generic components and mechanisms that available mashup solutions, both consumer and enterprise, provide and their programming-nature limit the possibilities of composition of domain-experts to "toy applications".

We argue that the main reasons for these limitations regarding most of the available tools need to be searched in their aim to be generic-application building tools. This general purpose attitude make it difficult for domain-experts to get familiar with components, functionalities and mechanisms representing concepts and entities they are not acquainted - and which they are not interested in. Moreover, such an approach aims at integrating components from different sources belonging to different domains, so, very often, making possible the communication among different components is a complex task still requiring specific programming efforts. Back to our example of Section 2, this would be the case for the scientist who wants to aggregate, according to her, valid venues of scientific resources (e.g., publishers, blogs, eprints) to incorporate them in her model. This would lead to complex mappings requiring, most probably, some programming skills.

Overcoming these issues requires a different mashup platform architecture allowing domain experts to work in their natural playground, where they are familiar with concepts and issues, so that they can tackle the development of their situational applications. To the best of our knowledge, there is no related work actually exploring the concept of domain-specific mashup. Next section will describe our proposed solution and architecture, aiming to enable real-life application development for domain-experts.

4 Domain-Specific Mashups

Domain-specific mashups is our proposal to exploit domain concepts at the mashup composition level in order to put domain-experts at the center by providing an environment that can really assist them in the composition of domain-specific mashups. So, we need specific solutions pushing domain concepts up to the composition editor level, so that users can play in their well-known conceptual environment. To achieve such a system, we propose a modular architecture including two main layers, as depicted in Figure 2.

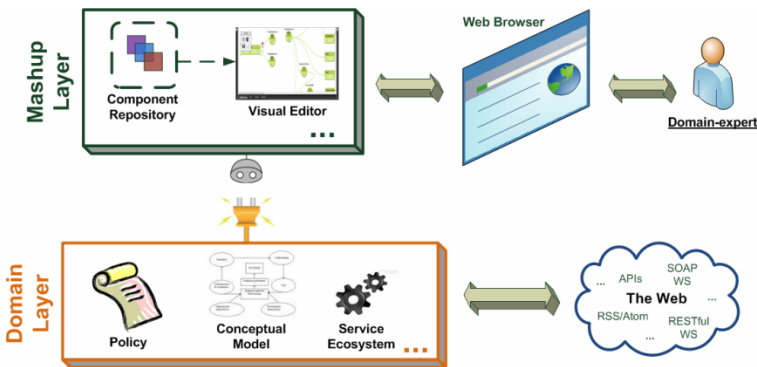


Fig. 2. High-level architecture of a domain-specific mashup platform

The upper layer is the actual *Mashup Layer*, that is, a mashup tool similar to some available today. In particular, this layer should - at least - include a composition visual editor, providing end-user friendly mechanisms for composition development through a common web browser, and a component repository, providing all the components that could be useful for building compositions in a given domain. In addition, other parts should be included at this level, like a runtime environment able to run the produced composition and other implementation-specific components, but those go beyond the scope of this work that is trying to focus on the composition-development phase seen from the domain-experts point of view. Our proposal to actually help and enable domain-experts to autonomously create their composite applications is to transform our generic mashup tool into a domain-specific one injecting domain related concepts into the development environment. Aiming at this, we create a *Domain Layer* that will be then plugged into the *Mashup Layer*. This lower layer is responsible for the domain characterization. In other words, it will define all the concepts and entities proper of the domain, their representation and general rules regulating the interactions among them. Furthermore, this layer will provide the domain related ecosystem of services, either implemented inside the layer or wrapping web-sourced services.

In this section we provide the two aspects covered by our proposal: modeling and characterizing the domain and its projection to the mashup platform.

4.1 Characterizing the Domain: Domain Layer

In order to leverage domain-experts knowledge in the composition, we need to understand the concepts, properties, rules and processes that make the domain. The definition of these elements is key to the selection of the right level of abstraction for users. To this end, we rely on the definition of the conceptual model, the business level operations and the domain rules.

Conceptual model. In the context of a particular domain, there are concepts and relations among these concepts that are known in the domain and familiar to domain experts. These concepts are commonly represented in a conceptual model. For instance, in Figure 3, we show a possible conceptual model for the example introduced in Section 2.

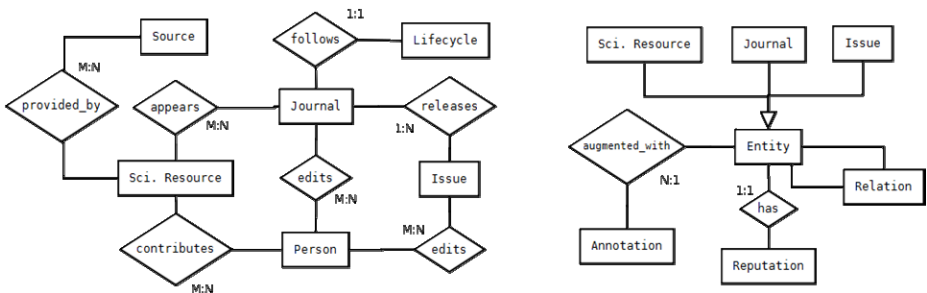


Fig. 3. Liquid journals conceptual model

The Figure above captures the concepts in the knowledge dissemination domain. It is based on the liquid journal model, which represents a family of models from the traditional ones to the ones more social and web-aware [11]. In this model we can see that *journal* is a first-class entity composed of *scientific resources* (papers, blogs, datasets, ...), organized in *journal issues*, driven by *editors* and that follows a certain lifecycle. We can also see that *scientific resources* belong to certain sources (venues).

Business-level operations. Operations and processes that affect the shared concepts are highly relevant. These relate to users' every-day life and as such provide the level at which the expert can better reason. Following our example, these are the operations which are meaningful in the domain such as *publish*, *evaluate*, *review*, *submit*, and other more social such as *share*, *annotate*, *search*, etc.

Business rules. Business rules are well known by domain experts. They are very important as they give shape to the business logic and processes. In our example, we could establish as a business rule that whatever publication model we follow, we need to first select/review a paper before publishing it in journal.

The information we have described above is present in the domain but not exploited in the mashup composition. Mashup composition environments strongly rely on the domain expert to build application from usually low-level components, and so they do little or nothing to assist users. To inject these elements into the composition environment we propose to build a *Domain Layer* as a way of hiding the unnecessary complexity which is currently exposed to users. Although we are convinced that the complexity could be pushed into the component design, having such a platform will make component development much easier and would ensure consistency and, finally, smooth composition. Another good reason for having such a layer is the increasing existence of domain specific ecosystems. Thus, in practical terms, concepts and operations are captured typically by a platform exposing an API. In our example, the *liquid journal* platform provides the services and key entities via RESTful services⁴. This platform builds on existing sources of information (e.g., Google Scholar, eprints, DBLP) and allows upper layer to access them using the common conceptual model in Figure 3. Solving the heterogeneity at this level has the advantage of not only providing homogeneous programmatic access, but also of avoiding taking complex mappings to the mashup environment by preparing the components according to the shared concepts. Of course, taking these and all the domain elements described here requires some work to make it happen. We discuss these and other related issues in the next subsection.

4.2 Taking the Domain into the Mashup Platform

Injecting the domain information provided by the ecosystem into the composition environment requires some work on both the component development and the mashup platform. More precisely, it requires (i) a proper design of the components in what regards the level of abstraction and composition, (ii) making the composition environment aware (to some extent) of the business rules by defining some composition rules, and thus taking to the mashup environment what is already on the backend, and (iii) making

⁴<http://docs.google.com/Doc?docid=0ARoLwpXLTjBGZGt6Mng0c18yZG00N3Y4Y24&hl=en>

the environment aware of the coupling among the components (and concepts managed by the components) in order to assist users in the component selection.

In what follows we describe our approach using mashArt as reference platform, albeit the ideas described here could be applied to other mashup platforms.

Building domain components. Good quality components are key to the success of any mashup platform and derived applications, and so is the case for domain-specific components. Thus, in addition to known practices for component development (e.g., [12]), building domain components puts some extra usability requirements. To reach users, we must select the right level of abstraction for the components and composition. It should be the one at which users find in the environment only concepts they know, expressed with the same terminology, i.e., components should be meaningful to the domain expert and be related to the business-level *concepts and operations*. In practice, we pass from components that represent just technology (e.g., a component connecting to a service) to components that have a precise semantic that is familiar to the domain expert (e.g., the component that publishes a paper). This is a conceptual shift.

Components should also be designed for smooth composition. Composing components should be straightforward to domain experts and complex mappings avoided to the possible extent. To this end, components events and operations I/O should be presented in terms of the domain concepts. In practical terms, this means that a domain-specific *namespace* should be made available to the component definition. Additionally, to ease and, at the same time, check the component development process, the platform could possibly provide a domain-specific component editor able to guide developers in the generation of new components (particularly for their descriptors), based on the knowledge coming from the *Domain Layer* (e.g., available entities and their representations).

In Listing 1, we illustrate the definition of a component designed taking into account a domain-level operation for providing familiar functionality, and domain-concepts in the I/O to ease the composition.

```
<?xml version="1.0" encoding="utf-8" ?>
<mdl version="0.1" xmlns:lj="http://liquidjournal.org/schema/liquidjournal.xsd">
<component name="Publish" binding="component/UI" stateful="yes"
url="http://mashart.org/registry/X/Publish/">
<event name="Paper published" ref="onPublish">
<output name="Published Entity" type="lj:entity"></output>
</event>
<operation name="Publish paper" ref="doPublish">
<input name="Entity" type="lj:entity"></input>
</operation>
</component>
</mdl>
```

Listing 1. MDL of the component Publish

As seen in the *Publish* component I/O refers to the type *entity*, an abstraction introduced in the conceptual model. A strong point of this approach is that it does not introduce any change into the MDL (mashArt Description Language, used to define each component of the mashArt platform) but it introduces higher level types based

on the conceptual model. In Listing 2 we show part of the definition of the XSD used to make the mashup platform aware of the conceptual model⁵.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema ... xmlns:tns="http://liquidjournal.org/schema/liquidjournal.xsd">
...
<xs:complexType name="entity">
  <xs:choice>
    <xs:element ref="tns:liquidjournal" />
    <xs:element ref="tns:issue" />
    <xs:element ref="tns:sciResource" />
  </xs:choice>
</xs:complexType>
...
</xs:schema>
```

Listing 2. XSD Definition of domain-level concepts

Business-level composition policies. As mentioned before, business rules are important and usually enforced at the backend. In order to ensure some patterns in the output and help users in the definition of consistent mashups we believe it is important to abstract these rules and take them to the level of composition. Of course, domain rules can be very complex to be completely pushed to the composition, so we target composition policies as a tool for "assistance" rather than "enforcement". There could be different ways of defining such policies, but in this paper we consider syntactic constraints based on category of components as simple starting strategy (that will be then extended in future). Components providing some common functionality could be categorized and category-level policies defined on them. Thus, policies allowing/denying cross-category couplings could be defined. Taking our scenario as an example, we could define categories such as *review processes*, *selection modes*, *publication modes*, *sources and venues*, *people*, *metrics* and *entities*, and on these define policies such as publication cannot be performed before the selection. Note that categorization is not mandatory and so "free" components not regulated by the policies are perfectly allowed. Implementation-wise, policies can be defined using XML and the mashup editor can check and guide the user during the composition based on the rules regulating that domain.

Mashup composition environment. The information about the domain components and policies should finally be reflected on the mashup composition environment. Domain components provide the opportunity to make meaningful suggestions based on components coupling (I/O matching) and so introduce a basic yet useful proactive behavior in the component selection. In its simplest, we could see composition as a domino where the domain-expert select components among the compatible ones. Domain policies provide even richer information. Policies will have higher priority over coupling based suggestions, filtering out and ranking eligible components. In addition to this, the mashup composition environment should provide intuitive UI

⁵ The complete XSD can be found here <https://dev.liquidpub.org/svn/liquidpub/prototype/ljdemo/server/resources/meta/liquidjournal.xsd>

representation for components and the connections (e.g., meaningful icons for components) to ease the selection.

Finally, having composition information at this level will enable further improvements in the composition environment. It would make it easier to extract usage information that could be used to improve the selection process (by reusing past experiences), since all components are defined at business level, making possible to extract the semantics of the compositions and usage.

5 Conclusion

In this paper we have introduced domain-specific mashups as a way to inject domain knowledge into the mashup composition, with the ultimate goal of providing domain-experts with the tools to compose mashup applications from familiar domain concepts. Our approach rather than proposing a technological change, proposes a paradigm shift in going from generic platforms with mainly low level (and technological-oriented) components to domain-specific vertical extensions. To this end, we have introduced a layered architecture in which we distinguish the mashup layer from a domain layer that can be plugged in. The definition of this domain layer is what allows us to describe the level of abstraction familiar to the user. In addition, the separation among the two layers allows the same mashup tool to be reused for different domain verticals, simply replacing the underlying domain layer.

As immediate future work we need to investigate more on how to define and model the domain characteristics (entities and rules) such that they are at the same time useful to help the composition phase but also not too rigid, guaranteeing the needed flexibility. Then, we plan to go from the conceptual modeling to the actual implementation of the extensions to the mashup environment. In particular, we plan to do that working on the mashArt platform and on the domain of scientific publishing, as introduced in this paper.

References

- [1] Jhingran, A.: Enterprise information mashups: integrating information, simply. In: VLDB 2006, pp. 3–4. VLDB Endowment (2006)
- [2] Floyd, I., Jones, M., Rath, D., Twidale, M.: Web Mash-ups and Patchwork Prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. In: Proc. of HICCS 2007 (2007)
- [3] Bitzer, S., Schumann, M.: Mashups: An Approach to Overcoming the Business/IT Gap in Service-Oriented Architectures. In: Proc. of AMCIS 2009 (2009)
- [4] Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: Proc. of the SIGCHI 2007, pp. 1435–1444 (2007)
- [5] Yahoo! Pipes project, <http://pipes.yahoo.com/>
- [6] Intel MashMaker project, <http://mashmaker.intel.com/>
- [7] Daniel, F., Casati, F., Benatallah, B., Shan, M.: Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In: Proc. of ER 2009, pp. 428–443 (2009)

- [8] Shan, M.: Software as a Service(SaaS) The challenges of application service hosting. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607. Springer, Heidelberg (2007)
- [9] Daniel, F., Soi, S., Casati, F.: Search Computing - Challenges and Directions. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 72–93. Springer, Heidelberg (2010)
- [10] Tuchinda, R., Szekely, P., Knoblock, C.A.: Building Mashups by example. In: Proc. of the 13th International Conference on Intelligent User Interfaces, IUI 2008, pp. 139–148 (2008)
- [11] Baez, M., Casati, F., Birukou, A., Marchese, M.: Liquid journals: Knowledge dissemination in the Web Era,
<http://eprints.biblio.unitn.it/archive/00001814/>
- [12] Daniel, F., Matera, M.: Turning Web applications into mashup components: issues, models and solutions. In: Gaedke, M., Grossniklaus, M., Diaz, O. (eds.) ICWE 2009. LNCS, vol. 5648. Springer, Heidelberg (2009)

Conceptual and Usability Issues in the Composable Web of Software Services

Abdallah Namoun¹, Tobias Nestler², and Antonella De Angeli¹

¹ Manchester Business School, Booth Street East, Manchester,
M13 9SS, United Kingdom

{abdallah.namoune, antonella.de-angeli}@mbs.ac.uk

² SAP Research Center Dresden,
Chemnitzer Str. 48, 01187 Dresden, Germany
tobias.nestler@sap.com

Abstract. Enabling the diffusion of lightweight service composition approaches among end users necessitates the appropriate understanding and establishment of the correct user requirements that lead to development of easy to use and effective software platforms. To this end, a user-centric study which includes 15 participants is carried out to unravel users' mental models about software services and service composition, their working practices, and identify users' expectations and problems of service composition. Several examples and prototypes are used to steer this elicitation study, among which is a simple composition tool designed to support non-programmers to create interactive service-based applications in a lightweight and visual manner. Although a high user acceptance emerged in regard to "developing service-based applications by end users", there is evidence of a conceptual issue concerning understanding the notion of service composition (i.e. end users do not think about nor do they understand connections between services). This paper discusses various conceptual and usability problems of service composition and proposes recommendations to resolve them.

Keywords: web services, light weight service composition, requirements, end user development, presentation layer, usability.

1 Introduction

Europe is continuously spending tens of millions of Euros to fund Service and Software Architectures research projects that aim at facilitating the development of interactive service-based systems through user-friendly software platforms. Sadly much of this research effort is dedicated to solving technical complexities and implementation problems rather than understanding what end users really need, how they think about service composition, and the natural working ways to support their activities and increase service creation and consumption. It is not uncommon that some of the funded projects decide on implementation strategies and software solutions for their target user group before even starting the project either because they want to pursue their own research interests or claim to understand the needs of their users without

carrying out user studies. This explains why, despite the rapid advancement in Service Oriented Architecture, user research in that area is still in its early infancy. User studies that focus on understanding user development abilities, needs, and mental models about service composition are evidently required to establish correct and well-informed requirements.

Web 2.0 enables Internet users to add content to the web, interact with and customize web pages, share information, and collaborate to accomplish tasks. In general, these activities do not necessitate acquisition of IT specialist knowledge since, for example, the act of modifying a wiki page is fairly simple for anyone who knows how to use a computer and browse the Internet. A promising approach to empower users beyond web content development is to use and reuse loosely-coupled software components such as web services [1] in order to create composite applications tailorable to their diverse needs. However, there is no substantial proof that naïve users (i.e. users with no Computer Science/IT background) can combine functionalities together to produce augmented software services. The question as to whether “users understand or think about uniting/connecting independent functionalities to form greater assemblies” is the key to the success of many service composition research projects.

Changing and customizing web content is different from composing software services since the latter is more complex and challenging. Creating computer programs usually requires strong modeling abilities and problem-solving skills which most ordinary web users do not acquire, a fact that has always intimidated end users. Transforming ordinary consumers of services into actual producers of services invites various unanswered research questions: what do users understand by web services? are they just black boxes, user interfaces they interact with, snippets of code, etc? are end users willing to uptake development activities at the cost of time and efforts? how can end users be helped to better understand service connections and encouraged to develop software applications? which representations and metaphors should be used to enable easy development of service-based applications? Such understanding is crucial to using the appropriate representation of services and service composition in authoring and modeling environments.

In this paper, we present a user study in which end users with no computing expertise viewed and commented on several examples of service composition and interacted with an early prototype of a visual composition tool. The current paper endeavors to:

1. Capture users’ true understanding of web services and composition of service-based applications. We refer to this as “mental models” which explain how end users visualize web services and the inner workings of service composition. This finding will impact the way software services should be represented to end users in development environments.
2. Identify conceptual and usability problems that relate to service composition, as probed by realistic examples and prototypes of a visual composition tool.
3. Improve the design of service development environments through a set of design recommendations; thus enhancing the diffusion of services among Internet user.

2 Existing Work on Data Mashups and Service Composition

At present the web offers users the capability to build personal pages through customizable web portals, such as iGoogle¹ and MyYahoo!², where they add web feeds and gadgets (i.e. programs that provide services) to their personalized pages. In principle, users browse a list of services, e.g. Weather, Google Map service, and add the desired ones to their pages. They can also edit these services and modify the look and feel of their pages by applying a desired theme or moving the gadgets within the page outline. Widget-based applications and customizable web portals are easy to use but do not support the creation of complex software applications because services can not be combined with each other, in other words users can not create or manage connections between web resources and services. It would be more useful if ordinary users are enabled to compose software components together in order to produce rich and complex service-based systems that fulfill their specific needs, but are also allowed to extend and customize these applications easily.

Although Web 2.0 facilitates the formation of web-based communities and consumption of services, such as: wikis, video-sharing, and social-networking sites, it does not allow the formation of powerful applications that consist of web services interacting together ([11], [14]). Service composition is well-understood and covered by existing approaches for technical developers using composition languages (e.g. BPML, BPEL4WS, WSCDL ... etc) [13], however tools and methodologies for enabling end-user service composition have been largely ignored ([6], [12]). The current technical aspects of service composition are not of much interest to ordinary users who want to capitalize on the benefits offered by Service-Oriented Architecture. What really matters to end users is how these technologies are presented to them and how they can use them to perform their desired tasks in an easy manner and without the need to learn programming languages.

Promising approaches for a lightweight user-driven application design are Mashup platforms (overview provided by [7]), which appeared with the growth of Web 2.0. The graphical composition style constitutes a first step towards user empowerment and increasingly shifts the creation of individual applications to domain experts. Mashup platforms like Yahoo! Pipes³ or Open Mashup⁴ enable the creation of more sophisticated service-based applications by aggregating web feeds, pages and services from different sources. Unlike customizable web portals, users can define relationships between modules by dragging and linking them together within a specialized visual editor. The output of one module can serve as the input of another. However, mashups mainly focus on data aggregation and still lack concepts to create composite service-based applications by end users ([5], [12]). Furthermore, they require modeling skills and good understanding of computing concepts such as message and data passing which most web users do not have [15].

Efforts to make programming accessible to a wider range of audience exist in the literature, such visual programming [2] which uses visual notations and languages to

¹ <http://www.igoogle.com>

² <http://www.my.yahoo.com>

³ <http://pipes.yahoo.com>

⁴ <http://www.open-mashups.org>

facilitate programming, and programming by demonstration [4] which enables users to demonstrate the desired program by examples in a step by step manner. As regards service composition, Namoune et al conducted focus groups to discuss the risks and benefits of end user composition of service-based applications. End users were mainly concerned about the privacy and security of their personal data, as well as the underlying technical complexity they might encounter when composing service-based applications [9].

Despite the continuous progress in service-oriented technologies, service composition by end users (non-programmers) is an area in its early stages. Therefore, identifying the needs and specific requirements of ordinary users is a crucial prerequisite to the design of “*easy to use*” and “*easy to understand*” service composition environments. The challenge to service and Human-Computer Interaction (HCI) research lays in finding new methods to open up service composition to a larger population supplying non-technical users with an intuitive development environment that hides the complexity of services and service composition. This goal although desirable, opens other interesting challenges to HCI.

3 Experimental Set Up

The study at hand was conducted in the form of a contextual interview/inquiry with the aim of exploring end users’ natural mental models and the potential problems of service composition. This knowledge will help us create and shape service development environments that facilitate end user composition. Beyer and Holtzblatt argue that contextual interviews are very useful for identifying user contextual needs and how specific actions are performed in detail [3]. Moreover, they enable researchers to understand users’ environments and their work conduct; thus portraying actual user behavior. 15 non-technical students from the Manchester Business School participated in the study. Each individual interview took approximately one hour to complete.

3.1 Procedure

During the interview we set up a focus in regard to the objectives of the ServFace project⁵ and how these objectives will be fulfilled. In this study, the focus is to enable ordinary users to build composite software applications that are tailored to their needs using a light-weight simple composition tool called *ServFace Builder*. To guide the interview several widespread examples (e.g. iGoogle, Google Map Search Service), low fidelity prototypes, and a high fidelity prototype of a future authoring tool were used. Each participant was instructed to perform the following tasks.

1. Define web services, widgets, and web applications, and explain how these software artifacts work from a user perspective. Following the participants’ answers, the interviewer provided the exact definition of each term with examples
2. View a mock-up of the composition tool and make initial comments and impressions, the interviewer asked elaborating questions such as: what do you see? What do you think it does? How do you think it works?

⁵ www.servface.eu

3. Walk through a simple service composition example “student course enrolment” in which the purpose and main features of the tool were explained by the interviewer (Figure 1). Through this selected example that suits participants’ environment and background we aimed to effectively communicate the idea of “service composition by end users”
4. Indicate their views regarding service composition and evaluate the mock-ups of the tool
5. Go through a service composition scenario and build a composite service using an early prototype of the tool (Figure 2). Concrete task description: “You are a team assistant for a team of 50 people. You can use MS Office well and like to play around with the tools you use and customize them to fit your needs. Since your colleagues often need to attend conferences it is your responsibility to organize their trips. In the past you spent a lot of time searching and booking suitable flights and hotels. In the future, you want to build an application that allows your colleagues to book their trips without spending a lot of time on it. *Thus, your main task is to build a software tool that facilitates travel booking using the ServFace Builder*”
6. Indicate their final views regarding the composition tool and the general composition approach

3.2 Materials

3.2.1 Motivating Examples

To simplify the definition of software services, widgets and web applications to our non-technical audience, we used examples that most people are familiar with, in particular: Google Map Search Service, Date and Time Gadget by Google, and Google Suit (email, Calendar, Documents, Web, Reader, etc). It is worth noting that these examples were only shown to participant after they had provided their definition and examples.

3.2.2 Low-Fidelity Prototypes and Student Scenario

Participants were shown a set of mock-ups of our service composition tool using Microsoft PowerPoint (Figure 1). The mock-ups demonstrated how a student can create a composite application that allows her to register for a particular course of study. The interviewer went through the process of visual service composition and explained the necessary steps. Subsequently, participants were invited to make comments or ask questions.

3.2.3 High-Fidelity Prototype

The evaluated ServFace Builder was developed in the frame of the EU-funded research project ServFace. The tool utilizes the advantages of web service annotations [8] enabling a rapid development of simple service-based interactive applications in a graphical manner. The tool applies the approach of service composition at the presentation layer, in which applications are built by composing web services based on their frontends, rather than application logic or data [10]. During the design process, each web service operation is visualized by a generated UI (called service frontend), and can be composed with other web service operations in a graphical manner. Thus, the

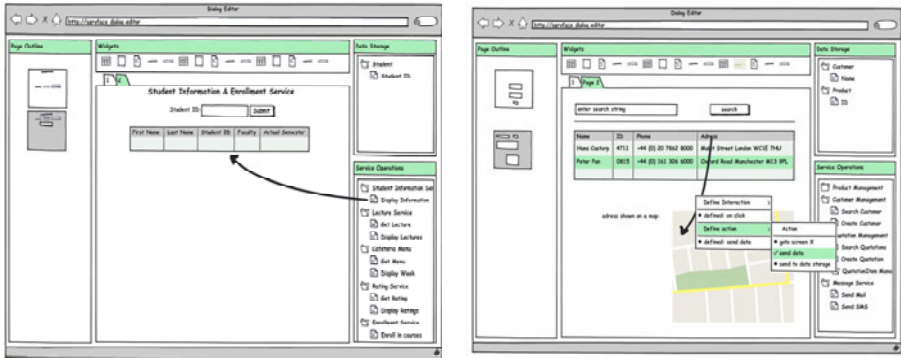


Fig. 1. Mockups of the Potential Simple Composition Tool –ServFace Builder

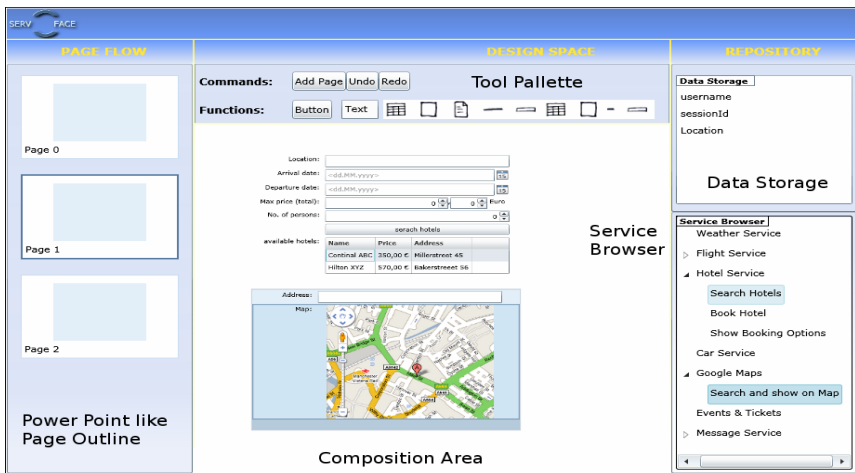


Fig. 2. Early Prototype of the Simple Composition Tool –ServFace Builder

user, in his role as a service composer and application designer, creates an application in WYSIWYG (What you see is what you get) style without writing any code. It is worth noting that the tool depicted in figure 2 has been improved and enriched with many features now based on users’ feedback and design recommendations gained through this study. However, for the interest of this paper, figure 2 shows the version of the tool we used to steer the contextual interviews.

4 Results

4.1 Users’ Mental Models of Software Services

Users provided diverse definitions for web services. 3 users defined services as services which are available on the web, while another 3 users defined services as the provision of information or knowledge. The remaining users referred to web services

as: a tool to build the web and applications, online communities, search engines, and interactive elements. 6 users argued that unlike traditional web pages which are static, services are interactive elements which enable them to perform tasks. When prompted to report examples of services 7 users mentioned search engines (Google, Yahoo) and E-commerce sites (Amazon). Others mentioned social-networking systems (such as: Facebook), website tools, and learning environments (the University portal).

Once the interviewer provided the exact definition of services to the participants and presented the Google Map as an elaborating example, all users were able to explain how it can be used. They had a very clear idea of how to interact with it, data can be entered by typing “search query or term” in the text field and clicking the “Search Maps” button. The service then returns the results back to the users in the form of an image (i.e. a map). Some users (3 users) also indicated that they can interact with the service by editing its options (e.g. show satellite imagery and show traffic). All users referred to the information they supplied as “*input*” and to the results returned by the service as “*output*”, showing that this terminology is commonly shared amongst users with no technical background.

The concept of widgets appeared to be more difficult. Although some users acknowledged to have heard the term, no one was able to define or guess what a widget is. 5 users defined web applications as applications that run on the web, for example iGoogle, Google docs, and Hotmail. Once the “Google suite” was introduced to the participants, they all commented that it is useful to have one application with many services bundled together as this is more convenient, saves time, reduces workload, and prevents errors.

To sum up users were able to provide a very basic and general definition of services abstracted from technical details. However, they were able to describe how they would interact with web services and web applications. Users seem to perceive service-hosting sites as single services instead of a collection of web services as shown by the provided examples. Users considered the term “widgets” technical and were not able to define it. Users knew about Web 2.0 and had already used its technologies like blogs. Surprisingly many of them had already built their own websites which confirms that web users are becoming proactive about developing the web.

4.2 Users’ Perception of Service Composition

All users liked “to develop their own software applications that suit their needs and interests” with the help of authoring tools. They argued that this will allow them to perform complex tasks more easily and rapidly, both in their personal and professional life. They also pointed out that assembling many services, for instance booking a flight, finding a hotel, booking and insuring a car service, within a single application is a powerful feature missing in today’s applications which are usually designed for one purpose. In normal circumstances, users have to access different online resources and services to accomplish their goals, which is a cumbersome process. Users appreciated that the introduction of service composition will reduce the amount of work required to perform tasks (i.e. logging into one service instead of many services/ applications), reduce the chance of making mistakes, and it will be more convenient to group heterogeneous services together in one application. Furthermore, integrating services using their front-ends only without worrying about the integration of data and business logic reduces the complexity of application development.

4.3 End User Composition Problems

As previously indicated one of the main objectives of this evaluation is to identify conceptual problems that can be generalized to other mashup editors and service composition environments and propose measurements to resolve them. Motivated by the travel booking scenario, participants pinpointed several drawbacks with the light-weight composition. Table 1 lists and explains both conceptual and usability problems that our end users faced while composing services. Other usability problems that are very specific to the ServFace Builder only are not reported below. Each of the detected problems was rated on a three-point rating scale (low, moderate, high).

Table.1. Conceptual and Usability Problems of Service Composition by End Users

Conceptual Problem	Severity	Usability Problem	Severity
1-Awareness of service composition/connection: despite introducing the concept of “building applications by users” in the walkthrough examples, participants had problems understanding the purpose of the composition tool (i.e. that the tool is designed to develop applications). Instead, they thought single services are software systems which operate independently. All users failed to mention that web services can be connected together. After we informed them about the possibility of combining services together and asked whether they want the tool to perform it on their behalf, all users preferred to be involved in the process of combining services because they do not fully trust the system and feared it might cause problems.	High	1-Direct and visual manipulation of services: selecting and placing services into the main canvas was not intuitive to the participants and caused problems. Upon placing services into the design area, users had difficulty trying to move them around to create an organized visual layout. Moreover, users wanted to adjust the size of services layout but this was not supported by the tool at the time of the study.	Moderate
2-Definition of execution flow of services and logic of application: users were confused about specifying the execution order of the services they added to the design space. In other words, they had problems specifying which service the application should execute first, which one should come next, and so forth.	High	2-System support: users were not sure if they were doing the right actions and complained that the system did not inform them about the consequences of their activities, emphasizing that proactive help from the system is important.	High
3-Understanding of technical terms: users were intimidated by some technical jargon used in the tool and their meaning such as service operation and parameters, and they asked for explanation. Inability to understand elements and concepts used within a design tool may result in users quitting the tool.	High		

Table 1. (Continued)

<p>4-Security: there was a security concern from users in relation to using web services that require supplying sensitive information such as: bank details. Users were worried that services retrieved by the tool could disclose their personal information to third party service providers or could be compromised by malicious intruders and hackers.</p>	<p>High</p>	
<p>5-Distinction between design and runtime: users had difficulty understanding the difference between design time and run time. Some users started to input data into entry fields of the services during the development phase and expected the application to process results accordingly. Clearly they had misconceptions about the two phases.</p>	<p>Moderate</p>	

5 Discussion and Recommendations for End User Composition

Although users were some times confused about the purpose of the tool, they showed a high likeability towards “composing service-based applications that are tailorable to their needs”. This agrees with the current trends that end users are becoming proactive about developing the web [11]. Assembling various services within a single application was favored by the participants because it saves time and effort, is convenient, and offers multiple functionalities, agreeing with [9].

The most interesting result of this evaluation revealed that the tool was not self-reflective of its composition aspect as users did not attempt to create links between services in the task scenario. This may be attributed to the innovative idea of combining different software components together which end users are unfamiliar with. In contrast to customizable web portals (e.g. iGoogle) and social networking sites (e.g. Facebook), where users usually search for and add external services to their pages without having to define relationships between services, this design tool and the alike require users to wire atomic services together.

Another challenge exposed by this tool is how can end users with minimum service composition experience specify the steps they are required to undertake in order to build an application. In particular, it is not an obvious task for users to recognize the services that contribute towards the accomplishment of a particular user goal.

Surprisingly adopting common practices in future design tools does not necessarily improve user experience and their usability. For example, although some users were able to link the page flow section on the right hand side of our service composition tool to the Microsoft Power Point slides section, they demonstrated poor ability to use it appropriately. It is important for designers to come up with and compare several design alternatives before committing to a particular design solution.

Other aspects of the tool that created confusion were related to the service computing terms, such as: service operation and parameters. This can be attributed to users' unfamiliarity and poor knowledge of technical terms.

In light of user understanding of software services and service composition and according to the identified service composition problems the following tentative design recommendations are suggested:

Rec 1- Service understanding and representation in service composition environments: users showed a poor understanding of the technical details of web services; thus, we encourage service designers to represent services via user interfaces to naïve users since visual representations can communicate and express the purpose and details of these services more effectively. Abstract representations of services (i.e. black box representations) are difficult to interpret and understand by non-programmers, whereas snippets of code are designed for serious programmers.

Rec 2- Service composition strategy: there are two fundamental issues to service composition, (1) connecting services and (2) identifying the order by which these services should be executed. To resolve these issues we propose to use a semi-automatic approach (i.e. system-guided composition) that seamlessly creates links between services while giving users the power to modify those links as they see most appropriate. Whenever a new service is selected from a list of services and added to the design space (e.g. service 3, Figure 3), the system should check for service compatibility issues and create the desired connection (Serv 3 and Serv 19, Figure 3). The system also highlights the possible links between a new added service and existing services. For the second issue we recommend to use a task modeling view by which users can specify the goal of their application and the tasks they need in order to accomplish their goal. Once the task analysis tree has been created users can associate services to particular actions (e.g. book hotel). The overall aim is to specify the services that contribute towards the accomplishment of a user goal without worrying about service connections. Furthermore, it would be very useful if a library containing several task analysis templates of possible assemblies is made available to users who can then reuse and extend them according to their specific needs.

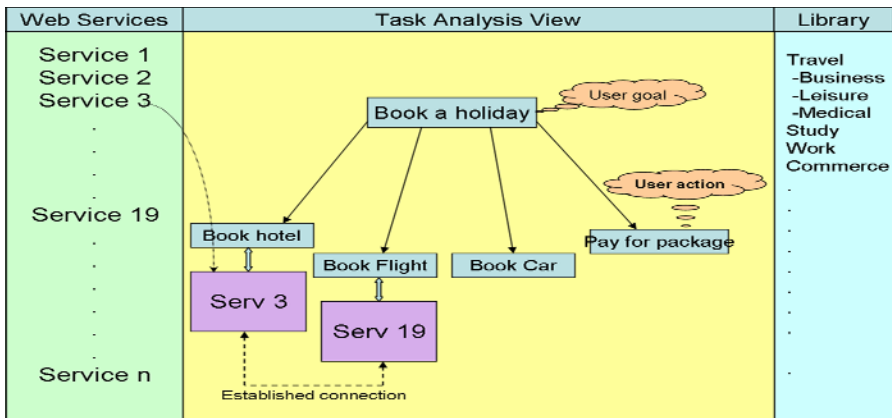


Fig. 3. A Potential Service Composition Design Solution

Rec 3- Service-related terms: technical jargon (i.e. service operations, parameter, Data types, widgets...) is not well understood by ordinary web users; therefore, we advise to use friendly and self-explanatory terminology to elevate technical complexity and enhance users' understanding of service composition aspects.

Rec 4- Visual Manipulation of services: service composition environments should provide a convenient and large design area. In addition, users should be enabled to easily interact with and visually manipulate services and their features (i.e. adding, moving and deleting services from the canvas, changing their dimensions, color...).

Rec 5- Secure services: the system has to deal with security and privacy aspects professionally. In that respect, the retrieved services must be trustworthy and reliable and this should be clearly communicated to the users through special symbols (e.g. a secure digital e-sign, verisign identity protection), as well as providing guarantees from service providers to compensate service consumers in case of frauds.

Rec 6- Continuous user feedback: users were sometimes inquisitive about the current state of their design; hence, we suggest adding proactive assistance (e.g. intelligent software agents) that continuously gives assurances and notifies users about the consequence of their actions, especially in critical situations.

Furthermore, we noticed that most of our users designed their application in one page instead of multiple pages. Therefore, we recommend enabling service composition within one design page as it is more convenient and less confusing. In regard to the design tool's name, users suggested using names that reflect their personality and give them a feeling of ownership and control over the tool such as: "myTool, myApplication, youDesign ... etc".

6 Conclusion and Future Work

This paper discusses users' mental models of services and service composition, and the main issues end users with no modeling skills or programming expertise may face during the composition of services using service development environments. Users favored the idea of being able to build their own applications but results showed they were not thinking about linking services together to form augmented assemblies and had difficulty distinguishing between design time and runtime concepts. We therefore propose to support service composition environments with intelligent mechanisms that automatically define connections between services (e.g. control and data flow) "system-driven composition" while allowing users to control and customize these connections. In future work, we plan to improve the current service composition tool by addressing some of the identified and inferred problems to simplify the process of building service-based applications. Subsequently, a user study will be conducted to evaluate the usability of the latest version of the ServFace Builder.

Acknowledgments. The work presented herein is supported by the EU-funded project ServFace.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web Services: Concepts, Architectures, and Applications*. Springer, Heidelberg (2004)
2. Burnett, M.: *Visual Programming*. In: Webster, J.G. (ed.) *Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons Inc., Chichester (1999)
3. Beyer, H., Holtzblatt, K.: *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers, San Francisco (1998)
4. Cypher, A.: *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge (1993)
5. Daniel, F., Casati, F., Benatallah, B., Shan, M.C.: *Hosted Universal Composition: Models, Languages and Infrastructure in mashArt*. In: *Proceedings of ER 2009* (2009)
6. Dustdar, S., Schreiner, W.: *A Survey on Web Service Composition*. *International Journal of Web and Grid Services* 1(1) (2005)
7. Hoyer, V., Fischer, M.: *Market Overview of Enterprise Mashup Tools*. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) *ICSOC 2008*. LNCS, vol. 5364, pp. 708–721. Springer, Heidelberg (2008)
8. Janeiro, J., Preussner, A., Springer, T., Schill, A., Wauer, M.: *Improving the Development of Service Based Applications Through Service Annotations*. In: *Proceedings of WWW/Internet* (2009)
9. Namoune, A., Wajid, U., Mehandjiev, N.: *Composition of Interactive Service-based Applications by End Users*. In: *The Proceedings of UGS 2009 - 1st International Workshop on User-generated Services at ICSOC 2009*, Stockholm (2009)
10. Nestler, T., Dannecker, L., Pursche, A.: *User-centric composition of service frontends at the presentation layer*. In: *The Proceedings of UGS 2009 - 1st International Workshop on User-generated Services at ICSOC 2009*, Stockholm (2009)
11. O'Reilly, T.: *What Is Web 2.0?*, <http://oreilly.com/web2/archive/what-is-web-20.html> (Retrieved April 20, 2010)
12. Ro, A., Xia, L.S.Y., Paik, H.Y., Chon, C.H.: *Bill Organiser Portal: A Case Study on End-User Composition*. In: Hartmann, S., Zhou, X., Kirchberg, M. (eds.) *WISE 2008*. LNCS, vol. 5176, pp. 152–161. Springer, Heidelberg (2008)
13. Van der Aalst, W.M.P., Dumas, M., Ter Hofstede, A.H.M.: *Web Service Composition Languages: Old Wine in New Bottles?* In: *The Proceedings of the 29th Conference on EUROMICRO* (2003)
14. Wong, J., Hong, J.I.: *Making Mashups with Marmite: Towards End-User Programming for the Web*. In: *Proceedings of CHI* (2007)
15. Zang, N., Rosson, M.B., Nasser, V.: *Mashups: who? what? why?* In: *CHI 2008 extended abstracts on Human factors in computing systems*, Florence, Italy (2008)

Crowdsourcing in the Document Processing Practice (A Short Practitioner/Visionary Paper)

Ehud D. Karnin, Eugene Walach, and Tal Drory

IBM Research Haifa Lab, Haifa University Campus, Haifa 31905, Israel
{karnin,walach,tald}@il.ibm.com

Abstract. The processing of scanned documents calls for automatic recognition of the text by OCR (Optical Character Recognition) computer programs, followed by human validation and correction. Crowdsourcing of these essential manual tasks is a good option, provided one can take care of some key challenges, so that the quality level expected by the customer is met. We show how tools for efficient validation and correction are adapted and enhanced to address issues associated with crowdsourcing, such as data privacy, quality control, crowd monitoring, and job quality assurance. We started to implement these ideas and technologies in our COoperative eNginE for Correction of ExtRacted Text (CONCERT), which is used in book digitization projects.

Keywords: Enterprise Crowdsourcing, documents processing, quality control, productivity tools, quality assurance.

1 Introduction

The process of recognizing text in images is a huge business, which is prevalent in virtually every industry. Forms (e.g., medical and insurance claims) and checks processing, license plate recognition, and book digitization are just few examples. Despite advances in automatic recognition by OCR programs, there are always characters which cannot be identified, or recognized with a low confidence level. The risk of character substitution, which may be painful in many cases, drives the thresholds up. In other words, only high confidence characters are accepted, and all remaining characters need to be verified or corrected by a human operator.

The naïve way for an operator to work is to get the form with the unrecognized or unsure characters highlighted in some way within the image of the document. Operators move over these areas on the form and correct where necessary. This is very inefficient; therefore other methodologies and tools were devised in order to enhance the productivity.

Years ago we developed an approach that we have called SmartKey [1] to boost the operator's productivity. In the following sections we shall explain how key challenges in crowdsourcing are addressed thanks to this approach and its extensions, therefore we precede by a short description of the method.

In SmartKey we collect all character-images that the OCR engine has identified as a certain character with a confidence level that is below a certain threshold, and present them in Character Sessions, or carpets. For example, Fig 1 shows a carpet for the digit 3.



Fig. 1. Character Session for the digit “3”

The operator would reject the two S (we highlighted them in Fig. 1) that were misidentified as 3. Doing so all other characters are thus validated. For the sake of argument, suppose that the 32 characters in this session were collected from 32 different forms and in each form only this character was recognized with a low confidence. 30 Forms are thus validated with the handling of a single screen, versus the need to go over 30 screens in the naïve approach. This explains the huge productivity boost. As for the 2 rejected characters, they will move to a Word Session, where operators will see them in the context of the word to which they belong. (There is more in Smart-Key, but this suffices for the follow-on discussion on crowdsourcing).

2 Virtual Service Delivery Centers

For many enterprises the document processing task is not a core business, so they may opt for outsourcing it to services delivery centers, on-shore or off-shore, provided the later can do the job with the required quality level. We have advocated [2] that the role of business services delivery centers will change, and they will basically turn into trusted Service Exchanges between enterprise-consumers and providers. The emerging Virtual Service Delivery Center, or VSDC for short, will be the broker that ties consumers to a providers pool, namely the crowd.

The consumers submit jobs that require both automatic and manual service. VSDC processes the job, performs as much as possible automatic processing, then segments the workload into subtasks suitable for the individual providers, matching tasks to skills. When the work is done, VSDC assembles and qualifies the completed subtasks and returns to requester, as well as handling the accounting.

We emphasize that VSDCs must establish “trust” in order to appeal to both consumers and providers. Consumers will look for guaranteed availability, quality, anonymity, privacy, and security. The small and medium businesses will also look for low entry barrier, and obviously every consumer is interested in low cost and fast setup. The providers would go for a trusted brokerage if they are guaranteed to be paid for their services, and would like the convenience of working over the internet (no commuting, flexible working time).

An efficient operation of VSDC relies on sophisticated technologies that introduce as much automation as possible into the process, and make a smart use of the manual work of the crowd. In the following sections we demonstrate how such technologies are applied to document processing, implementing the components of trust mentioned above.

3 Working with and Monitoring of the Crowd

We enlist key challenges in working with the crowd who provides the services, and show how they are addressed in our approach.

On-line quality control is maintained by introducing pseudo random errors in the Character Sessions (and other sessions) and measuring the percentage p_i of the missed characters by employee number i . This figure can be used for *fair rewarding* of the employees, with the compensation being a monotonically increasing function of $N_i \cdot (1 - p_i)$, where N_i is the amount of work done (e.g., number of screens validated by employee i). Further, *incentives plans* can be derived from this figure, and since it is being computed on-line it can also help urging an employee to take a break when the measured performance seems to drop due to fatigue.

A similar mechanism can be used for *training*, where screens that had already been processed are presented to new employees, along with instructions. While the employee is practicing, his or her performance is monitored, and feedback can be applied to maintain progress.

Privacy, or securing the *anonymity* of the individuals who filled the forms, is a key issue, which often limits the use of outsourcing, let alone crowdsourcing, when you do not know who will see the form. Clearly Character Sessions can be seen by anyone, since the operator has no idea if a digit comes from a social security number, date, \$ amount, or some other field in a form.

For the more advanced sessions, rules can be set concerning the *authorization* of employees to see various pieces of information, or perform operations on them. For example, words can be seen and corrected by people who have been passed some screening procedures; At the other extreme, resolution of addresses versus some ID numbers, which call for examining a large portion of the form, but would rarely be needed (as most of the forms are corrected at lower level stages) would be done in-house (i.e., not outsourced).

4 Satisfying the Enterprise's Quality Requirements

The service delivery center, which handles the document processing for its enterprise customers, usually signs a service level agreement (SLA) with its customer. The keep up with the SLA the delivery center should assure the job quality, and its timely availability.

Crowdsourcing poses a challenge for both requirements, as the delivery center is not at full control of the crowd. People would join crowdsourcing operations for the convenience of flexible work, so they do not commit their work time, and they are an anonymous group with initially unknown skill level and performance.

The way we decompose the jobs in the document processing practice, most of the workforce is occupied at the low level correction and validation stages. Hence people can be pulled into the working force in a relative short time, after fast training sessions. In such a way we can keep a large pool, mitigating workforce availability issues.

For *quality assurance* we take advantage of the on-line monitoring of the crowd. Suppose the customer demands a quality level $1 - p$, meaning that, at most, a small fraction p of characters remains erroneous. It might well be that for each i , $p_i > p$, so

no single employee can be assigned to this job. However, we can send the job to say employees number j and k , provided $p_j * p_k < p$. (We have made the plausible assumption that their errors are independent). Of course this technique can be generalized to the case where more than 2 providers are needed to get the expected error rate below the required threshold p . In other words, depending on the employee skill level, the same task may be performed either by one, two or three operators.

Another key advantage of our job decomposition methodology is that workloads are assigned to employees based on their skills. For example, in the processing of medical claims, the simple digit recognition will be done by basic level employees, while the validation of names of diseases will be done with people who have some medical background (either before their started to work, or via knowledge acquired by training).

5 Cooperative Correction System

COoperative eNginE for Correction of ExtRacted Text (CONCERT)¹ is a system that we develop for crowdsourcing workloads in book digitization, as part of the IMPACT project, one under the European 7th Program [3]. Some of the tools described above have been implemented, and the system is being used in a pilot. We shall describe the system in the Crowdsourcing Workshop.

References

1. US patent 5,455,875: System and method for correction of optical character recognition with display of image segments according to character data
2. Karnin, E., Walach, E.: Virtual Service Delivery Centers. Presented in Frontiers in Service 2007 conference (2007)
3. IMPACT Project, <http://www.impact-project.eu>

¹ The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2011 under grant agreement 215064.

Definition of a Crowdsourcing Innovation Service for the European SMEs

Fábio Oliveira, Isabel Ramos, and Leonel Santos

University of Minho, Department of Information Systems,
Campus de Azurém, 4800-057 Guimarães, Portugal
fjroliveira@gavea.dsi.uminho.pt, iramos@dsi.uminho.pt,
leonel@dsi.uminho.pt

Abstract. Based on literature review and on the study of the most known and referred Crowdsourcing brokers, there's a clear trend to implement this model by large companies and mainly within the North American context. Our research team is focused in bringing this approach closer to the European culture, more specifically the cultural factors underlying the dynamics and motivation of communities available to solve the innovation challenges of Small and Medium Enterprises (SMEs), that we call Crowdsourcing Innovation. We believe that, due to the common lack of resources for innovation in these companies, a service capable of involving them in large networks filled with useful and reachable knowledge, and capable of supporting these companies through all the innovation process, is crucial to the future competitiveness of the European SMEs. Although our team is focusing on several aspects related to Crowdsourcing, my main research focuses the information services and supporting applications to create a web platform adapted to the key economical, organizational, legal and cultural differences that make current Crowdsourcing Innovation businesses less popular among European SMEs than in North America.

Keywords: Crowdsourcing Innovation, European SMEs, Intermediaries.

1 Introduction

There has been a large interest in the Open Innovation phenomenon and in the Crowdsourcing as an Open Innovation model by academics and practitioners. However, these studies and discussions have focused large, North-American companies.

There are a number of important factors that lead us to believe that the exploration of the Crowdsourcing Innovation model in the European SMEs can be of decisive importance in the near future. First of all, it has been said that Europe has a handicap in competitiveness, greatly related with a weak investment in innovation[1]. If we take into account that SMEs have great impact in the European economy, and that due to the lack of structure and resources, these companies usually don't have the capacity to have formal R&D structures, then, it is easy to understand that these companies should be able to improve their innovation capabilities, and manage their innovation processes. They are forced to catch short windows of opportunities, often in collaboration with other SMEs or larger companies within their contacts network [2], [3] and [4].

Some authors consider that SMEs, being more flexible, have a competitive advantage against larger companies in radical innovations [5], and that this is the way to innovate in SMEs. But how many can generate radically new products? And for how long is such an advantage sustainable?

Based on these reflections, we believe that the use of the Crowdsourcing Innovation model by SMEs could be of great benefit. With the help of an adequate Crowdsourcing Innovation intermediary, these companies would be involved in a network able to generate useful and reachable ideas, knowledge and technology. By being involved in such a wide and rich network, more new ideas can be generated, more new ideas can be turned into new or better products and more of these can reach the market. The intermediaries in the market today, at least the larger and most well-known, fail to support these companies. There is also a lack of scientific literature about these services. So, my main research goal is to define an adequate architecture for a Crowdsourcing Innovation intermediary service focusing the unique characteristics of the European SMEs.

The objective of this paper is to present and discuss my motivations, findings and future work. In this article we will present our vision of Crowdsourcing Innovation, our prediction of what a Crowdsourcing Innovation intermediary for the European SMEs should be and our future work to achieve the proposed goals.

2 Crowdsourcing Innovation

Open Innovation and Crowdsourcing have become “buzz words” and their application and concepts have been broad, being sometimes applied, maybe wrongly, to almost any kind of collaboration over the Internet. The purpose of the term Crowdsourcing Innovation is to narrow the scope of the term Crowdsourcing and to better specify our goal.

Open Innovation has appeared as the new and promising way of achieve and manage innovation, by opening up the firms’ boundaries and letting ideas and technologies flow between the firm and its environment [6]. We understand Crowdsourcing as a way of outsourcing to the crowd tasks of intellectual assets creation [7], often collaboratively, with the aim of having easier access to a wide variety of skills and experience (as in Wikipedia). We focus on Crowdsourcing Innovation as a particular way to open up the innovation process, using large networks of individuals to access, capture and explore external knowledge, technologies and competencies, in other words to bring the “wisdom of crowds” into the company to help it innovate. This may lead us to question what really an Innovation is.

An Innovation is generally defined as a new way to do something. However, we like to take in consideration the ideas of Schumpeter, who defines inventions as new ways to do things, and innovations as inventions successfully applied in practice [8]. This means that an innovation should lead the company adopting it to a competitive advantage, either by cost or differentiation [9]. Sure that an invention is a step closer to an innovation, but it’s still far from being an innovation. We believe that Crowdsourcing it not only useful in creating inventions (the first stages of the innovation process) but is also useful in converting them in innovations (the latter stages of the innovation processes). This turns even more important in SMEs, which, as stated earlier, due to the smaller structure have, normally, serious difficulties to complete the innovation process.

3 Crowdsourcing Innovation Service for European SMEs: Challenges and Opportunities

The Open Innovation and Crowdsourcing Innovation models have been studied and applied mainly in large companies and the brokering services in market today reflect this trend. SMEs are typically the innovation and employment engines in society since they are usually more nimble and responsive to the business environment than the larger companies [10]. Crowdsourcing Innovation model can add to that the access to a wide collective intelligence available as a result of the governments' investments in education, science and technology. However, this can only be true with the help of a specific intermediary capable of answering these companies' needs and specificities, since they usually do not possess the resources to manage a network of creative people and to deal with the many complexities associated with knowledge and technology transfer [11].

These companies are likely to consist of talented people, entrepreneurs with particular skill sets. However, these companies may lack the full range of skills or capabilities to complete large assignments or fully commercialize innovative solutions. Consequently brokering services must provide support to SMEs' needs along all the innovation process' stages, and pre-invention and post-innovation support must be considered, unlike the brokers specialized in the innovation needs of large companies. Also, smaller companies may lack legal and financial resources to adequately protect their intellectual property, unlike larger ones. Therefore, the networking and sharing of their innovation problems can be seen as risky. The crowdsourcing innovation intermediary has to be particularly trustable and must help these companies to protect their intellectual property and sensitive information [12]. Also the business model behind the most well-known brokers is largely supported by consulting activities to prepare companies to open up the Innovation process. This seems inadequate for SMEs since it may increase the costs of using intermediary services. The business model must be re-thought to address this difficulty, both by ensuring a high number of associated SMEs and by using present innovation infra-structures already in place in the world, and in special in Europe, namely Living Labs, Knowledge centers.

The European context carries extra challenges. Firstly, the European companies have several cultural differences; for instance they are known for being more averse to taking risks. This makes that Open Innovation is being met with resistance [13]. Also, the European patenting system is under developed if compared to the North-American. The motivators for the crowd may also be different since job mobility is different in Europe and career development is highly valued. This may indicate that reputation development and community sense may be even more important than financial rewards. Another aspect is to develop formal ways of acknowledging the relevance of contributions of crowd members by their employers, present or prospect .

4 Research Plan

This research is in an initial stage and the several relevant issues are still being studied. The research question guiding the future work is: "What is the adequate architecture of

a web platform to support a Crowdsourcing Innovation intermediary service for the European SMEs?"

In order to answer this question the research will have three main conceptual focuses. First, it is necessary to deepen the understanding of the needs and specificities of the European SMEs' and the expectations they hold in embracing Crowdsourcing Innovation. Secondly, we want to understand the differences between the existing intermediaries based in the USA and those based in Europe, and to understand if these intermediaries act differently when dealing with SMEs and larger companies. To do so, we have already identified, based on [14], some intermediaries supporting Crowdsourcing Innovation in Europe that we will compare the most known intermediaries based in the North-America. Thirdly, we will study what motivates people to join the intermediary's network and how to best protect their intellectual property.

This project will be conducted as a Design Research project. As described by Vaishnavi and Kuechler [15], our work is currently in the end of the first phase, where this proposal is expected as an output. For the next phase, we expect to make several interviews to SMEs managers, and to representatives of existing Crowdsourcing Innovation intermediaries, to better understand their needs and expectations, and current platforms. Based on all previous work, a first version of a conceptual model for the web platform should be designed, and a prototype developed. This prototype will be deployed and promoted with the help of SMEs associations. The platform would be under evaluation, based on general utilization and the opinions of an expert panel, during a period of eighteen months. During this period the conceptual model of the web platform and the prototype will undergo all the perceived improvements, resulting in a validated conceptual model for a web platform supporting a Crowdsourcing Innovation Intermediary service for European SMEs.

The scientific contributes of this study will be the study and general conceptualization of the information services and technologies of a web platform of a Crowdsourcing Innovation intermediary service for SMEs. For practitioners, this work provides knowledge about the existing Crowdsourcing Innovation intermediaries and will empower them to improve their services. Hopefully, we expect the prototype developed in this project to evolve into a platform for an actual intermediary supporting innovation in SMEs being defined by several PhD projects: governance, risks, community, learning and memory, and knowledge repository.

References

1. Vigier, P.: Towards a Citizen-driven Innovation System in Europe: A governance approach for a European innovation agenda. *Innovation* 20(3), 191–202 (2007)
2. Edwards, T., Delbridge, R., Munday, M.: Understanding innovation in small and medium-sized enterprises: a process manifest. *Technovation* 25, 1119–1120 (2005)
3. Rothwell, R., Dodgson, M.: Innovation and size of firm. In: Dodgson, M. (ed.) *Handbook of Industrial Innovation*, pp. 310–324. Edward Elgar Publishing Limited, Aldershot (1994)
4. Rothwell, R.: External networking and innovation in small and medium-sized manufacturing firms in Europe. *Technovation* 11(2), 93–112 (1991)
5. Laursen, K., Salter, A.J.: Searching high and low: what type of firms use universities as a source of innovation? *Research Policy* 33(8), 1201–1215 (2004)

6. Chesbrough, H.W.: Open innovation: the new imperative for creating and profiting from technology. Harvard Business School Press, Boston (2003)
7. Howe, J.: The Rise of Crowdsourcing. *Wired Magazine*, 176–183 (2006)
8. Schumpeter, J.: The Theory of Economic Development. Harvard University Press, Boston (1934)
9. Porter, M.E.: Competitive Advantage. The Free Press, New York (1985)
10. Dicken, P.: Global shift: Mapping the changing contours of the world economy. Sage, London (2007)
11. Ramos, I., Cardoso, M., Carvalho, J.V., Graça, I.: An action research on open knowledge and technology transfer. In: Proceedings of the Conference The role of IS in leveraging the intelligence and creativity of SME's (CreativeSME), Guimarães, Portugal (2009)
12. Souza, L., Ramos, I., Esteves, J.: Crowdsourcing Innovation: A Risk Management Approach. In: The Proceedings of the 4th Mediterranean Conference on Information Systems, Athens, Greece (2009)
13. Koulopoulos, T.M.: The innovation zone: How great companies re-innovate for amazing success. Nicholas Brealey Publishing, Mountain View (2009)
14. Diener, K., Piller, F.: The Market for Open Innovation: Increasing the Efficiency and Effectiveness of the Innovation Process. RWTH Group, Aachen (2010)
15. Vaishnavi, V.K., Kuechler, W.: Design Science Research Methods and Patterns: Innovating Information and Communication Technology. Auerbach Publications, Boca Raton (2007)

Script Programmers as Value Co-creators

Cristóbal Arellano, Oscar Díaz, and Jon Iturrioz

ONEKIN Research Group, University of the Basque Country,
San Sebastián, Spain

{cristobal.arellano,oscar.diaz,jon.iturrioz}@ehu.es

<http://www.onekin.org/>

Abstract. Website owners are gradually realising the benefits of viewing customers as co-creators of value. Unfortunately, current development models offer little help in understanding and managing this new form of value co-creation. The Metropolis Model has recently identified three realms of roles for crowdsourcing: the kernel (providing the core functionality), the periphery (the partners) and the masses (the end users). Technically wise, the periphery requires mechanisms for the commons to suggest, develop and maintain additional services on top of the kernel. This work concretizes the Metropolis Model for crowdsourced website development based on user scripts. We outline some technical challenges to foster the relationship between end users (the masses), scripters (the periphery) and the web site (the kernel) on the way to promote script-based crowdsourcing.

Keywords: Greasemonkey, JavaScript, Crowdsourcing, Web2.0.

1 Introduction

Crowdsourcing entails a change in how organizations perceive their relationships with their customers—“the crowds”—and how they leverage them and their resources. Kazman and Chen introduce the Metropolis models where the collaborative development of a city is used as a metaphor of the new crowdsourcing paradigm [2]. They distinguish three realms of roles (and associated infrastructure) within a Metropolis Model: kernel, periphery and masses. Example roles for people involved in the kernel include architect, business owner, and policy maker; roles at the periphery include *developer* and *prosumer*; and roles for the masses include customer and end user. This work focuses on the periphery.

The periphery is populated by developers and prosumers. Unlike developers that do not need to be consumers of the final application, prosumer activities entail the creation of products and services by the same people who will ultimately use them, on the grounds that a large rate of interesting innovations come not from the suppliers but from the users. However, while mechanisms to sustain the *developer* role have been proposed based on the open-software movement, prosumers have been generally overlooked. An exception is Web2.0 applications where wikis and blogs move *prosumers* at the forefront as the value

creators. Web2.0 sites offer consumers the means to become also the purveyors of the website content. So far, this contribution is basically limited to basic interactions such as typing some text, providing tags, or clicking some buttons for ranking.

This work departs from this scenario, and focus on consumers that have some programming skills: **the scripters**. Scripters are first, consumers of websites, but they use their programming skills to improve not just the content but the functionality of the website. Script repositories such as *userscript.org* accounts for up to 36,057 registered users, while downloads are counted by millions. This large number of downloads evidences that there is a lot of value to end-users to repurpose the valuable functionality of websites for their own needs. But, what is the gain for webmasters?

Scripting can imply a threat to the monetization model of the website (e.g. banner removal), hence, it is currently perceived as a threat rather than an opportunity. As a result, scripting has to face no-collaborative, if not obfuscatory practices from the side of the website. We claim however that there is a lot to be gained by moving scripters from the masses to the periphery, i.e. as co-creators of value. Our contention is that the larger the number of scripts available, the more valuable and useful becomes the website. In an increasingly crowded panorama with distinct websites offering similar functionality, the website’s periphery could mean the difference between success and failure.

This document outlines some technical challenges in developing the periphery, namely: (1) insufficient decoupling between user scripts and the underlying website (i.e. the platform); (2) no means for website customers to know about user scripts; and (3), lack of mechanisms for websites to certify scripts.

2 Script Development

So far, user scripting is a “lonely practice”. No collaboration among scripters, end users and website owners. Specifically, all effort involved in developing those scripts can become useless when the underlying website is upgraded. Since scripts act directly upon *DOM* trees, minor changes in the *DOM* tree can make *XPath* functions retrieve the wrong node. This “sword of Damocles” certainly discourages scripters for writing elaborate scripts, and distances your website from becoming a platform.

Crowdsourcing implies encouraging contributor participation. In our setting, this implies sheltering scripts from upgrades in the underlying website. To this end, we propose the notion of *Modding Interface* [4]. So far, scripts rest on low-level User Interface (UI) events. Those events are dependent on the current HTML realization. If the page is upgraded, the scripts can stop working. The *Modding Interface* aims at sheltering the scripts from UI events by abstracting UI events into so-called *Conceptual Events*. A *Conceptual Event* describes the happening of an action (e.g. load, mouseOver, etc) on a *Concept*. A *Concept* stands for a *data compound whose rendering is liable to be modded as a unit*. In this way, websites are described in terms of the concepts their pages

convey, hiding the circumstantial representation of these concepts in HTML. For instance, “Paper” could be a concept for *icwe2009.webengineering.org*, “Bookmark” for *del.icio.us*, “Book” for *amazon.com*, etc. Now scripts subscribe to these events rather than UI events. Hence, upgrades on how concepts are supported do not impact the scripts. The *Modding Interface* for the ICWE’09 site can be obtained at *modding.icwe2009.webengineering.org/mi.xml*.

3 Script Advertising

So far, script finding is achieved through script repositories. *Userscripts.org* is a case in point. These repositories act as yellow pages which offer general information about scripts. In most cases, the user is forced to install the script to see what the script looks like. This difficulty in both finding and understanding the script are detrimental for the end user but also for both the script programmer and the website.

The website should take a more active role in publicizing community-provided scripts. So far, websites provide no indication about the scripts available to augment the base experience. However, it is for the benefit of the website to expose those augmented experiences to its customers.

The first step is to make the user aware that the website offers some augmented experiences (i.e. user scripts). This situation is similar to advertise the existence of *RSS* channels. Current browsers are able to detect a special “meta” in the HTML heading that causes the *RSS* icon to be displayed in the menu bar. However, this is not the case for “script channels”. Therefore, the website itself should provide some rendering means to make users aware of this channel. After all, *RSS* icons are still visible in most pages offering this service. Figure 1 shows the sample page but now a script icon is displayed on the right side. By clicking on this icon, a menu bar is worked out that lists the distinct user scripts available for this page. Besides the name, each user script includes a brief description, one

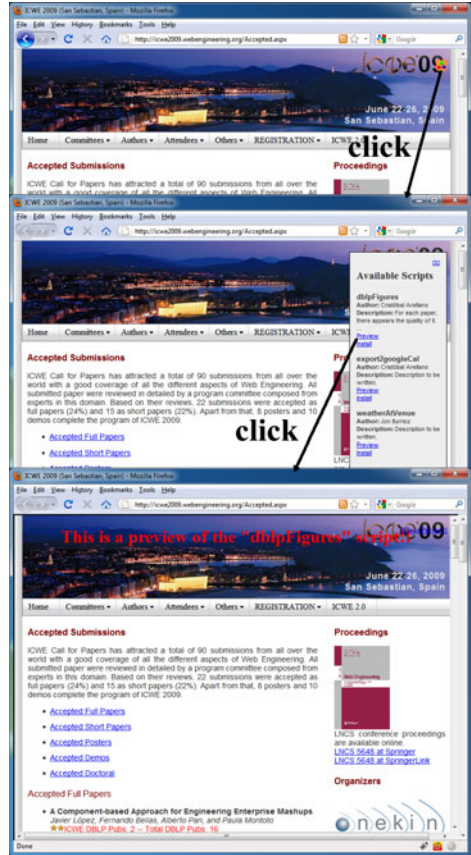


Fig. 1. Advertising user scripts through the web site

link to preview, and another link to install. Clicking on the preview causes the page to be refreshed. Now the base experience is augmented with the user script functionality.

4 Script Sandboxing

By publicizing community-provided scripts, the website’s reputation might be at risk. From a user perspective, the responsibility of user script malfunction tends to be handed over from the scripter to the website. Mechanisms are needed to ensure sound user scripts before being released to the general public.

Traditionally, the developer is responsible for the design, implementation and execution of the test cases. Now, the platform (i.e. the website) is at least as interested as the script developer to ensure the safety of the script. This first implies the existence of a clear interface between the platform and the user scripts. And second, an active involvement of the platform in defining the test cases to be passed for the script to be certified (i.e. for the script to be available through the website).

To this end, we introduce the *Modding Contract*. This contract states script functionality in terms of the causal relationship between event subscriptions and publications. These contracts are located at the script’s metadata block. For instance, the contract: *[loadPaper -> appendChildPaper]* states that each *loadPaper* event will cause an *appendChildPaper* to be signalled. It can be understood as follows: if the pre-condition “*loadPaper is risen*” is satisfied then, the postcondition “*the script will raise an appendChildPaper event*” will be ensured.

Unit testing can be used to check that a script meets its contract assuming its subcontractors meet theirs (i.e. the website generated appropriate *Publishing Events*). However, unit-testing design is time consuming and requires knowledge about how to obtain full coverage of test cases. This puts an extra burden on the script developer. On the other side, it is in the own interest of the website to thoroughly validate the user scripts. On these grounds, our approach leaves to the platform (i.e. the website) the duty of defining the test units. Scripts need first to be verified against these test units. Only if test passes, the user script is publicized, and liable to be installed.

References

1. Díaz, O., Arellano, C., Iturrioz, J.: Layman Tuning of Websites: Facing Change Resilience. In: The 17th International Conference on World Wide Web, WWW 2008 (2008)
2. Kazman, R., Chen, H.: The Metropolis Model: A New Logic for Development of Crowsourced Systems. *Communications of the ACM* 52, 76–84 (2009)

Quality Assurance for Human-Based Electronic Services: A Decision Matrix for Choosing the Right Approach

Robert Kern, Hans Thies, Cordula Bauer, and Gerhard Satzger

Karlsruhe Institute of Technology (KIT), Karlsruhe Service Research Institute,
Englerstraße 11, 76131 Karlsruhe, Germany
{robert.kern, gerhard.satzger}@kit.edu
hans.thies@gmx.de, cordula.bauer@gmx.net

Abstract. Crowdsourcing in the form of human-based electronic services provides a powerful way of outsourcing so called human intelligence tasks (HITs) to a large workforce of people over the Internet. Because of the limited control over that workforce, it is challenging to ensure the quality of the work results. Several approaches have been proposed that can be applied to specific types of HITs. However, it is difficult to identify a suitable quality management approach for any given type of HIT. This paper aims to provide a first sketch of a decision matrix.

1 Introduction

The idea of human-based electronic services is that they look like Web services but they are not performed by a computer, instead they use human workforce out of a crowd of Internet users. The success of Amazon's Mechanical Turk (www.mturk.com) platform and the growing number of companies that build their business model entirely on that platform demonstrate the potential of this approach. The MTurk platform acts as a broker between requesters who publish human intelligence tasks (HITs) and workers who perform those tasks in return for a small amount of money. Kern et al. proposed the term *people services* (pServices) for this type of human-based electronic services and define it as "*Web-based software services that deliver human intelligence, perception, or action to customers as massively scalable resources*" [3]. As there is limited control over the individual contributors, particular attention has to be paid to the quality of the work results. Several quality assurance (QA) strategies for pServices have been proposed [6, 5, 11, 2]. However, it is not obvious, which approach can be and should be applied to which type of pServices. The aim of this paper is to provide a sketch of a decision matrix as a basis for discussion.

2 Related Work

Sorokin and Forsyth distinguish between two strategies, that leverage the crowd for QA of pServices [6]: the collection of multiple results and the performing of a

separate review task (they call it grading task). In order to establish a common terminology we use the term **majority vote** for those approaches that *introduce redundancy by passing the same task to multiple workers and aggregating the results in order to compute the result with the highest probability for correctness* while we propose the term **review** for such approaches that *leverage individuals of the crowd for reviewing (e.g. validating) the results delivered by others* [2]. Majority vote mechanisms are already widely used in pServices scenarios today, e.g. on the MTurk platform. Initial research has shown that an amazing level of result quality can be achieved for basic tasks like natural language annotation, image labeling and data labeling: Sorokin and Forsyth identify objects on images by combining the drawings of silhouettes of distinct persons [6]. Snow et al. have ten distinct workers rate natural language expressions and compare their aggregated results to gold-standard labels given by experts [5]. Barr et al. mention the application of review for quality control on MTurk [1].

3 Decision Criteria

This chapter motivates and proposes a set of decision criteria for the selection of an adequate QA mechanism for a given HIT type. Four criteria are being proposed, two of them referring to characteristics of the task result, one to the execution effort and one to the level of required result quality. Each of them allows for a binary decision between two characteristics (Figure 1). In order to motivate the criteria we underline their relevance based on a series of examples:

Determinacy of the task results: For a *deterministic task* there is a well defined optimal result i.e. two workers who perfectly meet the task objectives will pass exactly the same result [2]. An example is the transcription of a speech recording if spelling and punctuation does not matter. In general, the majority vote (MV) approach can only be used for deterministic tasks. For non-deterministic tasks (e.g. creation of creative designs, text authoring and language translation), it won't work, because multiple results provided by different workers cannot be compared or aggregated in order to derive a single correct result. However, the *review* approach can obviously even be applied to non-deterministic results because a human reviewer is capable of dealing with variety.

Execution versus validation effort: Depending on the type of task, the effort for executing the task can be much *higher* than the effort for validating the result of the task. One example is text authoring. It is usually much simpler to decide, whether a given text meets certain quality criteria than to write the text. Another example is an image research tasks: it is much harder to find a picture which shows five bananas than to validate that there are indeed five bananas shown on the picture. For other tasks, the execution effort is *similar* to the validation effort. Examples are basic classification tasks and such research tasks for which a result validation is only possible by performing the research again.

Required level of quality: Depending on the required level of quality, the *wisdom of the crowd* [7] might be required in order to produce an acceptable result. Consider for example a one page language translation: If the quality needs

are *low*, a single qualified translator might be able to deliver an acceptable result. However, if the requestor is intolerant to mistakes (*high* quality needs) it is very hard for a single worker to meet the quality needs.

Number of equivalent quality relevant entities: If a result consists of many equivalent quality relevant entities, the result quality will be proportional to the percentage of those entities that meet the quality objectives. For example, a speech transcript will be the better the more words have been transcribed correctly. On the other hand, a painting consists of several graphical elements but its quality does not simply scale with the number of "good" elements.

4 Decision Matrix

The proposed decision matrix links the decision criteria to a set of five quality QA for pServices which are variations of the simple MV and review approaches:

Majority vote (MV): The basic majority vote mechanism as described above.

Validation review (VR): Simple review mechanism for which a reviewer or a group of reviewers provides a binary rating whether to accept or reject a result. VR does not improve the quality of a result but only acts as a filter which can be passed by a result or not. When combining it with a feedback loop, VR can help to raise the skill level of workers.

Majority vote with comparing review (MVCR): An extension of MV in which a reviewer compares results delivered by multiple workers and groups them into sets of equivalent results. The reviewer basically performs the aggregation which is done automatically for deterministic tasks in case of MV.

Improving review (IR): A review approach in which the reviewer not only rates the result delivered by another worker, but spends additional effort on improving the result: A chain of reviewers improve the result step by step until it meets the requirements of the requester. A similar approach is used by CastingWords (www.castingwords.com) for speech transcription.

Majority vote with improving review (MVIR): A combination of MV and IR where multiple workers are delivering results which are then improved by a chain of multiple workers, each seeing the results delivered by the previous ones. The approach is a pService implementation of a Delphi study which is known to have the potential to deliver high quality forecasting results [4].

At the top of the decision matrix (Figure 2) we differentiate between deterministic and non-deterministic tasks. MV can only be applied to deterministic tasks but depending on the execution effort for the task, MV might even not be recommended for those. For low quality needs, the basic recommendation is to use MV if the execution effort similar to the validation effort and to use the VR approach in all other cases. If the quality needs are high and the task is deterministic, the MV approach can be used and is recommended as long as the execution effort is not too high. If it is high, the IR approach is recommended which can even be applied to non-deterministic tasks. For those, also the MVIR approach can be used which can be assumed to achieve an even higher quality, because it massively increases the interaction between the workers.

Criterion	Characteristic
Determinacy of the task results	Deterministic: There is a well defined optimal result for the task i.e. two workers who perfectly meet the task objectives will pass exactly the same result, or the results can be automatically transformed (normalized) into a well defined optimal result. Non-deterministic: There is no well defined optimal result for the task.
Execution versus validation effort	Similar: Task can be executed as quickly as its result can be validated. High: Execution effort for the task clearly exceeds validation effort.
Required level of quality	Low: A single qualified worker is able to meet the quality needs. High: Quality needs exceed the capabilities of a single worker. The "wisdom of the crowd" must be leveraged.
Number of equivalent quality relevant entities	Low: Result is only made of one or few equivalent quality relevant entities. High: Result is made of many equivalent quality relevant entities.

Required level of quality		# of equivalent quality relevant result entities		Deterministic task	
				yes	no
low	high	Execution vs. validation effort		Validating review	
		similar	high		
high	low	Majority vote		Majority vote with comparing review	
				Improving review	

Fig. 1. Decision criteria for selection of an adequate quality assurance mechanism for a given type of pService

Fig. 2. Decision matrix for choosing adequate quality assurance approaches for pServices

5 Conclusion and Future Work

We have provided an initial sketch for a decision matrix which aims to identify an adequate quality assurance approach for a given type of human intelligence task (HIT). The matrix is based on four simple criteria that can be easily determined based on the characteristic of the HIT type, effort estimations as well as the quality needs. The decision matrix maps those criteria to a set of five general quality assurance mechanisms. In a next step, we plan to evaluate and concretize our concept based on a detailed analysis and comparison of the proposed mechanisms.

References

1. Barr, J., Cabrera, L.F.: AI gets a brain. *ACM Queue* 4(4), 24–29 (2006)
2. Kern, R., Bauer, C., Thies, H., Satzger, G.: Validating results of human-based electronic services leveraging multiple reviewers. In: *Proceedings of the 16th Americas Conference on Information Systems (AMCIS)*, Lima, Peru (2010) (forthcoming)
3. Kern, R., Zirpins, C., Agarwal, S.: Managing quality of Human-Based eServices. In: Feuerlicht, G., Lamersdorf, W. (eds.) *Service-Oriented Computing - ICSOC 2008 Workshops*. LNCS, vol. 5472, pp. 304–309. Springer, Heidelberg (2008)
4. Rowe, G., Wright, G.: The delphi technique as a forecasting tool: issues and analysis. *International journal of forecasting* 15(4), 353–375 (1999)
5. Snow, R., OConnor, B., Jurafsky, D., Ng, A.Y.: Cheap and fastbut is it good? evaluating non-expert annotations for natural language tasks. In: *EMNLP 2008: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 254–263. ACL, Stroudsburg (2008)
6. Sorokin, A., Forsyth, D.: Utility data annotation with amazon mechanical turk. In: *CVPRW 2008: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8. IEEE Computer Society, Washington (June 2008)
7. Surowiecki, J.: *The Wisdom of Crowds*, 1st edn. Doubleday, New York (2004)

Collaborative Workforce, Business Process Crowdsourcing as an Alternative of BPO

Gioacchino La Vecchia¹ and Antonio Cisternino²

¹ CrowdEngineering inc., 440 N.Wolfe Rd., Sunnyvale CA 94085 United States
gio@crowdengineering.com

² Dipartimento di Informatica, L.go B.Pontecorvo 3, 56127 Pisa
cisterni@di.unipi.it

Abstract. Crowdsourcing is the act of outsourcing activities to networked people. This paper presents Business Process Crowdsourcing, an alternative to Business Process Outsourcing where crowd activities are coordinated, work force contributions not wasted and final result guaranteed. The positioning paper shows how to transform canonical business processes in crowdsourced business processes where Web 2.0, social networks, and business process management are combined to deploy business critical process to the Internet, getting the same level of quality and control of traditional outsourcing approaches with conventional workforce.

Keywords: crowdsourcing, social production, collaborative intelligence, business process management.

1 Introduction

Crowdsourcing, term coined by Howe [2], is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call [1].

Crowdsourcing is rapidly becoming a common tool for addressing problems in organizations and businesses, finding always new applications and often exploiting social network to accomplish tasks in a human-wise way rather than resorting to smart algorithms. The technique has been applied with special purpose systems since 2001, for instance to ask hobbyist astronomers to classify Mars craters [3], though it slowly emerged as a more general pattern. Wikipedia [4] has shown how a product built through a collaborative effort by thousands of spontaneous contributors can surpass a linearly organized process performed by professional editors over several years, affecting significantly the encyclopedia business. Crowdsourcing has also been used in several applications, and many game with purpose are played today to exploit the work otherwise wasted in gaming. Fold It!, for instance, is a game with purpose [5] that already contributed to define and implement new synthetic proteins. Among several other areas of application, it is worth citing R&D managed as open calls over the Internet like Innocentive [6], product design or marketing and communication concept development like Threadless [7], new business concepts development like Quirky [8], product and service testing like uTest [9]. Amazon has been one of the

first actors in the business market that tried to exploit the workforce in a more systematic way with the Mechanical Turk [10], a system introducing a crowdsourcing task marketplace for applications that need many contributors.

Crowdsourcing problem can be abstracted as follows: a task T can be split into a number of sub-tasks t_i that can be assigned to one or more workers w_i taken from a set of users W . The crowdsourcing system is responsible to generate the associations (w_i, t_j) so that the task T can be accomplished ensuring that sub-tasks are completed, re-scheduling in case of failure. A *special purpose* crowdsourcing system is a system $C_T(W)$ designed specifically to associate workers to tasks for a specific task T . A *general purpose* crowdsourcing system $C(T, W)$ is capable of orchestrating and book keeping the whole process being parametric in both T and W . A set of constraints K , may permit or forbid specific associations among sub-tasks and workers (for instance because a certain micro-task requires a specific knowledge). The various crowdsourcing systems are characterized in the way C , T , K , and W are chosen; for instance, if workers are not trusted the system may associate more workers with the same sub-task, adopting a majority vote among the results produced.

Crowdsourcing is the natural evolution of outsourcing in the era of social networks giving better economical returns also where outsourcing is already in place, enabling also small/medium enterprises and small institutions to get benefits from crowdsourcing internal processes. Several existing vertical crowdsourcing sites have not been able to fully address the market of outsourced or to-be-outsourced business processes. The main reason for this partial result is to be found in the reluctance of large enterprises in outsourcing activities, without maintaining the control of the process because this could seriously compromise their core business.

2 Today's Crowdsourcing Scenario

Crowdsourcing is today well known as a cheap alternative to hire professionals; it may appear in the form of a contest to which several apply to propose their solution or their contribution, in answer to a call over the Internet. In marketing applications creative professionals propose a logo or an idea for an advertising spot and a group of experts, usually company internals, decide who the winner is.

This type of crowdsourcing, which we define "*contest*", is clearly a waste of workforce: for a single price winner the work of many others is overlooked. It could work only assuming that large crowds of contributors compete for a limited numbers of contests. Besides, if the number of tasks to outsource is large, talented people will unlikely waste their time with a low chance to get some reward.

Another kind of crowdsourcing that is becoming popular is the "*task marketplace*", like Amazon Mechanical Turk, where several simple tasks are posted by companies or individuals. These tasks are usually repetitive actions like tagging photos or moderating contents, or simple parts of a larger problem like translating a section of a document or answer questions of a questionnaire. Because of the simple task nature, the reward in these cases is low since it doesn't require special abilities, thus the set W is large implying a large offer of workers.

A family of crowdsourcing systems that is successfully working is the one in which the set W is constituted by experts are associated with complex problems submitted by users and the system associate tasks attempting to find the best professional

to fulfill the task at the lowest cost. For professionals this is a great resource to find new assignments and be paid organizing autonomously their work; we will refer to this model as “*bid*”.

3 Crowdsourcing and BPO

Is the crowdsourcing, as we know it today, a real alternative to outsourcing for large enterprises? Let's take in consideration the crowdsourcing scenarios we described in the previous section from the point of view of a large enterprise or institutions willing to crowdsource their internal business processes.

The *contest crowdsourcing* model is something that applies to specific niches of business processes. Let us consider as example a marketing department: a contest can be started on a company's own site or in a crowdsourcing site to select a new logo or a new design for a package; while the creative work is performed by crowdsourcing contributors a lot of work is left to the company; in the first place the selection process and the winner evaluation. Often this type of crowdsourcing is used to generate buzz and to stimulate discussions about a company or a brand and so enable cheap user generated marketing. However, this approach is not cost effective to handle complex internal business processes.

Task marketplace crowdsourcing is a great resource for processes compatible with a divide et impera solution approach. If you have a document to translate you can split it in several sections and post each section as a task to complete (to translate) and wait for the contributions. This works well in phase of preparation for simple processes where the split in single tasks is relatively simple. But still a significant effort is needed in order to monitor the contributors' work, the completion of the tasks, merging of the single parts, and the control of the quality of the final deliverable. It could help in some cases but again if the complexity of the business process or the need of a fine control and quick reaction to some event apply then it would be a mess to manage it in such a way.

Bid crowdsourcing is something not new to large companies; the main difference is that now there exist communities of experts, while some time ago companies already collaborating with the enterprise looking for help were invited to submit bids. In this kind of crowdsourcing the effort is both in the preparation, execution, selection, and acceptance phases. In a software project, for instance, it must be paid attention when writing the requirements, in order to avoid misunderstanding in the economic evaluation; moreover each submitted proposal should be carefully read.

In summary we think that currently available crowdsourcing models are able to address only a small percentage of the business process of an enterprise where the involvement of the crowd could bring benefits. The reasons are that core business processes are usually quite complex, need to be integrated with other company's business processes and eventually with software platforms supporting them. So orchestration of resources, control of the processes in term of time, delivery and quality are key requirements to enlarge the usage of crowdsourcing in the enterprise. So a new model is necessary to fully exploit the power of crowdsourcing and we'll call it *Business Process Crowdsourcing*.

4 Crowdsourcing in Customer Service

In a traditional customer service model requests are managed by internal Customer Assistants or by Outsourcing operators. When a new request R is asked it is assigned or taken in charge by a worker w_i . Assignment to a specific worker is performed in the most advanced call centers based on the type of request, expertise of the worker and value of the customer. Some of the typical challenges for a contact center are to maintain a level of expertise among assistants good enough to properly answer request at the first contact and to adequately size the workforce to face the intraday volume of requests incoming.

Crowdsourcing can help a lot here, the expertise distributed among users, in particular among customers of the same company providing support, is large enough to provide a good level of support. A widely used rule says that when you have an online community you can expect 90% of users to be passive, 9% to be intermittent contributors and 1% to be heavy contributors [11]. Online members are already a fraction of total customer base; you can expect to have between a 10% and 50% of total customer base to use online channels depending on the industry and the affinity of the service or product supported with an online presence. So if we take in consideration a 10 million users Telco customer base and we apply a conservative rate of 20% of users going online we have a 2 million community where 200k users contribute and 20k among them are heavy contributors.

A successful application of crowdsourcing here can change the economics of the call center. A 20% of deflection of requests from the contact center to the community is an extraordinary result for cost reduction. And by transforming voice calls requests in recorded messages for the community deflection rate can raise even more. At this point the challenge is not anymore in migrating users from voice to online channels but to make sure that community experts will be able to manage all those requests.

To achieve this result a fine grain crowdsourcing process must be in place deeply integrated with company enterprise platforms. The model applied must be very similar to the way complex enterprise processes are managed but considering the key differences of this workforce, because they are usually voluntary and must be rewarded, they are not under a contract but their performance must be evaluated, they are not respecting scheduled shifts of work but must be orchestrated and more their skill and ability to support must be evaluated day by day in order to involve the best(s) expert for each request. We'll define *Business Process Crowdsourcing* that crowdsourcing model taking in consideration all of these elements of the problem of outsourcing a complex internal business process to the crowd.

5 Cloud of Crowd

A natural convergence of crowdsourcing techniques is toward cloud computing. Cloud computing is a business model in which large computing infrastructures are made available to third party that have need for computing resources. Crowdsourcing shares many similarities with this business model and may be affected by the attention it will receive in the future. It is possible to define a cloud specifically designed for general purpose crowdsourcing in which the set of workers W is shared among different tasks T_k . In this way businesses may deploy new crowdsourcing services in an

elastic way, so that specific crowdsourcing services may be used without having to deploy a specific infrastructure.

If cloud services depend only from computational resources and software available on nodes, a cloud of crowd also depends on the know-how of the users working at the various sub-tasks, introducing constraints on the type of tasks that can be served by a specific cloud. Moreover, since the knowledge may vary among workers, it is responsibility of the system to find the optimal match between workers and sub-tasks and keep track of who did what.

Traditional outsourcing is a coarse grain model for externalizing a service reducing internal costs; however it imposes overhead because services are dimensioned to handle potential spikes. Cloud of crowd allows for a finer grain outsourcing mechanism, in which it is possible to track every single sub-task delivered introducing a pay-per use model. This is the dual problem with respect to micro-payments: work should be tracked with a very fine grained model, imposing that the tracking cost does not surpass the value generated by the worker. It should be noted, however, that human workers may perform tasks that computers may not be able to solve (for instance finding nicer objects of a given example) or solve it easily; therefore the value generated by a single worker may be relevant with respect to the system. Moreover, if the set W is large enough the probability that a sub-task is quickly assigned to a worker is high.

6 Summary

A new model of crowdsourcing is needed in order to graduate crowd involvement for enterprise business processes. Enterprise crowdsourcing is so possible but needs to be supported in the same way an enterprise business process is with all the distinctions considering the different nature of the workforce involved.

Deeply different from existing crowdsourcing models enterprise crowdsourcing must ensure that no resources are wasted, best contributor available is assigned to each single task, performances are carefully evaluated to build dynamic skill profiles and to reward contributors, a strong business process control ensure that unattended tasks are completed respecting the service levels in place, quality visibility points are constantly managed and monitored, involvement of the crowd and communication with them are facilitated in order to make the job easier and when possible amusing.

References

1. Crowdsourcing Blog, <http://www.crowdsourcing.com/> (Last accessed 20/6/2010)
2. Howe, J.: The rise of Crowdsourcing. WIRED Magazine (June 14, 2006)
3. Britt, R.R.: Nasa Wants you...to identify Martians craters (February 2001), http://www.space.com/scienceastronomy/solarsystem/click_workers_010202.html (Last accessed: 20/6/2010)
4. Wikipedia: Crowdsourcing, <http://en.wikipedia.org/wiki/Crowdsourcing> (Last accessed: 20/6/2010)
5. Fold.it!, <http://fold.it/> (Last accessed: 20/6/2010)

6. Innocentive, <http://www.innocentive.com/> (Last accessed: 20/6/2010)
7. Threadless, <http://www.threadless.com/> (Last accessed: 20/6/2010)
8. Quirky, <http://www.quirky.com/> (Last accessed: 20/6/2010)
9. UTest, <http://www.utest.com/> (Last accessed: 20/6/2010)
10. Amazon Mechanical Turk, <https://www.mturk.com/> (Last accessed: 20/6/2010)
11. Nielsen, J.: Participation Inequality: Encouraging More Users to Contribute, http://www.useit.com/alertbox/participation_inequality.html (Last accessed: 20/6/2010)

Connecting Smart Things through Web Services Orchestrations

Antonio Pintus¹, Davide Carboni¹, Andrea Piras¹, and Alessandro Giordano²

¹ CRS4 – Center for Advanced Studies, Research and Development in Sardinia
Edificio 1, Polaris, 09010 Pula (CA), Sardinia, Italy
{pintux, dcarboni, piras}@crs4.it

² Università degli Studi di Cagliari, Dipartimento di Ingegneria Elettrica ed Elettronica
piazza d'Armi, 09123 Cagliari, Sardinia, Italy
alegiordy@tiscali.it

Abstract. The Web of Things is an emerging scenario in which objects are connected to Internet and can answer to HTTP queries. To date, new applications in this field are mainly produced by designers and engineers while we claim that with simple and effective composition rules and easy-to-use building blocks, even users could invent new applications unforeseen by technology experts. In this paper, we describe a solution for modeling, implementing and running simple connections of smart things based on the point-click-and-compose paradigm. We envision a Service-oriented Architecture (SOA) where things are Web Services using WSDL standard and logical connections between things are modeled as Web Services orchestrations using the WS-BPEL language.

Keywords: Web of Things, SOA, WS-BPEL, Web services, orchestration.

1 Introduction

The Web of Things (WoT) is an emerging scenario in which every object is connected to a pervasive wireless/wired network and can communicate with other objects and services using HTTP-based protocols. Everyday surrounding objects like phones, domestic appliances, advertisement billboards, musical instruments become nodes of the WoT. Simple mechanisms to connect “things” can foster a huge number of unpredictable applications. Towards these objective, users, objects and networks are the ingredients to build a WoT in which users are also the programmers. This new vague must not be insulated from the existing Web ecosystem, rather the WoT should connect seamlessly to its existing declinations.

The Web has become a real platform for Service-Oriented Computing (SOC) and Service-Oriented Architecture (SOA) paradigms where complex distributed systems can be built through the use and composition of atomic, loosely-coupled, software modules called services. Undoubtedly, nowadays the Web Services (WSs) and related standards are the enabling technologies for SOC and SOA.

In this work we explore the feasibility of using the “service” abstraction to represent things in the WoT space and to draw a way to simply connect that things using well-defined patterns and services orchestrations, in particular extending the *pipe* paradigm to real objects, ideally mixing them with digital ones. Under some assumptions, our

system allows users to build networks of everyday objects. It relies on Web Services-based SOA languages and tools for the runtime composition of “things”. The paper is organized as follows: first we introduce the necessary assumptions and a taxonomy of things based on their connectivity, then the overall architecture is specified in terms of SOA patterns, finally a scenario of use and a concrete prototype are described and discussed.

2 Related Works

In this work, we provide a design based on WSDL and SOAP based WSs experimenting the direct generation of new process definitions according to user selection and pointing of real objects in the environment.

Other works rely on architectures using Web-based standards to connect devices and objects, but do not give users the chance to establish on-the-fly connections by selected-and-pointed objects. The SODA project [8] goes toward the definition of an architecture where devices are viewed as services in order to integrate a wide range of physical devices into distributed IT enterprise systems adopting a SOA. In similar way, the projects WS4D [24] and SOCRADES [7] apply SOA approach for embedded network.

In [12], authors consider RESTful [10] the only choice for a WoT architecture a cause of the programmatic complexity of WSs and according to their experience, not well suited for end-user to create ad-hoc applications. In special way because the discovery of WSs via UDDI is not suitable for sensors or devices because the UDDI-based discovery has not context information (e.g. where a sensor is placed). We bypass the discovery problem by the fact that services are discovered by users when they are close to an object using a proximity technology (the QR tags) and the programmatic complexity is totally hidden to end-users by the automatic generation of WS-BPEL executables.

Other works describe solutions based on pointing-and-clicking and proximity of users to get information by objects but they do not rely on Web standards to make concrete the virtual connection. They use different kinds of technologies to enhance objects and get their features: Bluetooth and RFID tags [20], Infra-Red (IF) tags read by smartphones [1] or by stylus and PDA (like GesturePen [21]), barcodes recognized by barcode readers (like AURA [4] and WebStickers [13]), 2D visual tags and smartphones [22]. In Cooltown [14], users get URLs containing thing’s information by several types of active and passive emitters called “beacons” and can create a sort of simple pipe of two modules sending that URL to one printers or projector with and embedded web server.

[5] depicts a similar interaction pattern even if system architecture and data formats are described at a general level while in this work we focus on architectural aspects with formal specification and adoption of SOA standards.

3 Issues and Assumptions on “Nature” of Things

In order to actively play a role in a pipe, our main assumption is that an object must be able to connect to the network and to run a WS stack. This general capability can be accomplished basically in two ways: the object itself is powerful enough to satisfy

the previous requirements or it have to be connected and “driven” by a proxy computer which satisfies the requirements.

For completeness, it is useful to explore how things can be classified into categories related to their connection and communication capabilities. The following list provides a classification of things:

- *Virtual Things*, like web sites, e-mail boxes and social networks, just to mention some. These objects can be easily wrapped and then referenced in a HTTP addressing space like resources (REST) or like services (WSDL) or they already provide such abstractions and interfaces.
- *HTTP-enabled Smart Appliances* that are already equipped with a network connection and a complete HTTP stack like wireless printers or networked screens or smartphones: potentially they don’t provide a WS stack so it is necessary to use a proxy for them (or to install a minimal WS stack in the device, where possible).
- *Internet-enabled Things* that are not equipped with a complete HTTP stack but can still communicate at the TCP/IP or UDP/IP level. For those objects is straightforward to build a HTTP wrapper and a WS stack as proxy.
- *Network-enabled Things* that cannot communicate over IP networks, but still can communicate with different protocols like ZigBee, Bluetooth or X10. For those objects a proxy can be deployed to present these objects in the HTTP addressing space also using WS technology standards.
- *Things not digitally enabled*, bare physical objects, for those a digital counterpart must be built and published online. For example, a real book has a virtual counterpart as a Web page in a online bookstore.

Taking into consideration the assumption and the classification of things made above we choose to adopt the Web Service Description Language (WSDL) as the formalism to describe what an object is able to provide. WSDL is powerful enough to describe the interfaces exposed by objects in a general manner without imposing the use of a particular set of names for operations and, moreover, adopting XML Schema for data types definition, it allows to rigorously describe that functional objects interface. In this way, an object can be considered as a SOAP-based WS. Another issue is related to the type of communications between objects and the definition or the adoption of a suitable related protocol. The design must take into account that objects can be different in the type of data exchanged, and can work in synchronous or asynchronous mode. For instance, an anti-thief sensor could launch asynchronous alarms, on the other hand a refrigerator could be synchronous inquired about the list of current stored foods. Both these interactions can be easily modeled and implemented using WSDL and adopting SOAP (Simple Object Access Protocol) over HTTP protocol for messages exchange.

Another type of logical connection we want include is the streaming of multimedia data. For instance, the user should be able to select a camera as MPEG stream source and a wall screen as MPEG stream sink. It is not convenient to embed multimedia streaming in SOAP messages, so another protocol should be used.

4 Vision Overview and Basic Concepts

In our vision, each thing in the WoT namespace is a real object/device thought as a process and specified as a WS, thus each thing publicly exposes its capabilities using standardized functional interfaces descriptions expressed using WSDL format and it is capable to communicate through the network using SOAP formatted messages over HTTP protocol. The main advantage of this approach consists in the fact that in this way each thing is able to expose its functionalities in a formal and precise manner, thanks to the WSDL format, including the defined data types described through XML Schema language. Of course, this assumption brings several issues related both to objects/devices capabilities and data type definition: these issues are faced in the next sections. The introduced service abstraction directly projects things into the SOA paradigm ecosystem enabling the possibility to build WoT novel applications using SOA and WSs concepts, technologies and composition languages and allowing to mix services of different “nature” such as real things and virtual modules, such as Web sites, for example. More in detail, because of things are represented as WS, we can explore the possibility to use existing standard for WS composition, such as orchestration ones, in order to describe and execute things connections and to manage communication between them. In particular, it is interesting to face the chance of using WS-BPEL 2.0 as composition and connection language. The Web Service Business Process Execution Language (WS-BPEL) [16] is an XML language for describing and executing WSs orchestrations where WSDL messages and XML Schema type definitions provide the data model used by processes.

In our approach, building WoT applications as SOA applications faces us with common issues and best-practices of SOA and to “classical” design patterns widely reported in literature, see [9] and [15] for instance. With respect to existing literature, we aim at further simplify the mechanism of composing “things” providing the concept of *pipe* as a process defined and implemented as a service orchestration based on WS-BPEL language.

A pipe is a logical connection between two things implemented as a process that retrieve data from one source, adapts and sends to a sink. The pipe design is inspired by Unix pipes and Yahoo! Pipes [17] and its goal is to have a practical tool that enables composition in a way as easy as sketching lines and box on a board.

In order to define the simple things connection patterns we introduce the following concepts:

- *thing*: the real or virtual device/object represented as WS and with its functional interface described by a WSDL document;
- *src (source)*: a generic thing functionality exposed as a WS operation in the WSDL document, which represents a functional capability of the thing returning a value due to its invocation;
- *snk (sink)*: a generic thing functionality exposed as a WS operation in the WSDL document, which represents a functional capability of the thing able to accept a value as input;
- *adapter*: a generic WS able to provide data transformation or data adaptation functionalities exposed as service operations;
- *pipe controller*: a WS interface provided by pipes which exposes VCR-like operations: start, pause and stop in order to allow to manage pipes execution;

- *processor*: it produces and consumes data. From the point of view of a source, it is a sink. While from the point of view of a sink is a source. Examples are online programs performing algorithms on inputs and producing outputs.

Processor must not be confused with adapters because processors are first class components in our architecture and are explicitly inserted by users, while adapters are components internal to the pipe management system and automatically invoked when it raises the need to convert a media type produced by a source to the media type consumed by a sink. In more formal description we can assume that a source is an operation with signature $src() \rightarrow data$, a sink is an operation with signature $snk(data) \rightarrow void$, and a processor is an operation with signature $f(data) \rightarrow data$.

A thing can have an arbitrary number of sources, sinks and processors. The notion of pipe can now be refined with the notation $pipe(x,y)$ where x is who produces data and y is who consumes data. If we consider a generic source src , a generic processor f , and a generic sink snk , with the above definitions the composition framework can be described as follows:

- $pipe(src,snk)$ is a pipe that cannot be composed any further.
- $pipe(src,f)$ is a pipe open on the right end and that can be seen as a source and further composed in another pipe $pipe(pipe(src,f), y)$ where y can be either a sink or a processor.

Given the former definitions we introduce the following *things connection patterns* modeled as pipes:

1. *synchronous pipe*: on an thing x is invoked an operation src , the result is *adapted* and then passed as an input to an operation snk exposed by a thing y ;
2. *asynchronous pipe*: the pipe registers itself as a listener for an event produced by an exposed operation src on thing x . When the event is fired, the data attached to the event is *adapted* and the sent to the snk operation exposed by the thing y ;
3. *streaming pipe*, a thing y receives from a thing x a stream of data (for instance an MPEG video from a camera to a screen). A suitable real time media transmission protocol should be used after a negotiation and handshaking phase between things. The pipe represents and enables the handshaking step.

To the previous connection patterns, dedicated to things connections drawing, we also introduce and add a fourth pattern, faced to pipes execution control:

4. *VCR pipes*: an established pipe between two things needs to be controlled and its execution managed. For these aims, this pattern claims to adopt a VCR-like functional interface, for each pipe, with three control operations: *start, pause, stop*.

5 Things Connection Patterns as WS-BPEL Processes

In previous section of this paper we introduced three things connection patterns using the pipe paradigm and another one for pipes execution control. This section explicates how these patterns are built.

Expressing pipes using WS-BPEL brings two main benefits to our vision: it is possible to associate a well-defined functional interface to each pipe in order to expose VCR-like functionalities, according to the *VCR pipes* pattern; it allows us to implement the pipes as WSs orchestration following the three enounced patterns.

For synchronous and asynchronous patterns we created two different WS-BPEL document templates which define all the required activities, message exchanged and service orchestration for the execution of each of them in a standard WS-BPEL engine. The synchronous pattern is a typical WSs orchestration scenario with a subsequent invocation of services. Asynchronous pattern basically is an orchestration in which the WS-BPEL document describes an asynchronous invocation of a service (the event producer) using a callback mechanism which invocation triggers an event causing a delivering of the event to the other service (the event listener). The streaming pattern uses WS-BPEL only for protocol negotiation and handshaking (mainly based on MIME-types) between the two services, in this way after these steps, the objects can instantiate streaming sessions in an independent way using the suitable chosen protocol.

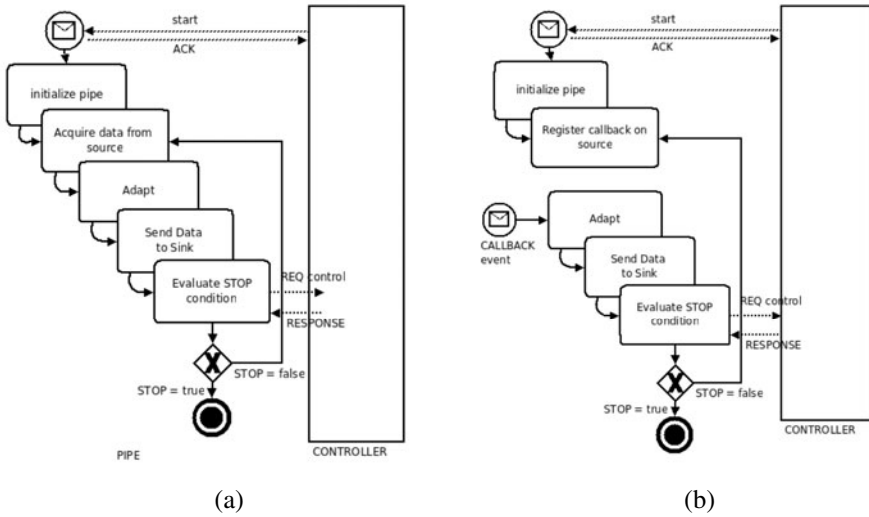


Fig. 1. BPMN-like processes for synchronous (a) and asynchronous (b) pipes. The CONTROLLER is the VCR pipe pattern implementation. In the asynchronous pipe, the callback endpoint is invoked when data is ready to be consumed. The streaming pipe pattern is not shown in figure.

From a WS-BPEL point of view the pattern (3) is equivalent to pattern (1) but the data exchanged are Session Description Protocols (SDP) instances and the streaming connection is completely delegated to endpoints. Because, the encapsulation of binary real time data inside SOAP messages is not efficient and it is preferable to use RTP or equivalent real time media transmission protocols and SOAP messages only to initialize the session for handshaking.

The difference between the synchronous and asynchronous patterns is that in the last the data source asynchronously emits a data value and requires that a callback endpoint is registered in order to consume data when ready.

6 Pipes User Interaction: Point, Click and Compose

Any graphical user interface (GUI) for computer application uses the Window-Icon-Mouse-Pointer interaction paradigm and the users have the habit to activate applications and open files by the sequence of actions defined as point-click-and-open. On the Web, it becomes point-click-and-download. We translate such a paradigm to the world of physical objects and their virtual connections with the point-click-and-compose interactive paradigm. Based on the fact that in a given situation, with some things and a given mood, a person would like to build new things just composing them. If we imagine the world as a giant sketch board we just want a way to draw a line from an object to another and build something useful as result.

In our point-click-and-compose paradigm, the user “points” an augmented object able to be a *source* and “clicks” on its direction. Pointing-clicking a second one that can be a *processor* or a *sink*, she “composes” the virtual connection. Until the last connected thing is a *processor*, the user can continue to point-click making virtual connection between the last selected thing and its previous, composing pipes more and more complex.

Objects are augmented with QR visual tags [6] directly stuck on the surface of physical things or displayed on a screen when associated to a virtual objects. In our model, each QR tags encodes the URL of the WSDL which describes the thing functionalities.

The point-click user action is then performed by means of a smartphone equipped with camera and software able to decode the QR tags. The general idea is that when the WSDLs are retrieved, the GUI on the smartphone enables the user to select and compose functionalities from diverse objects.

7 A Scenario

To test “nuts” and “bolts” of the architecture we have identified a scenario with multimedia and processing components: one TV screen, one webcam, and some image processing programs running on a PC.

The three objects are augmented with a QR tag (see Fig. 2). Not depicted in picture, but running behind the scenes, the *PipeController* module (explained in more detail in the architecture section) receives commands from the smartphone, generates on-the-fly the WS-BPEL document and deploy it to concretely launch the pipe. To make interaction easier in this first prototype, the application in the smartphone allows the user to select at once the three objects, then it suggests the possible interconnections and generates a pipe (either multiple or simple depending on the number of processors inserted in cascade). The webcam is embedded in a NetBook exposing the webcam's WSDL. It defines two different actions: a synchronous *snapshot*, and an asynchronous image event fired when a movement is detected in the scene (*moveDetected*). The TV screen

is a monitor connected to a PC not showed in the figure. Inside the piping are inserted two different processors that perform respectively a *color-to-gray* conversion with synchronous invocation, and a *supersampling* with asynchronous invocation. The different combinations that have been tested are:

- $pipe(snapshot,TVsink)$
- $pipe(pipe(snapshot,color-to-gray),TVsink)$
- $pipe(pipe(snapshot,supersampling),TVsink)$
- $pipe(pipe(pipe(snapshot,color-to-gray),supersampling),TVsink)$
- $pipe(pipe(pipe(snapshot,supersampling),color-to-gray),TVsink)$

The same combinations have been tested replacing the synchronous source *snapshot* with the asynchronous one *moveDetected*.

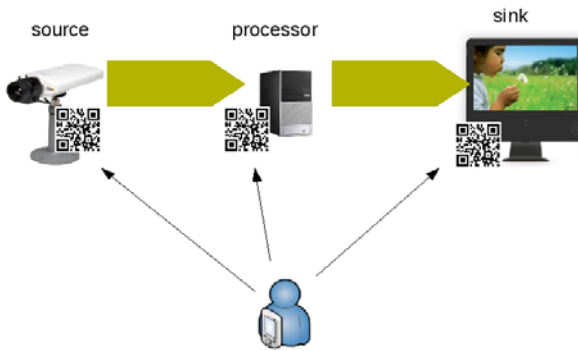


Fig. 2. Simple scenario with multimedia and processing components

8 The Prototype

In SOA applications one of the main issues are related to service publication and service discovery, often faced using registries and directory mechanisms (i.e. UDDI [23]). We adopt a “zero discovery” approach in which the services are real things in an environment and it is the user with the point-click-and-compose paradigm to choose which things she wants to connect. For this aim the user is assisted by an Android [2]-based smartphone application described later in this section.

The architecture of the prototype is composed of several modules (see Fig. 3). The things are represented by WSs. The pipe execution engine is a standard *BPEL Engine* (Apache ODE [3]) deployed in a Glassfish Application Server [11] instance. The central role is played by the *PipeController*, a software module which allows the server-side pipes creation, their management and control communicating both with the *BPEL Engine* and application client. The *PipeController* is able to compose and deploy at run-time a WS-BPEL document representing a generic pipe given the WSDL URLs of the things. The composition is driven by the functionalities, the exposed data types and the selected pattern.

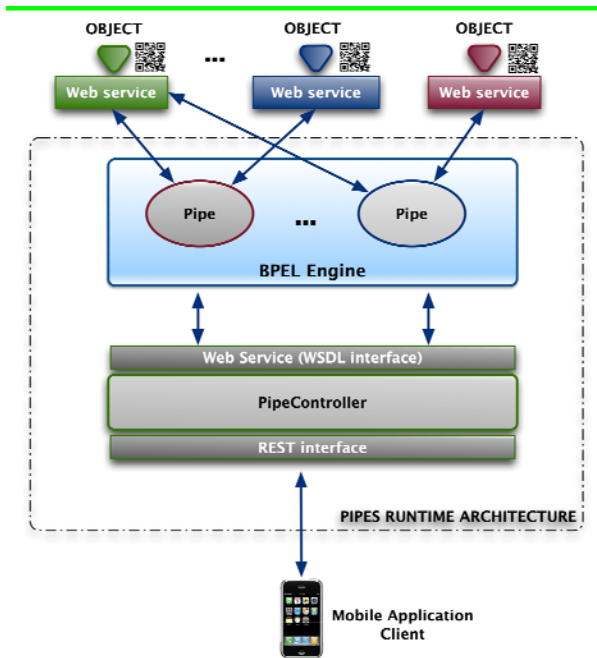


Fig. 3. General pipes run-time architecture. Objects are instrumented by WSs and pipes are implemented as WS-BPEL processes created and deployed at runtime. A mobile phone is used to point objects in the environment and to retrieve the WSDL specification.

Moreover, it exposes two communication interfaces:

- WSDL, in order to be queried by the running pipe process in order to check if a stop or pause status has been requested by the application client;
- REST interface which allows the application client to request a pipe composition, to start it and to control the pipe execution running VCR-like commands.

In our scenario, the REST style has been chosen in order to simplify the communication between the client on smartphone and the server-side modules. With the current prototype, by the mobile application the user collects in any order the objects, and related WSDLs could be used in a pipe. Given the WSDLs to the *PipeController*, it provides to the Android application all possible pipes through predefined data types inferences and user selects the own to run.

9 Future Works and Conclusion

In our work each "thing" is instrumented by a WS and its features are thus expressed by WSDL instances and translated into a list of actions available on the smartphone application. In this way, users are able to drive the generation of WS-BPEL at runtime and to create new executable processes by the point-select-and-compose interaction.

The overall design results well conceived for the transmission of “data as documents” between different objects while data streaming is less supported by the WS stack and SOAP is only used for exchanging session descriptions and that commuting to other protocols in the communication stack. Languages like SDP are suitable for multimedia but not for describing data streaming from a controller peripheral to a controlled entity like a PC. The choice to model things like opaque components able to perform operations poses some issues in the seamless connection with other web resources. It is clumsy to make a pipe having as endpoint a web page or a RSS feed because even if these are digital objects, they need to be wrapped by a WSDL interface and decorated with a service implementation.

Modeling the WoT as a network of intercorrelated processes presents some advantages. The use of formal methods like process algebras can aid the specification of WSs and their orchestration, similar results have been obtained in [19] and thoroughly discussed. Such an algebra is under development and will formalize pipes connections and flow control. They will be developed and integrated in next versions.

WS-BPEL, SOAP and WSDL 1.1 are the standards given the possibility to on-the-fly compose and orchestrate services. At the moment, RESTful API can not be applied because WS-BPEL does not support WSDL 2.0. RESTful systems are somehow “lighter”, simple to realize and to test. When RESTful API will be described by a formal specification and integrated with SOAP operation, we will perform a deeper investigation devoted to understand which is the best approach to build a WoT. Also will be interesting to explore the integration of our work with other WoT protocols and applications (see [18] for example).

Regarding to user interaction, we conclude that building a pipe between two objects results as a straightforward task and the use of QR has revealed to be a practical choice very easy to implement and quite easy for users to manage. Composing multiple pipes with processor in cascade is somehow less intuitive and requires the user to know how the underlying process is created. It is under development a feature that allow the user to bookmark objects' operations in the smartphone and to recall these when he/she wants to build a pipe in the future. Nevertheless, these conclusions on human interaction are merely preliminary and based on few test users, more accurate tests will be performed on interconnected pipes.

References

1. Ailisto, H., Pohjanheimo, L., Vällkynen, P., Strömmer, E., Tuomisto, T., Korhonen, I.: Bridging the physical and virtual worlds by local connectivity-based physical selection. *Personal Ubiquitous Comput.* 10(6), 333–344 (2006)
2. ANDROID, <http://www.android.com/>
3. Apache ODE, <http://ode.apache.org/>
4. Bernheim Brush, A.J., Combs Turner, T., Smith, M.A., Gupta, N.: Scanning objects in the wild: Assessing an object triggered information system. In: Beigl, M., Intille, S.S., Rekimoto, J., Tokuda, H. (eds.) *UbiComp 2005. LNCS*, vol. 3660, pp. 305–322. Springer, Heidelberg (2005)
5. Carboni, D., Zanarini, P.: Wireless wires: let the user build the ubiquitous computer. In: *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pp. 169–175 (2007)

6. Denso Wave, About QR Code,
<http://www.denso-wave.com/qrcode/index-e.html>
7. de Souza, L.M., Spiess, P., Guinard, D., Kohler, M., Karnouskos, S., Savio, D.: Socrades: A web service based shop floor integration infrastructure. In: Floerkemeier, C., Langheinrich, M., Fleisch, E., Mattern, F., Sarma, S.E. (eds.) IOT 2008. LNCS, vol. 4952, p. 50. Springer, Heidelberg (2008)
8. Deugd, S.D., Carroll, R., Kelly, K., Millett, B., Ricker, J.: SODA: Service Oriented Device Architecture. *IEEE Pervasive Computing* 5, 94–96, c3 (2006)
9. Erl, T.: SOA Design Patterns. Prentice Hall/PearsonPTR (2008), ISBN: 0136135161
10. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine (2000)
11. Glassfish - Open Source Application Server,
<https://glassfish.dev.java.net/>
12. Guinard, D., Trifa, V., Pham, T., Liechti, O.: Towards physical mashups in the web of things. In: Proceedings of INSS (2009)
13. Holmquist, L.E., Redström, J., Ljungstrand, P.: Token-based access to digital information. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 234–245. Springer, Heidelberg (1999)
14. Kindberg, T., Barton, J.J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, Places, Things: Web Presence for the Real World. *MONET* 7, 365–376 (2002)
15. Manolescu, D., Lublinsky, B.: Service orchestration patterns: graduating from state of the practice to state of the art. In: Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2005, San Diego, CA, USA, October 16–20, pp. 148–149. ACM, New York (2005),
<http://doi.acm.org/10.1145/1094855.1094907>
16. OASIS Web Services Business Process Execution Language Version 2.0,
<http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
17. Pipes: Rewrite the web, <http://pipes.yahoo.com/pipes/>
18. QuadraSpace, <http://www.quadraspace.org/>
19. Salaun, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. *International Journal of Business Process Integration and Management* 1(2), 116–128 (2006)
20. Salminen, T., Hosio, S., Riekkilä, J.: Enhancing Bluetooth Connectivity with RFID. In: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 36–41. IEEE Computer Society, Los Alamitos (2006)
21. Swindells, C., Inkpen, K.M., Dill, J.C., Tory, M.: That one there! Pointing to establish device identity. In: Proc. 15th annual ACM symposium on User interface software and technology, pp. 151–160. ACM Press, New York (2002)
22. Toye, E., Madhavapedy, A., Sharp, R., Scott, D., Blackwell, A., Upton, E.: Using camera-phones to interact with context-aware mobile services. Technical report, UCAM-CL-TR-609, University of Cambridge (2004)
23. UDDI Specifications, OASIS,
<http://www.oasis-open.org/committees/uddi-spec/doc/tcpspecs.htm>
24. Web Services for Devices-WS4D, <http://www.ws4d.org/>

Mashing Up Your Web-Enabled Home

Dominique Guinard

¹ Institute for Pervasive Computing, ETH Zurich

² SAP Research CEC Zurich
dguinard@ethz.ch

Abstract. Many efforts are currently going towards networking smart things from the physical world (e.g. RFID, wireless sensor and actuator networks, embedded devices) on a larger scale. Rather than exposing real-world data and functionality through proprietary and tightly-coupled systems, several projects suggest to make them an integral part of the Web. As a result, smart things become easier to build upon and the scalability and evolvability of the Web can be leveraged. In this paper we look at “physical mashups”, where tech-savvy create lightweight, ad-hoc applications on top of smart things just as they currently create Web 2.0 mashups. We present the early version of a mashup framework that provides services for composing things and illustrate this by means of two mashup editors.

1 Introduction

A seamless integration of smart things to the Internet brings them one step closer to the Web. Much is to gain from Web integration as it drastically eases the usually rather tedious development of applications on top of embedded computers.

The first step towards this goal is to bring things to the Internet, either directly using technologies such as IPv6 (lowpan) [1] or through translation gateways [2] (reverse proxies, in Web terms). Further, Web servers are needed on devices to truly achieve integration on the application, i.e. Web layer. Well established research in embedded Web servers and recent research in tiny Web servers makes this possible [3]. However, having a Web server running on an appliance is only the first step. Indeed, there is still a need for modeling access to the smart thing functionality in a Web-oriented manner [4]. Several projects, sometimes referred to as “Web of Things” projects, advocate using the REpresentational State Transfer (REST) architectural style for interfacing with smart things [2][5][6].

The Web enabling of smart things also delivers more flexibility and customization possibilities for end-users. As an example, tech-savvy, i.e. end-users at ease with new technologies, can also build small applications on top of their devices. Following the trend of Web 2.0 participatory services and in particular Web mashups, users can create applications mixing real-world devices such as home appliances with virtual services on the Web. This type of applications is sometimes referred to as *physical mashups* [2]. As an example, a HiFi system could be connected to Facebook or Twitter in order to post the songs one mostly listens to. On the Web this type of small, ad-hoc application is usually created through a mashup editor, e.g. Yahoo Pipes, which is a Web platform that enables people to visually create simple rules to compose Web sites and data sources.

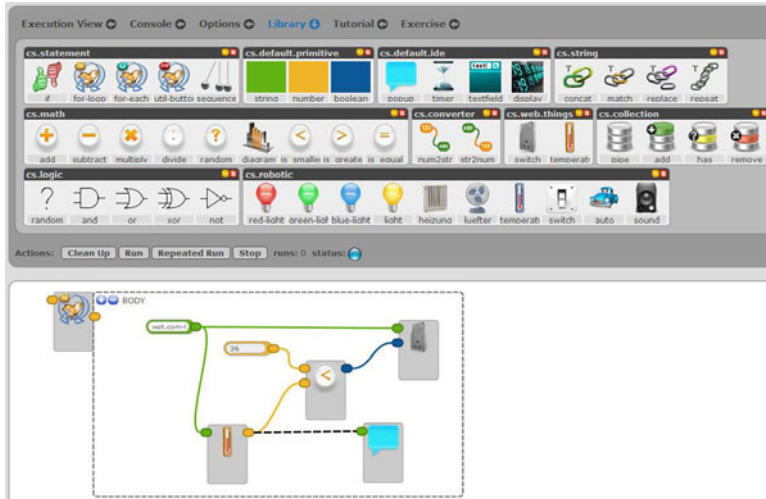


Fig. 1. A simple physical mashup using an adapted version of the Clickscript mashup editor

These concepts can now also be applied to empower users to create small applications on top of their smart things.

Our contribution in this paper is to outline a mashup framework that would support the creation of mashup editors for Web-enabled smart things.

2 Physical Mashups in the Web of Things

To illustrate this, we Web-enabled several devices. We first developed a Web API for Sun SPOT sensor nodes [7] which offers a broad range of sensors (temperature, acceleration, light) and actuators (leds, buttons, analog I/O ports). Requests for these sensors and actuators are formulated using URIs. For instance, typing a URL such as <http://.../spot1/sensors/temperature> in a browser, requests the resource *temperature* of the resource *sensors* of *spot1* with the HTTP method *GET*. Similarly, we Web-enabled the Ploggs energy meters [2], which allows us to control (turn on/off) and monitor the energy consumption of plugged appliances (e.g. a lamp) in a RESTful manner. Further details about the Web-enabling process is provided in [2]. To better understand the concepts and requirements of a physical mashup editor, we began by testing and extending an existing mashup framework: the Clickscript project [8]. Clickscript is a Firefox plugin written on top of an AJAX library that allows people to visually create Web mashups by connecting building blocks of resources (Web sites) and operations (greater than, if/then, loops, etc.). Since it is written in Javascript, it cannot use resources based on low-level proprietary service protocols. However, it can easily access RESTful services. Thus, it is straightforward to create Clickscript building

¹ www.ploggs.co.uk

² www.clickscript.ch



Fig. 2. Architecture of the physical mashup framework and screen-shot of a mobile mashup editor

blocks representing smart things. We used this approach to create building blocks for all the sensors and actuators of the Ploggs’ and Sun SPOTS’ RESTful APIs. As an example, we created a mashup shown in Figure 1. This mashup gets the room temperature by GETting the temperature resource of a RESTful Sun SPOT. If it is smaller than 36 degrees Celsius, it turns off the air-conditioning system plugged to a RESTful Plogg.

2.1 Requirements of a Physical Mashup Platform

While this integration with Clickscript worked well, the tool is not meant for creating physical mashups and has some shortcomings that helped us to identify a number of requirements for a physical mashup platform:

R1: Support for several UIs and modalities: Since interactions in the physical world usually occur beyond the desktop metaphor, it is hard to imagine creating a “one-size-fits-all” mashup editor. We rather suggest creating a mashup framework on which multiple editors (e.g. mobile, tablet, etc.) can be built. This also lets users create their mashups locally, e.g. on a mobile phone, and export them to a more robust framework for execution.

R2: Support for describing things: Manually creating building blocks for each smart thing is not scalable to the diversity of the physical world. Thus the need for the mashup framework to support partially automated integration (aka service discovery) of things.

R3: Support for event based mashups: While the pull-based interaction model of the Web is fine for controlling smart things, it is suboptimal for monitoring them (e.g. sensors). Thus, the mashup framework should be able to handle events coming from smart things.

2.2 A Framework for Physical Mashups Editors

Based on these requirements we present our first prototype of a mashup framework for the Web of Things. As shown in figure 2, the system is composed of four main parts. First, we have Web-enabled devices and appliances such as the RESTful Ploggs and

Sun SPOTs. In our prototypes, we tag them with small 2-D bar-codes (QR codes) in order to ease the identification of their root URL with mobile phones. We then have “virtual” services on the Web such as Twitter, Google Visualization API, XMPP, etc. In the middle, the mashup server framework allows to compose services of different smart things as well as virtual services on the Web. It is in charge executing the workflows created by end-users in their mashup editors. It discovers, listens, and interacts with the smart things over their RESTful API. The last component are the mashup editors. These applications allow users to create their mashups in a very easy manner. The mashup server framework is based on an open-source workflow engine called Ruote³. We extended Ruote in several ways in order to support our requirements. First, to fulfill R2, we created an RDFa schema and a compound Microformat that enables the discovery of smart things. Given the root URL of a smart things, the framework uses the RDFa or Microformat data to create a re-usable component representing the things’ functionality. Then, by using Web Hooks (i.e. callback URLs), we enabled smart things to push data to the mashups, fulfilling R3. Finally, we created a RESTful API that allows creating mashup editors on top of the framework, fulfilling R1.

2.3 Energy-Aware Mobile Mashup Editor

In order to prototype the use of the framework, we created a mashup editor on the Android Platform. The mobile editor offers wizards for creating simple mashups that are then exported to the mashup framework. Users first have to select the smart things they want to include in the mashup. They do this simply by scanning the things’ bar-codes using the phone’s camera. These codes are pointing back to the root URLs of the smart things’ RESTful APIs where the RDFa or Microformat description of the device and its services can be found. Users then setup the rules they want to enforce and the virtual services they want to interact with. As an example, we created a mashup that switches on appliances, e.g. turning the heating up, whenever your phone detects that you are coming home (based on your GPS trace). The right part of Figure 2 shows one of the wizard screens used to create this mashup.

2.4 Conclusion and Future Work

In this paper we introduced the idea of physical mashups and illustrated it by presenting the early prototype of a framework for creating editors of physical mashups. We briefly introduced a requirements for physical mashups. While we began addressing these, there is still work to be done to fully tackle them. As mentioned earlier, HTTP was designed as a client-server architecture where clients pull data. This interaction model works fine for control-oriented applications, however, monitoring-oriented applications are often event-based and thus Web-enabled smart things should also be able to push data to mashup frameworks, rather than being continuously polled. Overcoming the client-server architecture is now a core research topic in the Web community. Standards such as HTML 5 (and its “Web Sockets”) are also going towards asynchronous bi-directional communication.

³ openwferu.rubyforge.org/

Another major challenge for mashups in a Web of Things is search and discovery of smart things. Consider billions of things connected to the Web. Searching for the things to be included in mashups becomes very complicated. Thus, smart things also need to have means to describe themselves in order for users to find them and for mashup frameworks to be able to automatically create wrapper for them. How to describe a thing on the Web so that both humans and applications (e.g. mashup frameworks) can understand what services it provides? Here again, the recent developments in Microformats, RDFa and HTML 5 Microdata seem to be going in a promising direction to support describing tomorrow's Web of Things.

References

1. Yazar, D., Dunkels, A.: Efficient application integration in IP-based sensor networks. In: Proc. ACM of the First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys), Berkeley, CA, USA (November 2009)
2. Guinard, D., Trifa, V., Wilde, E.: Architecting a mashable open world wide web of things. Technical report, ETH Zurich (2010), <http://tinyurl.com/wotarchi>
3. Duquennoy, S., Grimaud, G., Vandewalle, J.J.: The Web of Things: interconnecting devices with high usability and performance. In: 6th International Conference on Embedded Software and Systems (ICESSE 2009), Hangzhou, Zhejiang, China (May 2009)
4. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, places, things: web presence for the real world. *Mob. Netw. Appl.* 7(5), 365–376 (2002)
5. Drytkiewicz, W., Radosch, I., Arbanowski, S., Popescu-Zeletin, R.: pREST: a REST-based protocol for pervasive systems. In: 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 340–348 (2004)
6. Luckenbach, T., Gober, P., Arbanowski, S., Kotsopoulos, A., Kim, K.: TinyREST - a protocol for integrating sensor networks into the internet. In: Proc. of the Workshop on Real-World Wireless Sensor Network (SICS), Stockholm, Sweden (2005)
7. Guinard, D., Trifa, V., Pham, T., Liechti, O.: Towards physical mashups in the web of things. In: Proc. of the 6th International Conference on Networked Sensing Systems (INSS), Pittsburgh, USA (June 2009)

A Triple Space-Based Semantic Distributed Middleware for Internet of Things

Aitor Gómez-Goiri and Diego López-de-Ipiña

DeustoTech, Deusto Institute of Technology
Avda. Universidades 24, 48007 Bilbao, Spain
{aitor.gomez,dipina}@deusto.es
<http://www.morelab.deusto.es>

Abstract. In the Internet of Things several objects with network capabilities are connected over a self-configured local network with other objects to interact and share knowledge. In this context, the Triple Space approach, where different processes share common semantic knowledge, seems to fit perfectly. In this paper we present our progress towards a semantic middleware which allows the communication between a wide range of embedded devices in a distributed, decoupled and very expressive manner. This solution has been tested in a stereotypical deployment scenario showing the promising potential of this approach for local environments.

Keywords: triple space, ubiquitous, mobile, embedded, semantic.

1 Introduction

Triple Space computing is a coordination paradigm based on tuplespace-based computing. In tuplespace computing the communication between processes is performed by reading and writing data structures in a shared space, instead of exchanging messages [4]. The Semantic Web vision aims to offer machine-understandable persistent data forming a network for machines instead of the current World Wide Web which is more human-centered (web services offer remote functionality to machines, but they are not really Web-based since they are message exchange-driven). Triple Space (TS) computing performs a tuplespace based communication using RDF triples, in which the information unit has three dimensions: "subject predicate object", to express this semantic data.

A Triple Space offers different type of autonomy which is not reachable with message exchange-driven communication, such as reference autonomy (the processes can communicate without knowing anything about each other), time autonomy (because of the asynchronous communication) and space autonomy (the processes can be executed in very different computational environments).

In context aware environments, a lot of devices communicate with each other and share their state in order to perform actions over the environment. Different approaches to define and store context data have been presented in several works [12], coming to the conclusion that ontology based models are the most

expressive models and fulfil most of the requirements of these environments. On the other hand, there are different context management models such as widgets, networked services or blackboard model [14]. The blackboard model post messages into a share media, which usually is stored in a centralized server. Triple Space not only expresses knowledge using the semantic model, but it also uses a similar mechanism to blackboard model, but in a decentralized fashion. The work presented in this paper describes an decentralized implementation over devices with limited computational resources.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 details our solution. Section 4 presents an experimental environment. Section 5 examines the results of using the proposed middleware. Finally, Section 6 concludes and outlines the future work.

2 Related Work

A number of approaches in the field of semantic tuplespace exist [11]. Conceptual Spaces, or cSpaces, were born to study the applicability of semantic tuplespaces to different scenarios including ubiquitous computing [10]. In cSpace the tuples have 7 fields, and in one of them the data can be expressed using first-order logic (ideally), description logics or RDF triples. Even when cSpace can be applied to ubiquitous systems, it is mainly based on client-server architecture which does not implement tuplespace paradigm itself in mobile peers and which restricts the reasoning process to few powerful devices.

Semantic Web Spaces [13] propose some new primitives to extend Linda coordination model. Its tuples are formed by RDF triples and a URI that identifies the tuple, it divides the tuplespace in nested and hierarchical spaces which may have access constraints and it allows RDFS reasoning. In Semantic Web Spaces two views on the coordination are defined: data view and information view. The first one contains syntactically valid RDF data and implements Linda primitives and the second one consistent and satisfiable data and offers some new primitives. The prototype was not designed to be run in small devices.

sTuples [5] was conceived by Nokia Research Center as a pervasive computing work. It provides description logics reasoning and a semantic extension of JavaSpace tuplespace middleware. In sTuples there are managers, which mediates between clients, available services and agents (p.e. recommenders). The prototype was deployed over a centralized server.

As it has been previously explained, in Triple Space Computing the tuples are expressed in form of triples. Currently, two main pure Triple Space Computing middleware implementation exist: tsc++ and TripCom.

TripCom is an European research project which was finished in 2009. It has different kernels hosted in servers which can distribute the semantic data through themselves. TripCom has a query processor to optimize queries, a security manager, a transaction manager and a web service API, but once again, is too server centered. TripCom clients are not part of the space and they could hardly be, because of the complexity of this software which is oriented to run on powerful machines (different modules of the same kernel can run in different machines).

The first Triple Space project was called TSC [3]. In TSC, triples can be interlinked to form graphs and semantic algorithms are implemented for template matching. It also offers a transactional context and a simple form to publish and subscribe to certain patterns.

tsc++ [7] is a new version of the former TSC project which basically offers the same API (without transactions) in a distributed way. To do that, tsc++ uses Jxta Peer To Peer¹ framework to perform the coordination and Sesame [1] and Owlrim [6] to store triples of each peer.

The nodes in tsc++ not only can query the space, but they can also store their own information, enabling the distribution of the space over all the peers it is made up of (this strategy is known as negative broadcast). This seems to adapt perfectly to ubiquitous system, where different devices share heterogeneous data entering and leaving the system, compromising data consistency and availability. In this aspect with tsc++ a sensor can provide information, but when it leaves the space, its information is also removed from there.

Nevertheless, tsc++ lacks some of the advantages of other alternatives: it does not make inference, it does not allow expressive querying (although it has been solved at the same time in the current version locally) and last but not least, it has not been designed for devices with reduced computing capabilities, because tsc++ middleware focused on architecture and implementation in large scale and we focused in short scale (local area networks).

As it has been seen, even if some works have analysed the convenience of the Triple Space approach in Ubiquitous computing [9,8] and others have offered solutions to this problem with tuplespaces [5], to the best of our knowledge TS has never been specifically designed and implemented to use mobile and embedded devices as another peer of these spaces and not only as simple clients. This approach allows heterogeneous devices communicate with each other limiting the necessity of fixed infrastructure and previous configuration enabling more dynamic environments.

3 Infrastructure Description

Our main goal was improving already existing Triple Space middleware to make it more suitable for IoT, enabling devices talking through different communication links and of heterogeneous nature (mobiles, embedded devices, PDAs, Tablets, even PCs) to talk to each other using standard communication protocols and also still be connected to Internet. These heterogeneous devices would communicate through a push-and-pull process using semantic and in a very decoupled fashion.

So, as one of the main concerns was allowing mobile and embedded devices to run Triple Space middleware, effort was put into developing a Java embedded adaptation of tsc++ which was still compatible with standard tsc++ namely *tscME*. It has also been developed another version for embedded devices such as Sun SPOTs or ZigBee spots which do not support Java ME specification completely. Finally, the current tsc++ implementation was also improved offering

¹ <https://jxta.dev.java.net/>

a more sophisticated API to perform more expressive queries which are spread through all devices of the space and a service API.

Thus, the API of the proposed middleware is divided in two parts. In the most basic one, primitives to manage spaces, write and query triples (in a destructive or non-destructive way), and for a subscription mechanism are provided. In the complex one, the possibility of using more expressive queries and a Triple Space based web resource oriented approach has been included. In the next sections each of these elements will be discussed.

3.1 API

The tsc++ project and TripCom project, which have their bases on Linda language, have inspired the proposed API. Our API has been structured in two levels called basic API and extended API. In the basic API the tsc++ project API has been maintained for compatibility reasons and it can be implemented both by normal devices and by mobiles. In the extended API some advanced primitives have been provided to allow service management and more expressive queries by using basic primitives. Unfortunately, not all the devices will have the capacity to implement those primitives, and consequently, they are optional.

In the basic API the following primitives can be found:

- `URI write(URI space, Set<ITriple> triples)`

When the write primitive is called, all the triples passed as parameter are stored together in the same graph associated to the specified space, which is identified by the returning URI (which would always be the same for the same set of triples). These triples are stored locally because *negative broadcast* is used.

- `Set<ITriple> read(URI space, URI graph)`
`Set<ITriple> read(URI space, ITemplate template)`

The read primitive returns a complete graph for a given graph URI or for a template which matches at least one of the triples within this graph. Reader must notice that only one graph is returned with read primitive, even if more than one graph in the space matches the graph. The remote semantic repositories will be checked before querying the local one.

- `Set<ITriple> take(URI space, URI graph)`
`Set<ITriple> take(URI space, ITemplate template)`

The take primitive is very similar to read. The main difference is that take reads a graph in a destructive way (removing it from the space).

- `Set<ITriple> query(URI space, ITemplate template)`

The query primitive returns all the triples which match the given template, no matter what graph they belong to.

- `URI advertise(URI spaceURI, ITemplate template)`
`void unadvertise(URI spaceURI, URI advertisement)`

A template is advertised to the peers subscribed to a certain template.

- `URI subscribe(URI spaceURI, ITemplate template,`
`INotificationListener listener)`

```
void unsubscribe(URI spaceURI, URI subscription)
```

Subscribe primitive expresses the interest in events related with the given template. When a peer advertises this template or another template with matches with it, the listener is called to notify the event.

The previously mentioned `ITemplate` class expresses a sequence of adjacent triple patterns which specify `WHERE`-clauses of SPARQL queries. In a recent `tsc++` version these templates have been improved allowing SPARQL queries, but the version currently supported only has triple patterns to ensure that all peers use the same API.

Our extended API defines the following primitives (more details about services in section [3.5](#)):

- `Set<ITriple> queryMultiple(URI spaceURI, ComplexTemplate template)`
Given a SPARQL query, `QueryMultiple` splits it into different `ITemplates` and sends it to other peers as one primitive, receiving multiple results for each template (partial results from the point of view of the initial query). Once it has all these results, it merges them and it makes the query again over them. Doing this in contrast with `tsc++` current solution, data stored in different peers needed by a SPARQL query can be retrieved potentially obtaining new results.
- `void register(URI spaceURI, IService service)`
`void unregister(URI spaceURI, IService service)`
It registers/unregisters a service in the space.
- `void invoke(URI spaceURI, IServiceInvocation invocation, IInvocationObserver observer)`
It performs an invocation of a service over an space.

3.2 Mobiles

In `tscME` it has been attempted to provide as much capabilities as possible keeping a small footprint and memory consumption. Unfortunately, the `Jxme` library (Java ME version of `Jxta`) is not as mature as `Jxse` (`Jxta` for Java SE) and it does not have the same communication mechanisms implemented.

Firstly, a `Jxme` peer is not able to talk to other peers using multicast, so it relies on a `Jxta` special peer called `Rendezvous` which forwards its messages to other fully capable `Jxta` peers.

Secondly, a `Jxme` peer is not completely able to exchange all kind of advertisements, the base communication unit in `tsc++`, with other peers, so we had to use pipe based communication. In pipe based communication virtual channels between peers are established and therefore they are not as flexible as advertisement based communication since it can not be configured how they work.

Because of that, `tsc++` had to be altered internally so that it communicate seamlessly with both the `tsc++` peers and the `tscME` peers.

Finally, since there are not Java ME semantic repositories right now, MicroJena [2] is used to express semantic triples and the `RecordStore` API to store them into disk whenever it is necessary (a memory *store* has been also performed because `RecordStore` caused a huge latency). Semantic reasoning has not been implemented yet in mobile peers since, to the best of our knowledge, it does not exist a low resource consuming semantic mobile reasoner publicly available.

3.3 Spots

As has been suggested, the initial goal was not making a proxy to allow the communication between really simple devices and more sophisticated ones, but both with Sun SPOTs and with XBee Gateway difficulties were found to implement a Jxta peer. To overcome this limitation, we focused on making spots communicate through some Proxies which were also Jxta peers.

The Sun SPOT (Sun Small Programmable Object Technology) [2] is a wireless sensor network mote, which can be developed in a limited version of Java ME using the Squawk Virtual Machine. Sun SPOTs do not support the IP protocol stack yet and therefore a `tsc++ gateway` to which the Sun SPOT base station, a special mote, is connected to has been used. In this gateway a really simple standard REST service server, which is also a normal peer of `tsc++` network, has been developed using Jetty server and Jersey framework.

XBee Gateway [3] is a special device which can communicate with XBee motes [4] and can be developed with Python programming language. Given that there is not a Jxta protocol library for Python, this gateway was made to communicate via sockets with a server which is a normal `tsc++` peer.

3.4 Normal Nodes

To improve the `tsc++` API, three issues have been taken into account: allowing semantic inference in each node, providing a primitive to make complex SPARQL queries over the space and offering a service API. The service API has been already outlined in section 3.1 and will be discussed further on the next section. To allow local inference in peers, the inference mechanism provided by Sesame (RDF and RDFS inference) and Owlim (OWL Horst inference) have been used.

To implement query decomposition, needed by *queryMultiple*, the SPARQL processor Jena ARQ has been used (Sesame's SPARQLParser would have been an option too) in order to split up each SPARQL query into subject-predicate-object templates that every peer in the space will answer.

² <http://www.sunspotworld.com/>

³ <http://www.digi.com/products/serialservers/connectport-ts-w.jsp>

⁴ <http://www.digi.com/products/wirelessdropinnetworking/sensors/xbee-sensors.jsp>

Input query

```

CONSTRUCT {
  ?measure ismed:hasValue ?value .
}
WHERE {
  ?measure rdf:type ismed:LightMeasure .
  ?measure ismed:hasValue ?value .
  ?measure ismed:hasDateTime ?datetime .
  OPTIONAL {
    ?measure2 rdf:type ismed:LightMeasure .
    ?measure2 ismed:hasDateTime ?datetime2 .
    FILTER(?datetime2 > ?datetime) .
  }
  FILTER( !bound(?datetime2) )
}

```

Result Templates after processing the query

```

?s rdf:type ismed:LightMeasure .
?s ismed:hasValue ?o .
?s ismed:hasDateTime ?o .

```

3.5 Services

Although *tsc++* has not got any service invocation or registering method, the necessity of providing this service infrastructure in Triple Space or not could be argued since the knowledge can be directly obtained from the space or written into it, working with resources in a very RESTful way.

More specifically in pervasive environments, the sensed data can be obtained querying the space, but some limitations when modifying actuators were discovered:

- Security. Since *tsc++* does not implement any kind of access control list, somebody might modify more knowledge than he or she wanted by mistake.
- Concurrency. If two different peers modify the same information at the same time, what information should be taken into account?
- Location of the information. Due to the nature of *tsc++*, when any information which initially belongs to *peer a* is modified by *peer b*, it is stored in *peer b* instead of being stored in *peer a*. If *peer b* leaves the space, some crucial information about the actuator will disappear. It seems logical that the information about a sensor or an actuator should be stored in the device which manages it (in the example, *peer a*).

Our solution aims to provide this control to the device which has the actuators being respectful with the asynchronous nature of TS. For that purpose, a really simple Service invocation approach which is very independent of the way the

semantic services are defined (we use our own service definition language for the scenario, but another standard languages can be used) has been designed. First, the service provider should register its service in the space (see figure 1a). The consumer would discover it querying the space, and then it would create an *invocation* using the master-worker pattern and advertise it (see figure 1b). An invocation is basically composed by an URI identifier and the input data the service may need.

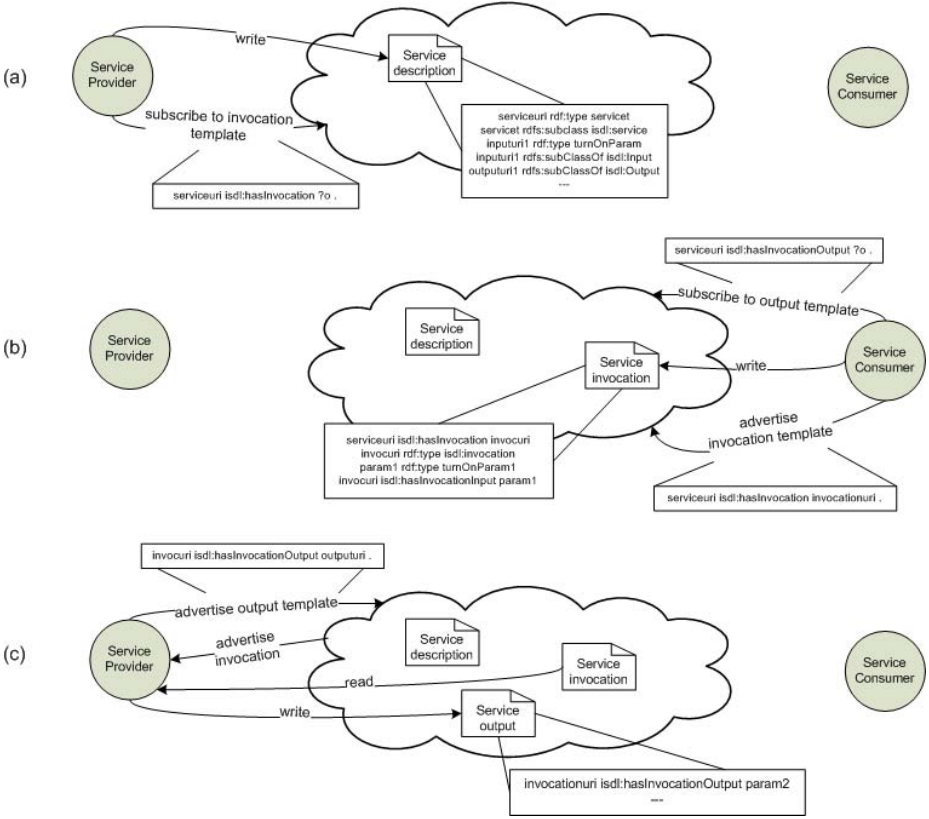


Fig. 1. Services over tsc++: a) registration, b) invocation from the consumer point of view and c) invocation from the service provider point of view

The service provider, which is subscribed to its services invocation templates, will notice the event (see figure 1c) and will retrieve the input data and perform the service (typically, performing a change in the environment using an actuator). When the invocation has been completed, the provider may write some output triples into the space and advertise the consumer in a similar fashion to the invocation.

4 Experimental Environment

There are many home automation or urban instrumentation scenarios where the proposed middleware could be used. One of this stereotypical cases could be the control of the room temperature. In this scenario, which uses all the primitives described previously, there are at least 5 peers, which are shown in the figure 2.

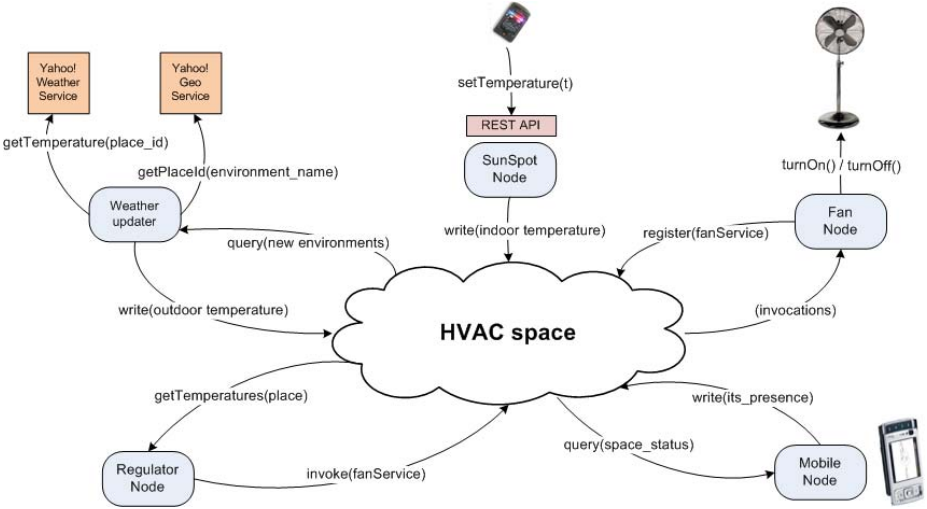


Fig. 2. Schematic view of implemented scenario

There are two different context providers: Sun SPOTS and weather provider. The first ones are physical devices which set their sensed data via the REST API. The tsc++ peer writes then this information in semantic way into the space. The weather provider, on the other hand, is a virtual context provider which gets information from the Internet and writes it as semantic information into the space. To do that, the peer polls for new environments in the space, and checks if Yahoo! has a place id for this environment name. If so, it queries the Yahoo! Weather service and it gets the current temperature in this location. The space has also an actuator device: a fan which ideally will decrease the temperature. It can be turned on or off to try to regulate indoor temperature. It can be also monitored what is happening in the space with the mobile phone.

The regulator node uses all this information and whenever someone is in a place (inferred when there is a device who belongs to somebody in the space), takes into account that the Spanish Government regulates the temperature inside public buildings to be below 26 °C whenever the air conditioning is turned on and another law regulates that in office the temperature must be between 17 °C and 27 °C Our regulator controls the temperature in a place taking into account not only the indoor but also the outdoor temperature. If the outdoor temperature

is lower than the indoor temperature, it does not make sense to turn on the fan when windows can be opened. Otherwise, the fan is turned on until the temperature reaches a lawful temperature.

So to regulate the temperature, this peer checks whether there is an actuator in the location it wants to regulate which provides a service related with the temperature. If so, a service invocation is carried out in case of needing to decrease the indoor temperature.

Basic indoor temperature control algorithm

```
while( indoort>26 ) // unbearable temperature
  if( outdoort<26 ) // user should open windows
    else // turn on the fan
```

This scenario is only a proof of concept. Obviously the fan is not the kind of device which can cool a room efficiently. Also, an easy improvement can be done introducing heating actuators and services in the space. Technically, a SWRL (Semantic Web Rule Language) could be introduced to define the temperature control rules in a more expressive and decoupled way.

5 Experimentation

Since there are no equivalent implementations apart from `tsc++` itself, we tried to enable the comparison between our middleware and the `tsc++` assessment which can be found in [7] to evaluate `tscME`.

On one hand, to assess the performance of `tscME`, we launched 5, 10 and 20 mobile peers in several emulators running on different machines of the same local network, joined to 1, 5 and 10 spaces. Each emulator, which had 64MB heap space, kept 50 graphs with 5 triples in each graph distributed homogeneously over all its spaces. Measuring the time needed to query with different primitives (see table II) it was appreciated that most of the time was spent parsing triples instead of waiting for answers. When about 10 responses were processed the time measure was quite poor, moreover, when the test was performed on a real phone the time needed increased about 2-3 seconds.

On the other hand, it seemed interesting to check how much of this time was spent in querying the knowledge base itself. As mentioned before, two different implementations have been developed: a persistent one and a non persistent one.

Table 1. TscME networking evaluation results (in seconds)

Kernels	1			10			20		
	1	5	10	1	5	10	1	5	10
read	0.23	0.22	0.26	3.48	2.99	3.04	10.01	9.97	9.8
take	0.2	0.21	0.28	3.40	2.89	2.61	10.28	9.93	11.07
query	0.4	0.27	0.24	7.05	3.66	3.34	24.83	11.9	10.56

Table 2. TscME data access evaluation results (in seconds)

Number of graphs		10	50	100
RecordStore	write	0.006	0.006	0.006
	read	0.127	0.486	1.381
	take	0.125	0.473	1.401
	query	0.226	0.934	3.001
Memory	write	0.004	0.004	0.004
	read	0.003	0.004	0.008
	take	0.007	0.010	0.017
	query	0.002	0.003	0.005

Each operation was measured over both implementations with 10, 50 and 100 graphs in each space and 10 triples in each graph (see the table 2).

A first observation demonstrates that the performance obtained with few graphs in each space is good enough when triples are kept in the RecordStore. The memory implementation shows a good performance in all the cases, specially with the query which is very fast even for 100 or more triples because of all the triples are stored not only in the graphs they belong to, but also in a common graph which is used as an optimization only for the query primitive.

Finally, we have tested the time needed for the presented scenario by using four devices: two computers (one of them containing three peers which, basically, introduces data in the space, and another one with the regulator peer), a mobile phone (Nokia N95) and a Sun SPOT. The time needed for each action in the scenario can be seen in the table 3. All the peers in this test have been configured to wait up to second for responses, bringing us to the conclusion that the scenario is performed quick enough to be applied in this case.

Table 3. Time measures for proposed scenario (in seconds)

Action	Time needed
Publish Sun SPOT indoor temperature in the space through REST API	4.69
Discover new locations in the space	5.74
Update weather measures for an unknown location	3.67
Update weather measures for a known location	2.03
Check changes on indoor and outdoor temperature	10.5
Fan activation since temperature manager invokes the its service	1.45

6 Conclusions and Future Work

This paper explores the possibility of bringing tuplespace based distributed computing to ubiquitous systems, where a lot of heterogeneous devices would share information in semantic notation asynchronously. This fits perfectly with the idea of the Internet of Things.

The results obtained in our stereotypical scenario have proven that the middleware has a reasonable performance. However in a more complex one (e.g. with more mobile peers) some scalability issues might appear depending on the capabilities of each mobile and the implementation of the scenario itself.

In addition to implementation problems, some of the used devices had not enough capacity to be part of the P2P semantic network that we have used, and they are not capable of reasoning, limiting the usefulness of the proposed middleware and resulting applications.

For our future work, we are planning to increase the expressiveness of query templates to provide comparison between strings and numbers and to implement advertisement based communication on Jxme. Finally, an overall code optimization, the attachment of a semantic reasoner and a performance analysis in a heavily instrumented deployment scenario should be taken into account.

References

1. Broekstra, J., Kampman, A., Harmelen, F.V.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
2. Crivellaro, F., Genovese, G.: Jena: Gestione di ontologie sui dispositivi mobili (2007)
3. Fensel, D.: Triple-space computing: Semantic web services based on persistent publication of information. In: Aagesen, F.A., Anutariya, C., Wuwongse, V. (eds.) INTELCCOMM 2004. LNCS, vol. 3283, pp. 43–53. Springer, Heidelberg (2004)
4. Gelernter, D.: Generative communication in linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7(1), 80–112 (1985)
5. Khushraj, D., Lassila, O., Finin, T.: sTuples: semantic tuple spaces (2004)
6. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM a pragmatic semantic repository for OWL. In: *Web Information Systems Engineering Workshops*, pp. 182–192 (2005)
7. Krummenacher, R., Blunder, D., Simperl, E., Fried, M.: An open distributed middleware for the semantic web. In: *International Conference on Semantic Systems, I-SEMANTICS* (2009)
8. Krummenacher, R., Kopeck, J., Strang, T.: Sharing context information in semantic spaces. In: *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, pp. 229–232 (2005)
9. Krummenacher, R., Strang, T.: Ubiquitous semantic spaces. In: *Conference Supplement to the 7th International Conference on Ubiquitous Computing* (2005)
10. Martin-Recuerda, F.: Towards CSpaces: a new perspective for the semantic web. In: *Industrial Applications of Semantic Web*, pp. 113–139 (2005)
11. Nixon, L.J., Simperl, E., Krummenacher, R., Martin-Recuerda, F.: Tuple-space-based computing for the semantic web: a survey of the state-of-the-art. *The Knowledge Engineering Review* 23(02), 181–212 (2008)
12. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp* (2004)
13. Tolksdorf, R., Bontas, E.P., Nixon, L.J.: A coordination model for the semantic web. In: *Proceedings of the 2006 ACM Symposium on Applied Computing*, p. 423 (2006)
14. Winograd, T.: Architectures for context. *Human-Computer Interaction* 16(2), 401–419 (2001)

Touch-Based Services' Catalogs for AAL

Jose Bravo, Ramon Hervás, and Jesus Fontecha

MAMi Reseach Lab – Paseo de la Universidad, 4 - 13071 – C. Real, Spain
{jose.bravo, ramon.hervas, jesus.fontecha}@uclm.es

Abstract. Elderly people living alone at home need support for daily activities. For this reason, the AAL initiative promotes technological adaptabilities but bearing in mind that it is crucial serving users in terms of usability. In this paper, we present a position paper for supporting elderly needs by simple touches of services' catalogs. Further, we propose a generalization of these services based on four examples: shopping list, catering, calls and mobile prescriptions.

1 Introduction

Nowadays population is aging alarming rate worldwide. A consequence, some problems like independence, safety, difficult to relate with technologies to support their daily routine due to cognitive declining, are problems that governments try to solve with many initiatives because the progressive aging of people has economical consequences, not only for future decades, but already these days.

AAL promotes the use of technologies for helping elderly people in order to maintain their autonomy, increasing their quality of life and facilitating their daily activities for augmenting the time living at their homes. For this reason we have considered the mobility as one of the most important problem for AAL [1]. In this line, the European Commission encourages technology and innovation for AAL. This fact makes that, research community in this area, believes important of adapting technologies for supporting elderly people living alone at home. However, this kind of technologies must be closer to this kind of users, minimizing the interaction, even trying to make disappearing it. The reason, as we mentioned before, is old people are not familiarized with technologies. This implies it would be desirable an interaction as simple as possible.

There are works that use touch-based technologies for supporting elderly activities. In [2], authors propose a NFC-based system for supporting elderly people to choose their meals to be delivered by means of a home care service. In [3] a proposal for NFC-based assisted living with a shopping list assistant is presented. Our experience for supporting elderly by adapting NFC technology is varied. In [4] a proposal for supporting Alzheimer caregiver can be seen. Finally, how to enable mobile prescription can be studied in more detail [5].

Bearing in mind all above described, we propose a solution to improve the care-dependent people's autonomy by using NFC-enabled mobile phones with tags services' catalogs. Moreover, since the user interacts with catalogs in an easy and natural way, people's reticence of using new technologies because most care-dependent people are not familiarized with them, will not be a problem.

Under these lines, we present four solutions for elderly: shopping list, catering, calls and mobile prescriptions. Next section we propose a generalization through tag-services catalogs with the correspondent taxonomy, architecture and middleware. Finally, conclusions are drawn about this position paper.

2 Touch-Based Applications Examples for Elderly

In our previous works for supporting daily activities for elderly people, we began with simple applications. In this sense, we have developed four proposals: the shopping list, the catering service, relatives and friend calls and the mobile prescription service. In the first one, we try to solve people barriers going to the supermarket. They need assistants for carrying out their daily needs of food or articles for cleaning home. In this sense, we have prepared carefully a products' list with a number of tags covered by the correspondent icons as it can be seen on the left of figure 1. By touching icons representing needed products, the assistant receives into the mobile phone the necessities in a transparent mode and carrying products to elderly home. The second application corresponds to the catering services (see the second picture in figure 1). Some dishes are disposed in starters, main and desert courses. It works similar to the shopping list. In the third application, calling relatives and friends is possible through NFC technology by only touching photos-albums (see the third picture in figure 1). Finally, the mobile prescription (see last picture of figure 4) is a more specialized solution although the same process by touching tagged medicines' boxes has to be handled. This fact solves the problem to obtain prescriptions where patients have to phone the healthcare center for arranging an appointment with the doctor. After that, they have to go there, wait for their turns to come and go back home with the prescription papers. All these activities make care-dependent people's life even more difficult, especially for those who have mobility problems. Mobile Prescription claims to improve care-dependent people's quality of life. To that end, patients will be able to obtain the prescriptions of their medicines from home, without having to go to the health-care center.

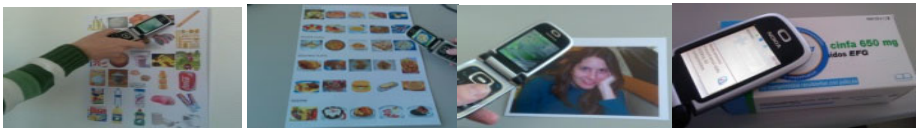


Fig. 1. Shopping List, Catering Service and Calls

3 Tag-Services Catalogs

After these experiences, we propose a solution for generalizing the idea and, at thus way more services can be managed by using this technology through touching interaction. In the next point we present our proposal for managing services catalogs involving providers and customers.

3.1 Architecture

Our architecture can be seen in figure 2. We propose a cloud services for customers. In it, providers can connect to obtain automatically the mode of producing their catalogs. In this sense, it is necessary a usable process to define products management and the communications process. Initially, we propose the Quality-of-Service Network model (VQN) in order to build an efficient and highly functional communication platform [8]. VQN is a semantic overlay network implemented as a distributed application by means of an object oriented middleware for distributed systems (i.e. ZeroC ice). For the communications process, our idea is a middleware whose structure is represented in figure 2.

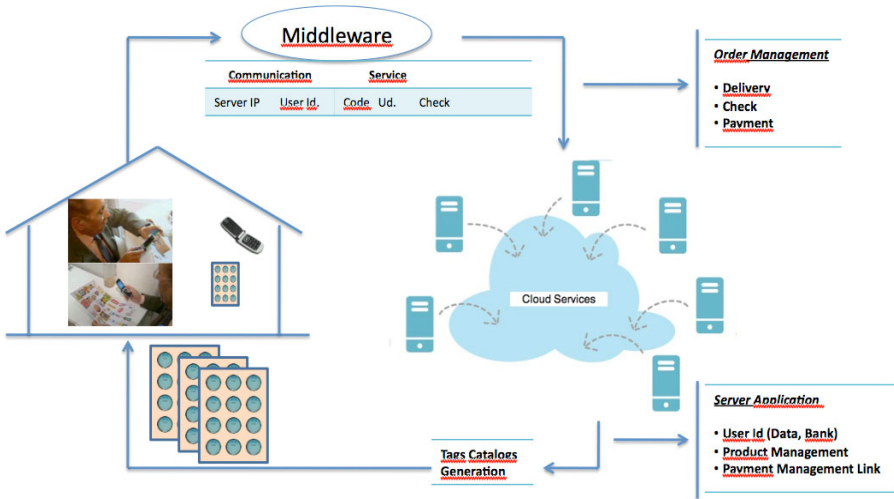


Fig. 2. Architecture and Middleware

3.2 Services' Taxonomy

In order to study services' needs, it is important to explore all the requirements of the information transmissions. Table 1 shows an example of the four mentioned applications. In it, we can see catering service (CS), shopping list (SL), phone calls (PC) & mobile prescriptions (MP) examples. Additionally, products descriptions with the identification and check control can be observed. In the case of shopping list or catering service, the check control is produced when receiving products at home through assistant catalogs touches. In the case of pharmacy, it is received by GPRS from the doctor server as a combination of user and product identification. This check will be necessary for obtaining drugs at the pharmacy.

With these needs we propose a middleware for mobile phone that includes this structure (communication and services identification), this fact will make possible that elderly do not have to change their mobile application when needing a new services' catalog. For that, providers, with our definition process in the cloud services, can adapt automatically their own products' management.

Table 1. Data Communication Structure

Communication		Service		
Server IP / Phone Number	User Id. / Phone Number	Product Id. / Relatives or Friends	Ud.	Check
167.165.76.32	1221450012	CS-Chicken-17001	1	Received? Ok
167.165.76.32	1221450012	CS-Salad-17005	1	Received? Ok
167.165.76.32	1221450012	CS-Banana-17011	1	Received? Ok
192.181.92.11	1221450012	SL-Milk-29002	1	Received? Ok
192.181.92.11	1221450012	SL-Bread-29087	1	Received? NO
189.178.11.43	1221450012	PH-Nolotil-34090	1	340901221450012
189.178.11.43	1221450012	MP-Espidifen-56897	1	568971221450012
189.178.11.43	1221450012	MP-Gelocatil-87576	1	875761221450012
677 432 657	675 666 980	Daughter Ann	-	-

4 Conclusions

We have studied how to address the elderly people's technological barriers for living alone by using NFC technology. First of all we have implemented some applications reducing the interaction by only touching tags. Then, a generalization is proposed through the combination of middleware, communication protocol and server applications in a cloud services. After all, we are seeking to find a manner to generalize touch-based applications on mobile phones, which have not the server-side deployment issues by leveraging from the Cloud Computing paradigm.

References

1. The Ambient Assisted Living (AAL) Joint Programme, <http://www.aal-europe.eu/>
2. Häikiö, J., et al.: Touch-Based User Interface for Elderly Users. In: *Mobile HCI 2007*, Singapore (2007)
3. Llorente, M.A., et al.: Analysis, development and evaluation of AAL applications based on NFC technology to support independent daily life. In: Tscheligi, M., et al. (eds.) *AmI 2009*, Salzburg, Austria (2009)
4. Bravo, J., López-de-Ipiña, D., Fuentes, C., Hervás, R., Peña, R., Vergara, M., Casero, G.: Enabling NFC Technology for Supporting Chronic Diseases: A Proposal for Alzheimer Caregivers. In: Aarts, E., Crowley, J.L., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.) *AmI 2008*. LNCS, vol. 5355, pp. 109–125. Springer, Heidelberg (2008)
5. Vergara, M., Díaz-Hellin, P., Fontecha, J., Hervás, R., Sánchez-Barba, C., Fuentes, C., Bravo, J.: Mobile Prescription: An NFC-Based Proposal for AAL. In: *2nd International Workshop on Near Field Communication*, Monaco (2010)
6. Urzaiz, G., Villa, D., Villanueva, F.J., Moya, F., Rincón, F., López, J.C., Muñoz, L.A.: A novel communication platform to enable the collaboration of Autonomous Underwater Vehicles. In: *International Conference on Wireless Networks (ICWN 2009)*, Las Vegas, Nevada (2009)

Designing Context-Aware Interactions for Task-Based Applications*

Pablo Muñoz, Pau Giner, and Miriam Gil

Centro de Investigación en Métodos de Producción de Software
Universidad Politécnica de Valencia, 46022 Valencia, España
{pmunoz, pginer, mgil}@pros.upv.es

Abstract. Since contextual information has potential to improve task-based applications, we provide an approach for integrating contextual information in task-based applications by considering simplicity as a major design goal. We present a context-aware application to support mobile workflows focusing our solutions in three important factors to organize and manage tasks: priority, location and time. Following the simplicity guidelines, we provide solutions based on these factors by means of visualizations that allow users to complete their tasks fluently on the go.

Keywords: context-aware, interaction, location-based services, mobile workflow applications, mobile devices.

1 Introduction

The increasing improvement in the communication ability of mobile devices allows users to access information and web services anywhere and anytime. Furthermore, these devices have become full of sensors that can access contextual information such as the current location of the user by means of GPS coordinates. By improving the access to this contextual information, we increase the richness of communication in human-computer interaction and make it possible to produce more useful computational services [10]. One of these services could allow making daily activities more fluent through reducing the need for data input.

In particular, this work is focused on task-based applications. According to the classification of web applications defined by Unger in [8], task-based applications are defined as “*a tool or collection of tools meant to allow users to accomplish a set of key tasks or workflows*”. As we have experienced in the development of Presto [1] contextual information has potential to improve task-based applications. This work extends the architecture components defined in Presto with interaction mechanisms to allow users to interact with the system taking into account different kinds of context information.

Interaction aspects are a crucial factor for determining the application’s success or failure since a bad interaction can cause that user does not use the application.

* This work has been developed with the support of MICINN under the project SESAMO TIN2007-62894 and co-financed with ERDF.

Specially, we have put emphasis in simplicity as one of the main goals of our design solutions. Some products such as iPod, which was designed following a simplicity approach, has found success with a simple and non-intrusive design that does not invade the user's mind [5].

In this work we provide an approach for integrating contextual information in task-based applications by considering simplicity as a major design goal. In order to illustrate our approach we present a context-aware application to support mobile workflows. This application is in a prototypical stage but allows users to manage a to-do list in a different way according to context conditions. The user can discover what to do next by organizing their tasks according to different context conditions such as location, time or priority. Context information has been integrated following the simplicity principles allowing users to complete their tasks fluently.

The remainder of this paper is structured as follows. Section 2 relates the interaction design process applied in our approach. Section 3 introduces the application and its proposals for adapting it to the user context. Section 4 introduces related work in the area. Finally, Section 5 concludes the paper.

2 The Interaction Design Process

For the design of context-aware applications we prioritized simplicity since we wanted to focus on the task at hand without any user distractions. We avoid complexity as much as possible because it involves a high mental effort and forces the user to divert his attention from other tasks.

The problems presented by complexity are usually accentuated due to the constraints introduced by mobile devices in terms of interaction (e.g., small screens, reduced processing power, etc.). For this reason, the idea of achieving a simple interaction design is essential to get and keep success in an application or product. However, designers find hard to create simple interaction designs that cover all requirements because there is no process or method to support the design of simple interfaces.

Since we wanted to obtain simple designs in an early step we inspired in the “*Laws of Simplicity*” of Maeda [5] in order to design our interactions (see table 1). These guidelines are metaphors which are applicable to different areas such as design, business, technology and life. In our work we took some of these guidelines to detect and fix complexity problems. Specifically, in our work we apply the laws:

- **Reduce.** To preserve the most relevant information and operations in the main screen, diverting the rest to other contexts where user can navigate.
- **Organize.** To group similar components and operations.
- **Time.** To avoid unnecessary navigation.
- **Learn.** To apply popular and intuitive knowledge that is movable to an application
- **Context.** To emphasize the more relevant factors according to the user needs.

According to these laws, we refined our interaction designs in different iterations in which new ideas are emerged such as contextualize the zoom controls. At the end, we obtained interaction designs more focused in the end-user.

Table 1. Maeda's Laws of Simplicity

Law	Description
1 Reduce	The simplest way to achieve simplicity is through thoughtful reduction.
2 Organize	Organization makes a system of many appear fewer.
3 Time	Savings in time feel like simplicity.
4 Learn	Knowledge makes everything simpler.
5 Differences	Simplicity and complexity need each other.
6 Context	What lies in the periphery of simplicity is definitely not peripheral.
7 Emotion	More emotions are better than less.
8 Trust	In simplicity we trust.
9 Failure	Some things can never be made simple.
10 The one	Simplicity is about subtracting the obvious, and adding the meaningful.

3 The Application

In a mobile environment where this work is situated, the appropriate information provided by the device is determined by the changes caused in the context of use (e.g., changes in location, in time, in activity, and more). Some of these changes, specially “*location*” and “*time*”, were appointed as important types of context in [10].

Therefore, in the mentioned interaction design process, we considered the user's needs in different contexts by analyzing the typical patterns that users follow for completing their tasks. For this, in a brainstorming the authors analyzed the different features, some of them included in several to-do applications such as GeoLife, Remember the Milk and Astrid. Thus, we defined some possible requirements for the application. These possible requirements were: “*the necessity of view the current tasks, identify tasks by priority criteria, know how arrive to the place where the task must be performed, view a general state of the agenda in just a screen, generate an optimized route, delegate tasks, establish order between task and support collaborative tasks, etc.*”. After the brainstorming session we determined that priority, location and time were the main factors for task organization and we limited application's functionality through selecting some of these requirements that were related to these factors.

Faced with these established necessities, we proposed solutions by means of an Android¹-based prototype that provides three different visualizations. Each of these visualizations is focused on addressing the requirements detected for managing tasks in a particular context. Since users can move from view to view, they allow an adaptation to the user's current context of use. The three proposed visualizations are:

- **Map View.** It provides a view where *the tasks are geo-located in a map*, allowing the user to know in which locations has more pending tasks. In this way, user is urged to plan his optimized routes. This view shows also the user's current location, and it is centered on this context information.
- **Timeline View.** It provides *a guided view over the progress of the tasks in time*. Its objective is informing the user of the next tasks that must be performed with

¹ <http://www.android.com/>

its associated cost. Similar to the *Map View*, this view is focused on time factor, since it is the most important attribute to take into account. So, we center the *Timeline View* in the next task that must be performed. In the same way that *Map View* centers his view on the most view’s important attribute, location, *Timeline View* centers its view in its most important attribute, time, so we center its view in the next task that must be performed.

- **Cloud View.** It responds to the need to *prioritize tasks*. With the *Cloud View*, users get a high-general view of a set of tasks organized by priorities. In this way, they obtain a general view of its agenda in just a screen. The visualization consists in showing the tasks like a cloud of tags, where each tag is the name of a task and where each tag has a particular font according to its priority.

3.1 Common Matters

Although the proposed visualizations are different from each other, there are some common matters between them that we relate below:

General Interaction Concerns. Regarding to the general interaction we maintain two common interaction controls in all the visualizations. Thus, we applied the forth law of simplicity “*learn*” because the common use of the same controls in all of the visualizations reduces the mental effort for learning how to use the application. These interaction controls are filters that allow focusing the user’s view in a smaller set of tasks (this idea applies the first law “*reduce*” in order to hide non-relevant data through the removal of far future or low-priority tasks).More specially, these filters are a time filter and a priority filter (see figure 1). The time filter allows showing tasks situated in a different ranges of time, which they are: “*today, next three days, next week, next month, all time, and tasks that are not programmed for a concrete time (e.g. clean the basement)*”. Besides, the priority filter allows showing tasks with specific priority types, which these priority types are: “*high, medium and low*”. Thus, the possible total combinations between priority and time values cover much of the user’s needs for filtering tasks according to time and priority factors. For example, one of these possible combinations should be “*gettingonlyhi priority tasks for the next week*”.



Fig. 1. Filter controls

Mechanism for switching visualization and adaptation of the visualization. According to the changes in user’s context and applying the sixth law of simplicity “*context*”, we define mechanisms that allow a change between visualizations in order to cover the current context needs. Furthermore, when the visualization is changed, another factor like the filters’ state of the selection of a task should not be changed. So, we decide preserve these factors over context changes and focus the new visualization on the selected task and preserving the filters’ state. Thus, we do not change the current user environment unexpectedly.

Moreover, the current time and location are context information that is relevant for change dynamically the view. An automated change in view is only enabled when user is not interacting with the application. This generates good emotions in user (we apply here the seventh law of simplicity: “*emotion*”) according to an underlying received data (such as higher proximity to a location where user has a pending task). We take advantage of this by means of the interpretation of the changes in location and time, two of the three main factors for task organization. So, when user is using any visualization and changes its location by other nearby where it has pending tasks, the application changes to the *Map View* automatically and sends a notification to the user. Besides, when the next task must soon be performed, the application changes automatically to the *Timeline View* and sends a notification to the user.

Task’s interaction. Although one of the main goals of a task-based application is to provide users with a well-organized list of tasks, mechanisms are also required for handling this information. Thus, in our application we considered the operations “*complete, delete, edit and send to contact*” for manipulate tasks on the go.

For the purpose of organize the position of these operations we used two laws of simplicity: “*reduce*”, “*organize*” and “*time*”. We applied “*reduce*” to show only the most important operation “*complete*” in the overlay showed when user wants to interact with a task. The rest of the operations were grouped (using the second law of simplicity “*organize*”) in the detailed view of the task, which is reachable by means of the hyperlink showed in the overlay. These operations are the “*delete, edit*” operations, and an operation that allows user to delegate its tasks to its contacts called “*send task to contact*”. About this operation, note that if user is prevented to perform a task because of a change in user’s context, “*send task to contact*” provides a mechanisms for collaboration by sending tasks to others.

Furthermore, as user can navigates to a more detailed view of a selected task, the other operations are accessible by only once navigation (applying in this way the third law “*time*” that avoid to waste time).

Task’s data. Since we follow an approach based on simplicity, we wanted different sources of information to be easy to integrate. Tasks are represented using JSON² (JavaScript Object Notation). JSON is a widely used format in web services which are guided by the REST approximation. The use of JSON facilitated the development since it is easy to manipulate using different technologies which allows a seamless-transition during development (e.g., from using a static text file to a task list dynamically generated by a web service).

At the prototyping stage the task dataset has been synthetically generated. It is composed by 34 tasks that provide variability in our prototype. Nevertheless, thanks to the use of JSON the integration with other task sources (e.g., services such as Remember the Milk, Google Calendar, etc.) is straightforward.

After organize requirements we determined task’s attributes. These attributes represent information which is related to identification, location, transport, time, priority or completeness state. Note that each visualization shows and uses a subset of these attributes to achieve his goals.

After viewing general matters that concern all visualizations, next we detail how each visualization adapts it to the context in a non-intrusive way.

² <http://www.json.org/>

3.2 Map View

Main objective. The goal of this visualization is to inform the user about the tasks that can be completed in a given location. This allows the user to plan routes, organize his mind and avoid meaningless displacements. To achieve this goal, the visualization shows a map where the user's tasks that have an assigned location are represented as markers in a geo-located way.

Adapting the visualization. For adapting the visualization according to location is important to represent the current user's location (e.g., by means of a person icon) in the screen because this inform the user where is respecting his pending tasks. Furthermore, as the underlying mobile workflow is centered in user, we focus the view on this person icon (applying the sixth law of simplicity "*context*").

Interacting with the Overlays. The different tasks are represented in the map by means of different markers. In order to complete tasks on the go, user selects a marker to interact with tasks geo-located in marker's position. As different tasks should be performed in the same location, markers adapt it to this situation. So, on the one hand, if exist only one task in a location, the marker only shows the representative icon of the task and has a background color according to the task's priority. On the other hand, if exists two or more task in the same location, the marker shows the number of tasks assigned to this location with a background color that depends on this number.

Regarding the interaction with a task, when a marker is selected appears an overlay with a list of task elements (see figure 2). In this overlay, we used the first law "*reduce*" for represent the tasks only by his name and his representative icon. We applied also the forth law "*learn*", taking advantage of the popular knowledge of hyperlinks. This knowledge associates an underlying text with a hyperlink. Specially, we represented the name of the task as a link to the more detailed view of the selected task. Finally, in order to "*complete*" a task is added a checkbox.

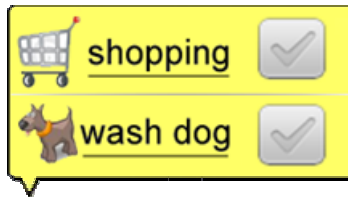


Fig. 2. Overlay example

Adding contextual zoom in and zoom out controls. The zoom controls of an application provide a general or a concrete representation of the current view. Underlying this, we think that these controls are no longer a magnifying glass. We think the zoom controls should adapt the visualization (this idea uses the sixth law "*context*" defining the most relevant information in the different situations, which in this case are the different values of the zoom level) . So for example, user can visualize a view of its city with its associated tasks as we see in the second image of figure 3. Next, user can use zoom out control for obtain a view of its country with his associated tasks. At this

moment, the overlays of the tasks that are located in all neighborhoods of the city converge in only an overlay that represent the number of tasks located in the city as we can see in the first image of figure 3. On the contrary, if user is in a view of his country and focus the view on the user's city, the general overlay diverges on different overlays which are located with more accuracy. This cleans the screen of a set of markers showing the same information in a more general view.

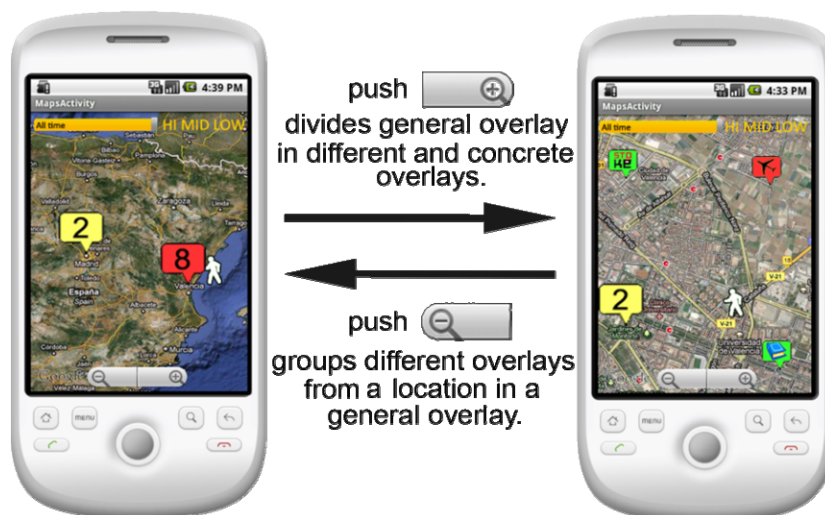


Fig. 3. View adaptation according the zoom level

3.3 Timeline View

Main Objective. The goal of this visualization is to provide a temporal order between the selected tasks with assigned time for its performance. This ordered vision informs the user about the next tasks that must be complete first, with associated and useful information such as temporal or economic cost or a link to the task's location for complete this.

Timeline Contents. In this visualization, the tasks are showed as a column set where each column represents one task. Therefore, applying the sixth law "*context*", we determine the information to be showed. So, a column shows the task's representative icon surrounded by a border that refers to its priority. Above this, its estimated duration with its associated economic cost is showed (e.g. buy a book should cost about 20\$ and should need about 10 minutes). In the same way, below the task icon, the visualization shows the mean of transport defined to arrive to the task's place where it must be performed with its associated time and economic cost (e.g. return home by bus costs 25 minutes and 1.5\$). Note that, by applying the first law "*reduce*" we hid avoid information (i.e. everybody knows that return home by foot or wash dog is for free).

Finally, at the bottom of the screen we represent the time bar (see figure 4) that indicates the time advance, showing the tasks' times when must be performed.



Fig. 4. Time bar

Adapting the visualization to the time. Since time is the most important attribute in this visualization, contextualize the view according this concern is an important requirement. Therefore, we decide first to focus the view on the current time. So, in order to inform the user which is the task that must be performed at first time we highlight the next task to be performed. Besides, for express the times of the tasks we considered the current time in order to adapt the showed information to this value. Thus, the application shows the hour when a task that must be performed is programmed for the current date. Otherwise, it shows the date where the task must be performed. This is because when a task is far from the current date, the user's mind prioritizes the date attribute against the hour attribute to organize its mind. However, for user is not relevant to visualize the date for today task since this is redundant information. The interaction design for this view is represented in figure 5.



Fig. 5. Timeline View

Interacting with a task. In this view, task's interaction is different to the *Map View*. In this case, when user selects a task anyoverlay is showed. In this view, we associate the double-tap gesture with link to the task's more detailed view. Moreover, if user selects the mean of transport icon is because user is thinking in a route or in how arrive to the location where task must be performed. For this purpose, applying the forth law of simplicity "*learn*", we link this icon to move the view to the *Map View* that is centered in this task's location.

Besides, in this view we take advantage of gestures in order to include the "*complete*" and "*delete*" operations in the main view. So, introducing gestural over the task icon, users can use an "*up movement*" when they want to complete a task and they can use a "*downmovement*" when they want to delete the task.

Specifically, we propose to associate the upward movement to change the representative image of the task by a "completed" icon, and the downward movement to change the representative image of the task by a "deleted" icon. These movements may be corrected during 5 seconds by making an opposite movement. For user, this allows a quick-useful "undo" operation for correct mistakes. However if time expired, the task is removed from the view. Note that this method of correction avoids launching out a dialog alert. For example, when user uses the negative operation "delete", we put in use the third law of simplicity "time" in order to earn user's time. In Figure 6 we see all the possible transitions occurred between these movements.



Fig. 6. Gestural interaction in Timeline View

3.4 Cloud View

Main objective. The main objective of this visualization is to offer a high-general view of selected tasks, where these tasks are clearly differentiated according to its priority level. It provides user with a general view of his agenda in just a screen. For this purpose, we applied the first law of simplicity "reduce" to abstract user of any other attribute such as location, time or costs. Only the task's name and task's priority are needed for a high-general view.

This visualization (see figure 7) consists in showing the task data set as a Tag Cloud, where each tag corresponds with each task name. In order to differentiate easily the organized task by priority, the visualization uses different fonts, each one for each priority type. Thus, high priority tasks are highlighted by means of bold red color font, medium priority task are represented by means of white bold color font, and low priority tasks are showed discretely via a white no-bold font.

Interacting with a task. Interacting with a task in this view is similar with the task's interaction in *Map View*. When a task is selected the same overlay related in the *Map*

View is presented, applying the same laws of simplicity. Furthermore, the most important operation, i.e. “complete” operation is also added in this overlay. At the same time, this selected task's font is changed for differentiate it from the rest of tags.

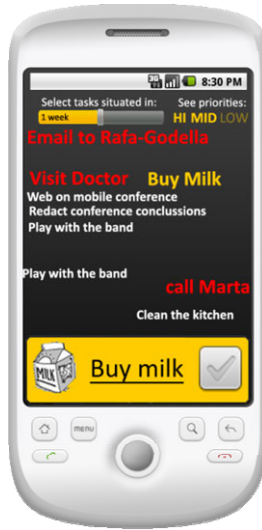


Fig. 7. Cloud View

4 Related Work

User Centered Design (UCD) approaches [9] stress the relevance of the user needs when designing interaction. One of the goals of UCD is to avoid the complexity introduced by technology and provide designs that fit better with the users and their activities. However, too much attention to the needs of the users can lead to a lack of cohesion and added complexity in the design [6].

UCD approaches include different steps such as specifying the context of use and requirements (e.g., by means of field studies or contextual observation), produce design solutions (e.g., prototyping) and evaluate designs (e.g., by means of field and laboratory studies). Simplicity can be considered as an implicit design goal in most of the UCD approaches, but it is not normally stated explicitly as an independent step in the process. In this work we considered this aspect as an explicit goal during design.

We applied the above design principles for task-based applications. A task-based application model allows users to access the services they need easily [2]. In particular, different projects exist with the goal of producing context-aware task-based applications. The Castaway project [3] stresses the relevance of location when supporting user tasks. The map view provided by Castaway has some common aspects with our map view such as the capabilities for clustering tasks. However, the Castaway does not consider the integration of different sources of contextual information. Other task-based systems such as Taskminder [4] consider time in addition to location. Taskminder represents tasks by means of a calendar and allows tasks to be set a different level of urgency and importance. Our application provides a timeline which connects task information with transport

and cost. The use of a timeline avoids the need to provide each time period a fixed-size representation.

5 Conclusions

This work has provided us with guidelines for the design of context-aware task-based applications that have been useful for the design of a workflow supporting application. The next steps are: to work on our design process to define concrete functions for our simplicity-based role and to evaluate our approach through obtaining user feedback from the provided prototype outside the lab.

The developed prototype was also valuable to detect different situations where a closer integration between physical and digital worlds is required. Considering context information not only about the user but also about other people that interact with the user could help to a better organization of the user tasks. In addition, the easy specification of compensation actions for automate the handling of certain tasks also requires further study.

Referentes

- [1] Giner, P., Cetina, C., Fons, J., Pelechano, V.: Developing mobile workflow support in the internet of things. *IEEE Pervasive Computing* 9(2), 18–26 (2010)
- [2] Goth, G.: The task-based interface: Not your father's desktop. *IEEE Software* 26(6), 88–91 (2009)
- [3] Kessell, A., Chan, C.: Castaway: a context-aware task management system. In: *CHI 2006: CHI 2006 extended abstracts on Human factors in computing systems*, pp. 941–946. ACM, New York (2006)
- [4] Landry, B.M., Nair, R., Pousman, Z., Tungare, M.: Taskminder: A context- and user-aware to-do list management system. Technical report, Georgia Institute of Technology (2003)
- [5] Maeda, J.: *The Laws of Simplicity (Simplicity: Design, Technology, Business, Life)*. The MIT Press, Cambridge (2006)
- [6] Norman, D.A.: Human-centered design considered harmful. *Interactions* 12(4), 14–19 (2005)
- [7] Radi, H., Mayrhofer, R.: Towards alternative user interfaces for capturing and managing tasks with mobile devices. In: *Proc. MoMM 2008: 6th International Conference on Advances in Mobile Computing and Multimedia*, pp. 272–275 (November 2008)
- [8] Unger, R., Chandler, C.: *A Project Guide to UX Design: For user experience designers in the field or in the making*. New Riders Publishing, Thousand Oaks (2009)
- [9] Vredenburg, K., Mao, J.-Y., Smith, P.W., Carey, T.: A survey of user-centered design practice. In: *CHI 2002: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 471–478. ACM, New York (2002)
- [10] Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. In: *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness* (2000)

Tourist Trip Planning Functionalities: State-of-the-Art and Future

Wouter Souffriau and Pieter Vansteenwegen

Katholieke Universiteit Leuven
Centre for Industrial Management
Celestijnenlaan 300A, 3001 Leuven, Belgium

Abstract. When tourists visit a city or region, they cannot visit every point of interest available, as they are constrained in time and budget. Tourist recommender applications help tourists by presenting a personal selection. Providing adequate tour scheduling support for these kinds of applications is a daunting task for the application developer. The objective of this paper is to demonstrate how existing models from the field of Operations Research (OR) fit this scheduling problem, and enable a wide range of tourist trip planning functionalities. Using the Orienteering Problem (OP) and its extensions to model the tourist trip planning problem, allows to deal with a vast number of practical planning problems.

1 Introduction

Many tourists visit a region or a city for one or more days. It is not possible to visit every tourist attraction or cultural heritage site during such a limited period, so the tourist has to make a selection of what he believes to be the most valuable Points of Interest (POI). This personal selection is based on information found on web sites, in articles in magazines or in guidebooks from specialised book stores or libraries. Once the selection is made, the tourist decides on a route, keeping in mind the opening hours of the POIs and the available time.

Tourists face several difficulties when following that procedure. Information provided in guidebooks can be out-of-date, e.g. opening hours may change. Also, guidebooks cannot provide temporal information: temporary exhibitions in museums change all the time, some POIs are (partly) closed due to renovation and theatres change their programme regularly (Dunlop et al., 2004). Tourists have to combine the information from different sources and decide which information is the most reliable. Moreover, selecting the most valuable POIs, i.e. those of the greatest interest to the tourist, is not easy. Usually tourists will be happy if they devise a somewhat attractive and feasible schedule, but they have no idea whether better schedules are possible.

Some guidebooks acknowledge these problems and propose generic visitor tours through a city or region. Of course these tours are constructed in order to satisfy the interests of the majority rather than the specific interests of individuals (Cheverst et al., 2000). Generic visitor tours do not take user context into account, e.g. the start and end location, the available time, the current time,

the weather, etc. [Kramer et al. \(2006\)](#) have analysed the diversity of gathered tourist interest profiles and conclude that they are surprisingly diverse. This conclusion supports the idea of creating personalised tours instead of proposing generic visitor tours. Furthermore, tourists today want to use their free time in an optimal way and they expect to be well informed on what a city or specific POI can offer ([Oppermann and Specht, 1999](#); [Keyson, 2004](#)).

Web-based decision support applications are excellent aids for tourists who want real support for tourist planning problems. Based on an interest profile, up-to-date POI information and trip information, a (near-) optimal and feasible selection of POIs and a route between them can be suggested ([Vansteenwegen and Oudheusden, 2007](#)). Also, most tourists today move within a limited crowded area of very attractive POIs. [Kramer et al. \(2006\)](#) state that a system enabling personal selection and routing of POIs will help to spread tourists more evenly across the destination region, which helps to prevent crowds.

Since the Second World War, the science of Operational Research (OR) has been applied to a vast range of problems in different sectors. OR is concerned with applying mathematics, statistics, optimisation technology, etc. to provide decision makers with (near-) optimal solutions to complex problems in military contexts, manufacturing, transportation, logistics, finance, etc. However, the field of tourism, in particular personalised trip planning, has been largely ignored.

[Godart \(2001\)](#) uses the Travelling Salesperson Problem (TSP) as a starting point to plan trips. His TSP with Activities and Lodging Selection (ALS) automatically selects POIs and lodging. The Multiple Objectives extension (MOTSP-ALS) minimises transport and accommodation costs at the same time. Finally, Preference Based MOTSP-ALS also maximises the attractiveness of the lodging and the activities. This complicated model turned out to be difficult to solve. This research resulted in a web based tour planner, YourTour¹.

[Vansteenwegen and Oudheusden \(2007\)](#) advocate the use of the OP and its extensions to solve Tourist Trip Design Problems (TTDP). The OP integrates automated selection of locations with finding the shortest path, and is therefore highly appropriate to model TTDPs. The objective of this paper is to demonstrate how a wide range of real-life tourist trip planning functionalities can be enabled by using the OP. First, Section 2 presents an overview of systems that compose tours of POI visits. Next, Section 3 discusses the wide range of planning-related functionalities that are offered by these systems and compares them based on the functionality they offer. Section 4 explains how the OP and its extensions can be used to model trip planning functionalities. Finally, Section 5 concludes this paper.

2 State-of-the-Art

Instead of recommending pre-packaged tours, or sorting POIs by estimated interest value as recommender systems do, scheduling approaches typically try to determine the combination of POIs that maximise the joint interest.

¹ <http://www.yourtour.com>

[Soo and Liang \(2001\)](#) present a software agent that recommends a trip plan through dialogue with the user. Custom trips to the city of Taipei, China, are proposed by first letting the tourist select his hotel(s) and next automatically filling the available time with POI visits in an nearest-neighbour fashion.

[Ardissono et al. \(2002, 2003\)](#) describe their INteractive TouRist Information GUiDE (INTRIGUE) for the city of Torino, Italy. It is a fuzzy logic based recommender system that is able to provide an explanation why a recommendation has been made. Moreover, INTRIGUE has a tour scheduling functionality that enables composing group tours taking into account opening hours and locations of the POIs and time restrictions of the tourist group members. Unfortunately, very little technical details on the scheduling algorithm are provided.

[Suna and Lee \(2004\)](#) present a multi-agent system that advises personalised tourist routes using a vector based recommendation technique to calculate personal interest values in geographical objects. A shortest path algorithm minimises the normal cost of arcs in the road network divided by their personal interest values in order to calculate personalised point-to-point routes. A tourist trip is calculated by modelling the POI selection problem as a prize collection TSP, which is to be solved with the method of [Dell'Amico et al. \(1998\)](#). Despite the promising ideas, [Suna and Lee \(2004\)](#) do not evaluate the system thoroughly.

[Maruyama et al. \(2004a,b\)](#) present P-Tour, a personal navigation device that calculates tourist routes. They use a variant of the TSP with profits that aims at finding a circuit that minimises travel costs minus collected profit ([Feillet et al. 2005](#)). The P-tour routing algorithm selects and routes a number of POIs that are defined by a location, a visiting duration, an importance score and an optional constraint on arrival time. [Maruyama et al. \(2004a\)](#) maximises the weighted sum of (1) the importance of selected POIs, (2) the importance of selected POIs that satisfy a time restriction, minus (3) the total travel distance, while [Maruyama et al. \(2004b\)](#) only use (2) minus (3). The route search engine achieves a gap of less than 2% from the optimal solution in 15 seconds, for one small instance of 30 POIs. Users have to enter personal importance scores themselves for each POI.

[Shiraishi et al. \(2005a,b\)](#) extend P-Tour in two ways. First, they search for undesirable situations during the execution of the planned route (wrong route, behind or ahead of schedule) and warn the user in an appropriate manner. Second, the route search engine is extended to take multiple conflicting evaluation functions into account: they contrast the weighted sum function of [Maruyama et al. \(2004b\)](#) together with the total travel expense. [Kinoshita et al. \(2006\)](#) extend P-Tour for multiple days. The total set of POIs is partitioned across different areas. Every day a selection of POIs of a predetermined area is to be visited, including accommodation to spend the night. A three day instance is solved in 19.6s with a gap of 0.7% from optimality, taking into account the pre-partitioning of the POIs, which possibly discards significantly better solutions. [Nagata et al. \(2006\)](#) extend the P-Tour system in order to plan group tours. Every member of the tourist group is allowed to state a preference value, a duration and a latest arrival time for each POI. Also, every member has a

starting and an end point, a time restriction on the total tour and a speed. The objective is to find a schedule that forks and joins the group along the way to visit POIs.

Wu et al. (2009) extend the P-Tour system to take the weather forecast into account. For each POI, a timetable is given that contains probabilities for fine, cloudy and rainy weather, every hour. POI preference values are dependent on the current weather while visiting it. The objective is to construct a decision tree that maximises the total expected satisfaction degree. Limited experimental results are presented: an instance of 6 POIs is optimally solved in 6 seconds, compared to 16h with brute force search, and on an instance with 20 POIs, the quality of the greedy construction heuristic is improved 17,9% on average. Overall, P-Tour and its extensions provide interesting ideas, but fail to provide extensive computational experiments.

The Dynamic Tour Guide (DTG) of ten Hagen et al. (2005) calculates personal tours on-the-fly. A tour is a collection of so-called Tour Building Blocks (TBB): sights, restaurants, etc. An ontology consists of a tree of concept classes and describes each TBB. TBBs receive Interest Matching Points by a semantic matching algorithm, representing the interest of the user. After the assignment of Interest Matching Points, an algorithm constructs a tour by maximising the sum of the Interest Matching Points within a given time frame. The algorithm keeps a list of candidate TBBs, sorted by use of gain, i.e. the Interest Matching Point divided by the cost needed to visit the TBB. From the list, the TBBs are removed and inserted in the tour randomly until the available time runs out.

Lee et al. (2007) present a tourist tour planning system for the Jeju province in Korea. They adopt the interest estimation method of Kang et al. (2006) in order to compute vector-based similarities between POIs and users. Next, they add a maximum length constraint to their TSP formulation of the planning problem. Actually, without mentioning it, they use the OP as a model. However, they tackle the problem by solving 2^n distinct TSPs of n POIs, instead of 1 OP, which is a computationally very expensive approach. Their high performance cluster manages to offer a solution within 5 seconds when $n < 22$.

Castillo et al. (2008) present a multi-agent based system for planning tourist visits. A user agent first captures the user's interest. Next, a Case Based Reasoning agent predicts interesting activities. Finally, the planning agent takes these interesting activities as input and outputs a plan. The planning takes the following items into account: opening hours, preferences of the user, prices of meals and transports, locations and multi-modal means of transport. Two types of goals can be specified: totally and partially instantiated goals, e.g. visit a specific museum, respectively a type of museum. The planning problem is translated to predicate logic. Predicate logic AI planning modules use tree search to come up with a feasible plan and route to perform the activities. There is no integration of selection and routing, and no evaluation of the proposed system is presented.

Lee et al. (2009) present a recommendation system that allows planning personalised travel routes to Tainan City, China. Their ontology based multi-agent system consists of a context decision agent and a travel route recommendation

agent. The context decision agent first finds concepts of the ontology that match the tourist's requirements. Next, the travel route recommendation agent uses fuzzy logic to select and sort a top three of historic sites and a top five of local gourmet food stores. A TSP that deals with these eight locations, is solved. Experimental results present two small examples as evaluation, but no performance benchmarks. The system makes a distinction between selection and routing.

[Niaraki and Kim \(2009\)](#) developed a method for personalising route planning network impedances. They evaluate multiple criteria that are defined in an ontology describing road segments. The user states his preferences for attributes such as traffic volume, safety, POI presence, etc. based on which the weights in the road graph are calculated. Common shortest path algorithms such as Dijkstra ([Gallo and Pallattino, 1986](#)) and A* ([Pearl, 1984](#)) can be used to calculate personal point-to-point routes.

[Yu and Chang \(2009\)](#) developed a framework for the personalised recommendation of hotels, restaurants and POIs. They have combined these three functionalities in a tour recommendation process that recommends a personalised tour based on the user's current time and location and his interests. A prototype was built for the city of Taipei, Taiwan.

We developed the City Trip Planner², which is a web-based tourist decision support system that proposes city trips tailored to the user's context and personal interests ([Vansteenwegen et al., 2010](#)). The system plans city visits of multiple days, with for each POI multiple time windows which can differ from day-to-day. Moreover, lunch breaks can also be scheduled and the local tourist office can suggest a few POIs to be included in a trip. The City Trip Planner integrates the selection of POIs and the routing between them. It uses the OP to model trip planning problems and fast heuristic algorithms ([Vansteenwegen et al., 2009](#)) to solve them. To the best of the author's knowledge, such an integrated system is unique. An extensive evaluation, with user statistics, is available in [Vansteenwegen et al. \(2010\)](#).

3 Planning Functionality for Tourist Decision Support

This section first discusses the wide range of planning-related functionalities that are offered by the tour scheduling systems. Next, a comparison between these different systems is provided, based on the functionality they offer.

Personal Interest Estimation quantifies the interest of the tourist in a particular POI, the appropriateness of a hotel or the "beautifulness" of a scenic route. This quantified value can be used to sort POIs and hotels when presenting them to the user. Attributes of the user, which are collected into his user profile, are to be matched with attributes of the location or activity.

Selection and Routing automatically presents a customised route based on the user's current location, his destination and his available time, which limits the

² <http://www.citytripplanner.com>

length of the route. This limit implies a selection of POIs, in order to keep the length of the route feasible. When combined with personal interest estimation, the resulting route is tailored to the user's interest.

Mandatory POIs can be considered as “must see”. Whenever the tourist is in the neighbourhood of a mandatory POI, it should be presented at the top of his preference list. A POI can be determined mandatory by the provider of the POI information.

Dynamic Recalculation is needed when unexpected events occur. These can make the current selection of POIs or route invalid. Dynamic recalculation detects these infeasibilities and presents a new plan to the user, in “real-time”.

Multiple Day Decision Support enables planning for multiple days. The user receives a selection of POI visits for a series of days. Each POI visit only appears once in the total selection.

Opening Hours should be taken into account when visiting the “interior” of POIs. Therefore, the route of selected POIs should take into account the time of the scheduled POI visits, making sure that each visit is planned when the POI is open. The opening hours of each POI are defined by means of a calendar. In the simplest case, a POI is open for a consecutive period during the day, with one opening time and one closing time, for all days. However, POIs can e.g. be closed during lunch, resulting in two opening periods during the same day. Moreover, opening hours tend to differ on different days: a POI can e.g. be closed on Sunday afternoon.

Budget Limitations arise when the tourist has a maximum amount of money to spend. Next to the time budget of the selection and routing functionality, a money budget further constrains the selection of POI visits.

Weather Dependency influences the estimated appreciation of POIs by taking the weather forecast into account. During rainy periods, outdoor visits could be penalised, in favour of indoor visits.

Max-n Type constrains the selection of POIs by allowing to state a maximum number of certain types of POIs, per day or for the whole trip. E.g. maximum two museum visits on the first day.

Mandatory Types enable the tourist to state that a tour or a trip should contain at least one visit of a certain type, e.g. a visit to a church. Mandatory types extend the concept of mandatory POIs.

Scenic Routes allow to visit beautiful routes, next to interesting locations. When moving from one POI to the next, a tourist will not mind a small detour through a car-free street with medieval façades. Although this is not the shortest path between the two POIs, it will be appreciated more than a walk through a regular street.

Hotel Selection automatically selects appropriate hotels when visiting a region for multiple days. The personal interest in the different hotels can be estimated, based on attributes of the hotels such as comfort. The automated selection mechanism will need to take the price of a stay into account, in function of the budget of the total trip.

Public Transportation takes into account metro, train and bus schedules when travelling between POIs. These alternatives to walking need to be considered when distances between POIs are large. They can save considerable amounts of time that could be spend on POI visits.

Group Profiles enable planning for groups of tourists, which differs considerably from single-tourist planning, as a group of tourists may have a broad, possibly conflicting, range of interests. Possible strategies include optimising the joint interests of the group members by selecting locations they are all interested in, or taking turns and alternating interests so that no one feels he has been left out.

Table 1. Functionality Overview

	personal interest estimation	distinct selection and routing	integrated selection and routing	mandatory POIs	multiple days	opening hours	budget	limitations group profiles
(Soo and Liang, 2001)	x		x					
(Ardissono et al., 2002, 2003)	x	?	?			x		x
(Suna and Lee, 2004)	x		x					
(Maruyama et al., 2004a,b)			x			x		
(Shiraishi et al., 2005a,b)			x			x	x	
(Kinoshita et al., 2006)			x		x	x		
(Nagata et al., 2006)			x			x		x
(Wu et al., 2009)						x		
(ten Hagen et al., 2005)	x		x					
(Lee et al., 2007)	x		x					
(Castillo et al., 2008)	x	x		x		x	x	
(Lee et al., 2009)	x	x			x			
(Niaraki and Kim, 2009)	x							
(Yu and Chang, 2009)	x		x					
(Vansteenwegen et al., 2010)	x		x	x	x	x		

Table 1 presents a match of the existing tour scheduling approaches with the different functionalities presented above. Only those functionalities that appear in two or more approaches are included. Unknown features have been marked as “?”. For the sake of completeness, Castillo et al. (2008) are the only to mention mandatory types and public transportation, Shiraishi et al. (2005a,b) are the only to tackle dynamic recalculation, Wu et al. (2009) weather dependency, Niaraki and Kim (2009) scenic routes and Kinoshita et al. (2006) hotel selection.

All of the presented approaches use some form of personal interest estimation, except for P-Tour and its extensions, which in turn present a wide range of planning functionalities.

A number of approaches offer integrated selection and routing. [Soo and Liang \(2001\)](#); [Yu and Chang \(2009\)](#) use a nearest neighbour approach, which iteratively adds the closest available visit to the tour. [Suna and Lee \(2004\)](#), P-Tour and its extensions, except [Wu et al. \(2009\)](#), use the profitable tour problem as a basis for selection and routing. [ten Hagen et al. \(2005\)](#) use a tree-based search, while [Lee et al. \(2007\)](#) combine a selection problem with a TSP.

Based on these developments, it can be concluded that providing automated POI selection and routing is an upcoming trend in tourist recommender applications. It appears that a large amount of research effort is still required in order to devise efficient tourist decision support techniques, that are able to propose customised tours with acceptable response times. Providing adequate planning support for these kinds of applications is a huge research opportunity in the field of OR.

4 Modelling the Tourist Trip Planning Problem

This section explains how the OP and its extensions can be used to model trip planning functionalities. This approach is applied, for instance, in the City Trip Planner mentioned in Section 2. An OP is a mathematical optimisation problem that consists of a set of locations which are determined by coordinates and a score. The pairwise travel times between the locations are known. The goal is to find a tour that maximises the total score earned by visiting locations. The start and end of the tour do not need to coincide. The total travel time (or distance) cannot exceed a predetermined value, which is called the time budget. Each location can be visited at most once.

All locations with a score represent POIs. The score represents the estimated personal interest of the tourist in the POI, and can be calculated by means of POI recommender techniques. The time budget obviously represents the maximum amount of time the tourist has available. The time needed to visit a location is normally ignored in the OP. However, this visiting time can be added to the travel times between the locations: half of the visiting time is added to each incoming travel time, half to each outgoing travel time. A solution to the OP represents a tourist route. It is obvious that solving the OP entails an integrated solution of the selection and routing problem a tourist faces. We include mandatory POIs as locations with a score that is higher than the sum of the scores of the non-mandatory POIs. A quality algorithm will always select these mandatory POIs if that is feasible. Dynamic recalculation can be achieved by a fast algorithm, capable of offering a new solution in case of unexpected events that lead to a new TTDP instance.

The “Team OP” (TOP) extends the OP by allowing multiple tours, each limited by a time budget. Again, each location can be visited at most once. The TOP allows to model TTDPs for multiple days. Each tour or vehicle represents one day from a multi-day tourist trip.

The TOP can be extended with multiple constraints, in which each location has Z attributes for each day. Z additional constraints are defined, which limit the selection of vertices. In the envisioned tourist application, we use these additional constraints to enable modelling budget limitations to spend on entrance fees, “max- n types” for each day and for the whole trip and mandatory POI types.

In case of budget limitations, an extra location attribute is used to represent the entrance fee for a POI, and an extra constraint defines the money budget available to spend. Max- n types are modelled in a similar way: an extra location attribute is set to 1 if the location is of a particular type, 0 otherwise, and an extra constraint defines the maximum number of visits of that type. Note that the model also enables max- n type and budget constraints to be defined per day, e.g. visit maximum one church on the first day, or spend at most €100 on the second day. In this case, an extra constraint is added for that particular day.

Mandatory POI types, e.g. visit at least n churches, are a bit more complicated to model. Every visit to a POI of this type is copied. The copied visit receives a score that is higher than the sum of all regular visits, cfr. a mandatory POI. For each couple, an extra constraint is added, indicating that the original visit and the copied visit are mutually exclusive. For all copied visits of the considered type, one extra attribute is added and set to 1 for a copied visit, 0 for an original visit, and one extra constraint is added, which limits the total selection to the preferred number of visits of that type. When during the search this preferred number is not yet reached, the copied visits will be preferred over their original counterparts. When the number is reached, all other copied visits of the type under consideration will not influence the search any further. However, the original POI visits will still be considered with their normal scores.

The most studied extension of the TOP is the TOP with Time Windows (TOPTW). In this extension, each location is assigned a TW, with an opening time and a closing time. A visit to a location can only start during this time window. On arrival before opening, waiting is allowed, until opening, in order to collect the score. A feasible solution does not violate any TW constraint. The TOPTW enables modelling opening hours of POIs in the TTDP. However, only one TW can be defined per location, implying that a POI can have only one opening and closing time per day. Moreover, in the case of multiple days, a POI has an identical opening and closing time for any day.

We overcome this drawback by extending the TOPTW to the Multiple Constraint TOP with Multiple Time Windows. Each location is extended allowing W TWs for each day, instead of one. Also, the TWs can be different on different days. This enables modelling opening hours of POIs by allowing multiple opening hours per day, and different opening hours each day.

The multiple constraints extension described above, can be used to model multiple TWs: locations with multiple TWs are split up into different visits to the same location, with one TW, and adding an extra constraint allowing only one visit to the actual location. Moreover, weather dependency can also be modelled by splitting up a POI: the POI with a TW marking a sunny period

has a higher score than the same POI with a TW for a rainy period, and only one of both can be visited.

The City Trip Planner system, described above and evaluated in [Vansteenwegen et al. \(2010\)](#), proves that the OP and its extensions are very appropriate to model personalised trip planning. If the tourist trip planning problem is modelled as an (extension of the) OP, a large battery of algorithms are readily available for re-usage. The interested reader is referred to [Vansteenwegen et al. \(2011\)](#) for a recent survey on solution techniques.

5 Conclusions

Providing adequate tour scheduling support for tourist decision support applications is a daunting task for the application developer. An overview of systems that compose tours of POI visits, is presented. The wide range of planning-related functionalities that are offered by these systems, is discussed and the systems are compared based on these functionalities. Next, this paper demonstrates that existing OR models enable a wide range of current and future tourist trip planning functionalities.

We use the basic OP model, as it integrates selection and routing of tourist attractions. Mandatory POIs can be easily incorporated and dynamic recalculation is achieved by using fast solution techniques, which present solutions in nearly real-time. The model is iteratively extended with multiple tours, multiple constraints, time windows and multiple time windows. This paper explains how these extensions can be used to enable the planning of multiple days, budget limitations, max-n types, mandatory POI types, taking opening hours of POIs into account and how to tackle weather dependency.

The City Trip Planner described above shows that algorithms are available to deal with these models for real-life tourist planning problems. In addition to the City Trip Planner, we have also incorporated the model into another web-based tourist decision support system, namely an on-line cycle route planner for the province of East-Flanders offers personalised cycle routes based on user preferences. This web-based system is extended with an SMS service that provides cyclists “in the field” with routes on demand ([Souffriau et al. 2010](#)).

Future work includes incorporating support for scenic routes, hotel selection, public transportation and group profiles.

References

- Ardissono, L., Petrone, G., Segnan, M., Torasso, P.: Ubiquitous user assistance in a tourist information server. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, pp. 14–23. Springer, Heidelberg (2002)
- Ardissono, L., Goy, A., Petrone, G., Signan, M., Torasso, P.: Intrigue: personalized recommendation of tourism attractions for desktop and handset devices. *Applied Artificial Intelligence* 17(8-9), 687–714 (2003)

- Castillo, L., Armengol, E., Onaindía, E., Sebastián, L., González-Boticario, J., Rodríguez, A., Fernández, S., Arias, J.D., Borrajo, D.: Samap: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications* 34, 1318–1332 (2008)
- Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.: Developing a context-aware electronic tourist guide: Some issues and experiences. In: *Proceedings of ACM CHI Conference on Human Factors in Computer Systems*, The Hague (2000)
- Dell' Amico, M., Maffioli, F., Sciomachen, A.: A lagrangian heuristic for the prize collecting travelling salesman problem. *Annals of OR* 81, 289–306 (1998)
- Dunlop, M., Ptasiniski, P., Morrison, A., McCallum, S., Risbey, C., Stewart, F.: Design and development of Taeneb city guide - from paper maps and guidebooks to electronic guides. Technical report (2004), http://www.cs.strath.ac.uk/~mdd/research/publications/04dunlop_enter.pdf
- Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. *Transportation Science* 39, 188–205 (2005)
- Gallo, G., Pallattino, S.: Shortest path methods: a unified approach. *Mathematical Programming Study* 26, 38–64 (1986)
- Godart, J.M.: Combinatorial optimisation for trip planning. *Belgian Journal of Operations Research, Statistics and Computer Science* 41(1-2), 59–68 (2001)
- Kang, E., Kim, H., Cho, J.: Personalization Method for Tourist Point of Interest (POI) Recommendation. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) *KES 2006. LNCS (LNAI)*, vol. 4251, pp. 392–400. Springer, Heidelberg (2006)
- Keyson, D.V.: An electronic mobile guide for Artis zoo. Technical report, Intelligence in Products Group, Faculty of Industrial Design, Delft University of Technology (2004)
- Kinoshita, T., Nagata, M., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: A personal navigation system for sightseeing across multiple days. In: *Proc. of the 3rd Int'l. Conf. on Mobile Computing and Ubiquitous Networking (ICMU 2006)*, pp. 254–259 (2006)
- Kramer, R., Modsching, M., ten Hagen, K.: A city guide agent creating and adapting individual sightseeing tours based on field trial results. *International Journal of Computational Intelligence Research* 2(2), 191–206 (2006)
- Lee, C.-S., Chang, Y.-C., Wang, M.-H.: Ontological recommendation multi-agent for tainan city travel. *Expert Systems with Applications* 36, 6740–6753 (2009)
- Lee, J., Kang, E., Park, G.-L.: Design and Implementation of a Tour Planning System for Telematics Users. In: Gervasi, O., Gavrilova, M.L. (eds.) *ICCSA 2007, Part III. LNCS*, vol. 4707, pp. 179–189. Springer, Heidelberg (2007)
- Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: P-tour: A personal navigation system for tourism. In: *Proceedings of 11th World Congress on ITS*, pp. 18–21 (2004a)
- Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: A personal tourism navigation system to support traveling multiple destinations with time restrictions. In: *Proc. of the 18th Int'l. Conf. on Advanced Information Networking and Applications (AINA 2004)*, pp. 18–21 (2004b)
- Nagata, M., Murata, Y., Shibata, N., Yasumoto, K., Ito, M.: A Method to Plan Group Tours with Joining and Forking. In: Wang, T.-D., Li, X., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) *SEAL 2006. LNCS*, vol. 4247, pp. 881–888. Springer, Heidelberg (2006)
- Niaraki, A.S., Kim, K.: Ontology based personalized route planning system using a multi-criteria decision making approach. *Expert Systems with Applications* 36, 2250–2259 (2009)

- Oppermann, R., Specht, M.: A nomadic information system for adaptive exhibition guidance. *Archives & Museum Informatics* 13, 127–138 (1999)
- Pearl, J.: *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading (1984)
- Shiraishi, T., Nagata, M., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: A personal navigation system with a schedule planning facility based on multi-objective criteria. In: *Proceedings of 2nd International Conference on Mobile Computing and Ubiquitous Networking*, pp. 104–109 (2005a)
- Shiraishi, T., Nagata, M., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: A personal navigation system with functions to compose tour schedules based on multiple conflicting criteria. *IPSJ Digital Courier* 1, 528–536 (2005b)
- Soo, V.-W., Liang, S.-H.: Recommending a Trip Plan by Negotiation with a Software Travel Agent. In: Klusch, M., Zambonelli, F. (eds.) *CIA 2001. LNCS (LNAI)*, vol. 2182, pp. 32–37. Springer, Heidelberg (2001)
- Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D.: The planning of cycle trips in the province of east flanders. *Omega, The International Journal of Management Science* (2010) (in press)
- Suna, Y., Lee, L.: Agent-based personalized tourist route advice system. In: *SPRS Congress Istanbul 2004, Proceedings of Commission II*, pp. 319–324 (2004)
- ten Hagen, K., Kramer, R., Hermkes, M., Schumann, B., Mueller, P.: Semantic matching and heuristic search for a dynamic tour guide. In: *Information and Communication Technologies in Tourism*. Springer, Heidelberg (2005)
- Vansteenwegen, P., Van Oudheusden, D.: The Mobile Tourist Guide: an OR Opportunity. *OR Insight* 20(3), 21–27 (2007)
- Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* 36, 3281–3290 (2009)
- Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The city trip planner: an expert system for tourists. *Expert Systems with Applications* (2010) (under review)
- Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: a survey. *European Journal of Operational Research* (2011) (in press)
- Wu, B., Murata, Y., Shibata, N., Yasumoto, K., Ito, M.: A method for composing tour schedules adaptive to weather change. In: *Proc. of 2009 IEEE Intelligent Vehicles Symposium (IV 2009)*, pp. 1407–1412 (2009)
- Yu, C.C., Chang, H.p.: Personalized Location-Based Recommendation Services for Tour Planning in Mobile Tourism Applications. In: Di Noia, T., Buccafurri, F. (eds.) *E-Commerce and Web Technologies. LNCS*, vol. 5692, pp. 38–49. Springer, Heidelberg (2009)

Personalized Tourist Route Generation

Ander Garcia¹, Olatz Arbelaitz², Maria Teresa Linaza¹,
Pieter Vansteenwegen³, and Wouter Souffriau³

¹ Department of eTourism and Cultural Heritage
Visual Communication Technologies Vicomtech
Paseo Mikeletegi 57, E-20009
Donostia - San Sebastian, Spain
agarcia@vicomtech.org

² Department of Computer Architecture and Technology
University of the Basque Country
Donostia - San Sebastian, Spain

³ Centre for Industrial Management
Katholieke Universiteit Leuven
Leuven, Belgium

Abstract. When tourists are at a destination, they typically search for information in the Local Tourist Organizations. There, the staff determines the profile of the tourists and their restrictions. Combining this information with their up-to-date knowledge about the local attractions and public transportation, they suggest a personalized route for the tourist agenda. Finally, they fine tune up this route to better fit tourists' needs. We present an intelligent routing system to fulfil the same task. We divide this process in three steps: recommendation, route generation and route customization. We focus on the last two steps and analyze them. We model the tourist planning problem, integrating public transportation, as the Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW) and we present an heuristic able to solve it on real-time. Finally, we show the prototype which generates and customizes routes in real-time.

1 Introduction

Nowadays, the generation of personalized tourist routes that take into account the profile of the tourist and up-to-date attraction information is a time consuming task. During the pre-trip stage, tourists have to spend their time collecting information about the destination in order to start planning the stay. This is a difficult task because they have to choose the POIs they are going to visit and estimate visits duration and traveling times.

When tourists are at a destination, this task is often done by the staff of the Local Tourist Organizations (LTOs). The staff determines the profile of the tourists (cultural, romantic, family, etc.) and their restrictions (time, money, etc.). Combining this information with knowledge about local attractions (location, price, timetable, etc.), the LTO suggests a personalized route for the

tourist. Finally, they make small changes to this personalized route based on the feedback from the tourist.

A Personalized Electronic Tourist Guide (PET) should perform at least the same task fulfilled by the LTO, on a hand-held device, providing three main functionalities: recommendation, route generation and customization. A PET should select the most interesting POIs for tourists (recommendation); generate routes in real-time maximizing tourists' satisfaction taking into account different restrictions, attractions' attributes (opening and closing times, visit duration, entrance fee, etc.) and traveling times (route generation); and allow further adaptations to the route (customization).

Furthermore, public transportation information should be integrated, since that is identified as one of the most appreciated functionalities of a PET [1,2,3]. In this paper we integrate public transportation and present a prototype that focuses on the route generation and customization.

In Section 2 we review the state of the art regarding PETs and Operations Research (OR) algorithms. In Section 3 we give an overview of the general functionalities of a PET we are interested in. In Section 4 we focus on the route generation algorithm. In section 5 we introduce the customization functionalities. In Section 6 we show our prototype. Finally in Section 8 we summarize the conclusions and some further work.

2 State of the Art

Vansteenwegen [4] and Souffriau [5] present an extensive review of existing Personalized Electronic Tourist Guides (PETs). However, none of these guides uses advanced heuristics to solve the planning problem nor integrates the use of public transportation. Although Dynamic Tour Guide [6] generated and recalculated routes in real-time (less than five seconds), the routing algorithm was very simple and it had important restrictions: it could only create proper solutions for one day routes and a small number of POIs. A PET called P-Tour [7] applied a genetic algorithm to calculate routes. Nevertheless, tourists had to manually enter the POIs they wanted to visit with their details (visiting time, duration and tourist score). Moreover, the system needed nearly ten seconds to obtain a route for just twelve POIs. Again, no public transportation was taken into account. Castillo et al. [8] described a multiagent based system for planning tourist visits. Although they mentioned transportation, they did not integrate selection and routing nor give hints or details about the implementation or performance. Zenker et al. [9] presented ROSE, a mobile application assisting pedestrians to find preferred events and comfortable public transport connections composed by three main services: recommendation, route generation and navigation. They identified the route planning problem to solve and they described it as a multiple destination recommendation with public transportation and time windows. This is the same problem we propose. However, they did not find any algorithm able to solve it.

The requirements of the problem a PET has to solve can be modeled as the Tourist Trip Design Problem (TTDP) [10,11]. The Orienteering Problem (OP)

[12] is the most basic version of the TTDP. Its generalization to a multiple-day trip is known as the Team OP (TOP). Vansteenwegen et al. [13] present an extensive review of these problems, solution algorithms and applications. When POIs have an associated time window, the problem is called TOP with Time Windows (TOPTW) [14]. Interested readers can find a thorough revision of literature about OPTW and TOPTW in [15,16,17,18].

Fomin and Lingas [19] present the Time Dependent OP (TDOP), an extension of the OP where the time needed to travel from a POI i to a POI j depends on the leave time from i . Implicitly, they can deal with different transportation modes. However, they don't present an algorithm that can be used in real-time applications and no time windows or multiple-day trips are considered.

The model we propose is the Time Dependent TOPTW (TDTOPTW) [20], which combines the previous models and makes possible to integrate one or more public transportation networks to travel between POIs. The travel time required to move from one POI to the next, will vary according to the transportation mode (walking or public transportation) and the leave time. To the authors' understanding, there is no algorithm able to solve the TDTOPTW.

Although examples are present of finding the shortest path in public transportation networks or time dependent networks, they all focus on solving individual queries. Pyrga et al. [21] provide extensive information about the Earliest Arrival Problem (EAP). Ding et al. [22] focused on finding the best departure time to minimize the total travel time on time dependent graphs. A large number of public transportation providers implement these algorithms. For example, the public transportation service provider of San Sebastian has a best path finding application to move between two points (<http://www.dbus.es>).

As examples of more advanced applications, Zografos and Androutsopoulos [23] implemented an algorithm to calculate optimum itineraries in Athens's urban public transport system, including a stop at an intermediate point. In Hong-Kong, a similar system was tested using a different approach [24]. Within the tourism field, PECITAS [25] is a mobile personal navigation system to find the best path between two POIs regarding both the travel time and tourist's preferences. All these applications focus on POI-to-POI cheapest cost problems, while this paper focuses on a higher level selection problem of POIs and a routing problem between multiple POIs.

3 Personalized Electronic Tourist Guide (PET)

Souffriau [5] classifies the different functionalities PETs offer. Regarding the route generation, they can be gathered in three main basic functionalities (Figure 1). In Figure 1 we also represent the inputs and outputs of each step.

- **Recommendation.** Combining the tourist information of the destination with the tourist profile, the system has to create a list of recommended POIs. As a result, POIs will have a different score and visit duration for different tourists. Interested readers will find a deep review on tourist recommendation systems on the recent paper from Kabassi [26].

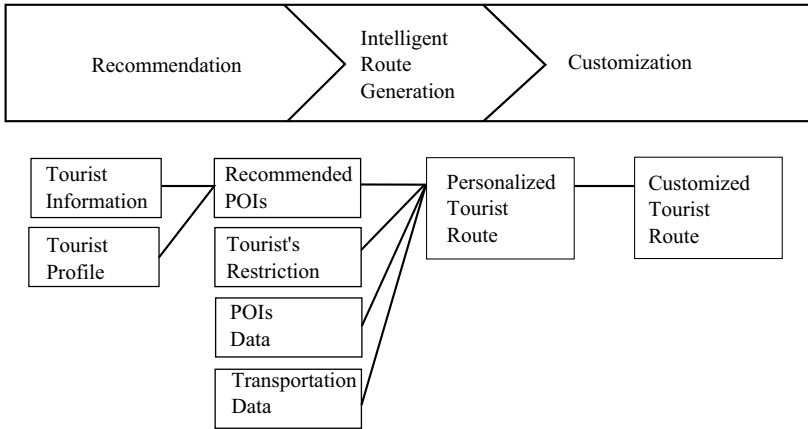


Fig. 1. PET flow

- **Route Generation.** Once the system knows which are the most appealing POIs for the tourist, an intelligent routing engine can combine this information with tourist’s restrictions (available time, number of days of the route, maximum budget, starting POI ...), POI data (location, opening hours, ticket price ...) and transportation data (traveling times, public transportation network data ...) to generate personalized tourist routes. This intelligent routing engine has to apply advanced algorithms from the field of Operations Research (OR) in order to generate routes in real-time.
- **Customization.** Finally, tourists appreciate to have the opportunity to customize the proposed personalized route to better fit their needs. Inserting new visits to the route, removing visits or reordering them are the basic functionalities of a customization engine.

The system we present in this paper focuses on the last two functionalities. First (Section 4) it applies an advanced OR algorithm to generate personalized tourist routes that take advantage of the existing public transportation network. Once a personalized route is generated, it gives to the tourists the opportunity to customize it (Section 5).

4 Route Generation

We have modeled the problem of generating personalized tourist routes with public transportation as a TDTOPTW. After analyzing different approaches, we finally designed a heuristic able to solve it in real-time [20,27], based on a fast heuristic for the TOPTW [17]. The inclusion of public transportation makes the process of building a solution much more complex: each distance calculation between two POIs becomes a Time Dependent Shortest Path problem (TDSP). A small increase in the planned leave time from one POI can cause a significant

increase in the arrival time to the next POI. For instance, when a tourist just misses the bus and has to wait for the next one or walk to the next POI.

Thus, in order to handle the difficulty of the TDTOPTW, we apply a hybrid approach combining two different heuristics. Each of them focuses on a different aspect of the problem.

First we solve in batch the TDSP problems between all the possible pairs of POIs with leave time steps of 1 minute. Then, we calculate an average travel time between each pair of POIs. Since these calculations can be made beforehand, the real-time requirement is eliminated for this part of the calculation and any suitable algorithm can be used to solve the TDSP problems. Delling [28] details the latest algorithms fulfilling this task. We have implemented a time dependent Dijkstra's Shortest Path algorithm modified to cope with public transportation [20]. Based on the conclusions of Pyrga et al. [21], we have applied a time dependent approach with simple transfer times. This way, we handle the extra difficulty added by the public transportation without the real-time requirement.

The complete calculation of the average travel time matrix for an instance with 50 POIs takes around 90 minutes on a PC Intel Core 2 Quad with 2.40 GHz processors and 2 GB Ram. The average travel times are stored in a database for a later fast retrieval.

Once we have calculated these average travel times, we solve the TDTOPTW in real-time as a regular TOPTW. Our design and implementation is based on the algorithm proposed by Vansteenwegen et al. [17] for the (T)OPTW. We give a general description of the algorithm here, for more details we refer to the (T)OPTW article.

The heuristic is based on Iterated Local Search (ILS) [29]. This method combines iteratively a local search with a perturbation phase. The local search heuristic inserts new visits to a route, one by one. For each visit i that can be inserted, the cheapest insertion time ($Shift_i$) is determined. For each of these visits the heuristic calculates a ratio, which relates the tourist score of the POI (the estimated satisfaction for a given tourist) to the time required to visit it. Among them, the heuristic selects the one with the highest ratio for insertion. This process is repeated until no more POIs can be inserted. A perturbation phase removes consecutive POIs from a route. After the removal, the heuristic shifts all visits following the removed visits towards the beginning of the route as much as possible, in order to avoid unnecessary waiting. The perturbation procedure and the local search heuristic are executed until a termination criterion is met. The heuristic returns the incumbent solution as the result.

Finally, once we find a final solution, we run a repair procedure introducing the real travel times between the POIs. Starting from the first POI of the route, we compare the average and the real travel time, taking into account the real leave time. If the real travel time is smaller than the average one, the travel time is adapted by advancing the arrive time to the visit towards the beginning of the route as much as possible. The waiting time and the leave time of the visit are also updated.

Otherwise, if the real travel time is larger than the average one, we arrive later to the visit. If this causes a visit to become infeasible, we remove it from the route and we update the route, moving the rest of the visits towards the beginning.

5 Route Customization

Once the system has generated a personalized route that maximizes the tourist satisfaction, the tourist can decide to change this route to better fit his/her own interests. Although this customization step is voluntary, it greatly increases the flexibility of the system.

The customization is based on six basic operations a PET should provide. The difficulty of these operations is directly related to the public transportation network. As the leave time from the previous POI changes, the travel time to the next POI does too. Thus, all the affected traveling times have to be recalculated. This process varies in function of the executed operation:

- **Add a visit.** When a tourist wants to add a new visit to a route, there are two possibilities to determine the insertion position. In the first one, the system finds the best insertion point, minimizing the total time (travel plus wait plus visit times). The second possibility consists on the tourist choosing the insertion order within a day. In any case, the system has to update the order of the following visits. The system has to insert the new visit and update the travel and wait times starting from the previous visit.
- **Remove a visit.** In this case the system has to remove a visit. Thus, all the following visits of the day are moved towards the beginning. Starting from the visit preceding the removed one, traveling and waiting times have to be recalculated until the end of the day is reached.
- **Move a visit towards the beginning of the route.** When a tourist wants to visit a POI earlier than scheduled, the system swaps this visit and the previous one. Starting from the one before the previous visit, traveling and waiting times have to be recalculated until the end of the day is reached. Obviously, this move can be repeated to schedule the visit even earlier.
- **Move a visit towards the end of the route.** When a tourist wants to visit a POI later than scheduled, the system swaps this visit and the next one. Starting from the previous visit, traveling and waiting times have to be recalculated until the end of the day is reached.
- **Move a visit to the previous/next day.** When a tourist wants to move a visit to the previous/next day, first the system has to remove the visit from the actual day. Once this operation is completed, the system finds the best insertion position (smallest required extra time) for the POI on the previous/next day.
- **Customization exception.** Once a tourist starts customizing a route, it is possible that a violation of some of the restrictions occurs. As the customization is a hand-made process realized by a tourist, in these cases the system shows an alert asking for confirmation before committing the change.

We have created the following exceptions to ask for explicit confirmation during the customization process:

- Maximum travel time/ budget exceed alert. This alert notifies that the maximum values introduces as restrictions have been exceeded.
- High wait/travel alert. This alert is launched if the proportion between visiting time and wait/travel time is high (higher than 40%).

We describe the prototype integrating the previous operations in Section [6](#).

6 Prototype

The prototype has been initialized with data about the city of San Sebastian. San Sebastian is a Spanish tourist city located near the French border. It has around 200.000 inhabitants and 50 POIs. Most tourists visit the city by combining public transportation with short walks. The public transportation network includes 467 bus stops, 26 lines and more than 65.000 direct bus connections between stops. The prototype has two different frontends. The first one is targeted for desktop interaction on the pre-trip stage. The second one is targeting mobile devices with a touch friendly user interface.

In relation to the application's interaction with the user, first of all, tourists have to enter their restrictions. Moreover, the system has to know their profile in order to assign the correct tourist score to each POI. Although there are advanced approaches available on the literature [\[26\]](#), this step is out of the scope of this paper. Thus, we have followed a simple approach, having four different profiles (nature, culture, gastronomy, leisure). For each POI and profile, the system administrator has to assign the estimated tourist score (0-100) and the estimated visit duration. Tourists have the opportunity to overwrite these scores. They can update both the tourist score and the visit duration. Besides, they can mark some POIs as must-seen/must-avoid POI. The system automatically assigns these POIs a tourist score of 100/0.

With this input data, the system generates a personalized tourist route in real time (less than 5 seconds). This route takes advantage of the public transportation network of the city, shortening the traveling times and increasing the mobility of tourists. Regarding the traveling information, the travel between the previous POI and the actual POI is divided in transportation links. Each transportation link stores details about how the traveling is done. If a tourist goes on foot the distance between POIs and the estimated travel times are stored. Otherwise, for each public transportation involved, the traveling distance from/to the previous/actual POI/stop are stored, containing details about the public transportation line, the estimated waiting and traveling times.

The following example details how to arrive from a hotel outside the city center (NH) to the cathedral of San Sebastian (Buen Pastor):

- Walk 2 minutes from hotel NH to bus stop Avda Tolosa 5
- Wait 5 minutes for bus number 5
- Take the bus and travel 12 minutes until bus stop Buen Pastor Cathedral
- Walk 1 minute to Buen Pastor Cathedral

Apart from the functionalities presented on the previous sections, once the customization process is finished, a tourist can access the details of the whole route. Besides depicting all POIs on a map, the system provides full details for each visit. As an additional functionality, tourists can generate and download a pdf of the whole route. Finally, tourists can navigate the destination using a map based interface. We show all POIs on a map and tourists can choose to access details about any POI.

6.1 Architecture

We have designed and implemented a client server architecture. The client consists on a thick client executed on a Web browser. The server is constituted by a database (MySQL), and an application Server (Apache Tomcat). The client is a Web client based on Google Web Toolkit (GWT). GWT is a framework from Google allowing to develop a Web application on Java. GWT's compiler transforms the Java code on the corresponding HTML and JavaScript code compatible with main browsers (including iPhone's and Android's browsers).

We have followed the best practices promoted by Google [30], which can be summarized as the adoption of the following elements: dependency injection, Model-View-Presenter (MVP) pattern, bus event and browser history. On the next subsections we briefly introduce both backends of the prototype.

6.2 Desktop Client

The desktop client offers two main functionalities: generation of personalized tourist routes and their customization. Besides, it allows to navigate through the destination, to load previously customized routes, to update user's profile and to update the tourist score and visit duration.

Once tourists have chosen their profile and updated tourist score for POIs and visit duration (optional), they have to enter their restrictions. Then, the system generates a personalized tourist route. Visits are presented ordered in a map (Figure 2(a)). The list of visits is also shown on the sidebar. We assign a letter to each visit according to its order (A,B, C, ...).

Selecting a visit, more information about it and its POI (arrival time, visit duration, leave time) can be seen. Moreover, tourists can start customizing the route using five different buttons (Figure 2(b)). Four of the buttons represent a direction: Up, to move a visit towards the beginning of a route; Down, to move a visit towards the end of a route; Left, to move a visit to the previous day; and Right, to move a visit to the next day. And finally, the last one, the one with a minus character label, allows to remove the visit. Finally, a button with a plus character allows to add new visit. Tourists have to choose whether they want to add the new visit at a certain order, at the end of the day, or at the optimal order (the one minimizing total route time). Once the customization process is over, the tourists can access a detailed summary of the route and generate a downloadable pdf.

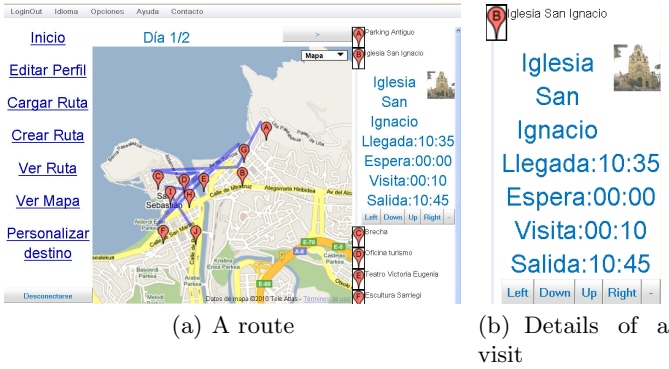


Fig. 2. Details of the desktop prototype

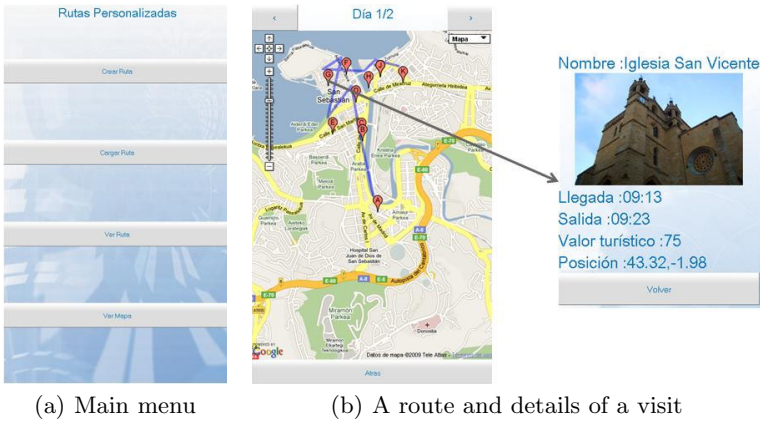


Fig. 3. Details of the mobile prototype

6.3 Mobile Client

The mobile client offers the same functionality, except for the customization. It has been taken out in order to improve the user friendliness of the prototype. The interface has been adapted to use with touch screens. Figure 3(a) shows the main screen with its button enlarged. Figure 3(b) shows a map with a personalized route and the details about one visit. Due to the reduced screen size, on the mobile client the details are shown on a new window.

7 Evaluation

We have evaluated the performance of our approach against test instances for the city of San Sebastian. Results are able to obtain routes in real time (worst case of 0.2 seconds for a 2 day 8 hour per day route) in a scenario with 50 POIs, 26 public transportation lines and 467 stops, and are comparable to the best

results available on the literature [20,27]. Moreover, most available examples are not able to generate routes in near real-time and the times they require to obtain a solution for comparable test instances are an order of magnitude higher.

We plan to test the tourist accuracy of the routes proposed by the prototype with real tourist for the city of San Sebastian during the summer of 2010.

8 Conclusions and Future Work

We have presented an intelligent routing system able to generate and customize personalized tourist routes in real-time and taking into account public transportation. Although the whole process is divided in three steps, we have focused only on the last two: route generation and route customization.

We have modeled the tourist planning problem, integrating public transportation, as the Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW). We have designed a heuristic able to solve it in real-time, precalculating the average travel times between each pair of POIs in a preprocessing step. Moreover, the system includes the basic functionalities required to customize the generated route. They allow to move, add and remove visits in a route.

Finally, we have shown a prototype which generates and customizes routes in real-time. This prototype has a desktop and a mobile frontend and has been successfully evaluated against test instances. Furthermore we plan to fulfil tests with real tourists during the summer of 2010 in the city of San Sebastian in order to evaluate the quality of the generated personalized tourist routes from the tourists' point of view.

Future works consists on extending the system to more cities with a different public transport network topology. The next one consists on integrating an advanced recommendation system in a wholly functional PET. Finally, we would like to insert social network capabilities, allowing to store, share and add travel experiences to better help tourists on the destination.

Acknowledgments

The authors would like to thank the Basque Government for partially funding this work through the neurebide and etourgune projects and the Centre for Industrial Management of the Katholieke Universiteit Leuven for hosting Ander Garcia as a guest researcher during 2009. Pieter Vansteenwegen is a post-doctoral research fellow of the "Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO)". Finally, authors would like to thank Javier Vallejo from "Compañía del Tranvia de San Sebastian" for providing real data about the public transportation network of the city.

References

1. Beer, T., Fuchs, M., Höpken, W., Werthner, H., Rasinger, J.: Caips: A context-aware information push service in tourism. In: Information and Communication Technologies in Tourism 2007, Ljubljana, Slovenia, pp. 129–149. Springer, New York (2007)

2. Schmidt-Belz, B., Laamanen, H., Poslad, S., Zipf, A.: Location-based mobile tourist services—first user experiences. In: *Information and Communication Technologies in Tourism 2003*, Ljubljana, Slovenia, pp. 115–123. Springer, Wien (2003)
3. Stroobants, R.: *Mobile tourist guides*. Technical report, Katholieke Universiteit Leuven, Belgium (2006)
4. Vansteenwegen, P.: *Planning in tourism and public transportation*. PhD thesis, Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium (2008)
5. Souffriau, W.: *Automated Tourist Decision Support*. PhD thesis, Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium (2010)
6. Kramer, R., Modsching, M., ten Hagen, K., Gretzel, U.: Behavioural impacts of mobile tour guides. In: *Information and Communication Technologies in Tourism 2007*, Ljubljana, Slovenia, pp. 109–118. Springer, Vienna (2007)
7. Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: A personal tourism navigation system to support traveling multiple destinations with time restrictions. In: *International Conference on Advanced Information Networking and Applications*, vol. 2, pp. 18–21 (2004)
8. Castillo, L., Armengol, E., Onaindía, E., Sebastián, L., González-Boticario, J., Rodríguez, A., Fernández, S., Arias, J., Borrajo, D.: Samap. an user-oriented adaptive system for planning tourist visits. *Expert Systems With Applications* 35(2), 1318–1332 (2008)
9. Zenker, B., Ludwig, B., Schrader, J.: Rose - assisting pedestrians to find preferred events and comfortable public transport connections. In: *ACM Mobility Conference 2009*. ACM, New York (2009) (accepted)
10. Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., Van Oudheusden, D.: A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence* 22(10), 964–985 (2008)
11. Garcia, A., Linaza, M., Arbelaitz, O., Vansteenwegen, P.: Intelligent routing system for a personalised electronic tourist guide. In: *Information and Communication Technologies in Tourism 2009*, Amsterdam, The Netherlands, pp. 185–197. Springer, Heidelberg (2009)
12. Tsiligrirides, T.: Heuristic methods applied to orienteering. *Journal of the Operational Research Society* 35(9), 797–809 (1984)
13. Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: a survey. *European Journal of Operational Research* (2009) (accepted)
14. Savelsbergh, M.W.P.: Local search in routing problems with time windows. *Annals of Operations Research* 4(1), 285–305 (1985)
15. Righini, G., Salani, M.: Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research* 36(4), 1191–1203 (2009)
16. Tricoire, F., Romauch, M., Doerner, K., Hartl, R.: Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research* 37(2), 351–367 (2010)
17. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* 36(12), 3281–3290 (2009)
18. Montemanni, R., Gambardella, L.: Ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences* 34 (2009)
19. Fomin, F., Lingas, A.: Approximation algorithms for time-dependent orienteering. *Information Processing Letters* 83(2), 57–62 (2002)

20. Garcia, A., Arbelaitz, O., Vansteenwegen, P., Souffriau, W., Linaza, M.: Hybrid approach for the public transportation time dependent orienteering problem with time windows. In: Corchado, E., Graña Romay, M., Manhaes Savio, A. (eds.) HAIS 2010. LNCS, vol. 6077, pp. 151–158. Springer, Heidelberg (2010)
21. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Efficient models for timetable information in public transportation systems. *Journal of Experimental Algorithmics* 12, 1–39 (2008)
22. Ding, B., Yu, J., Qin, L.: Finding time-dependent shortest paths over large graphs. In: EDBT 2008: Proceedings of the 11th international conference on Extending database technology, pp. 205–216. ACM, New York (2008)
23. Zografos, K.G., Androusoopoulos, K.N.: Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems* 9(1), 175–184 (2008)
24. Chiu, D.K.W., Lee, O.K.F., Leung, H.F., Au, E.W.K., Wong, M.C.W.: A multimodal agent based mobile route advisory system for public transport network. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005, p. 92.2 (2005)
25. Tumas, G., Ricci, F.: Personalized mobile city transport advisory system. In: Information and Communication Technologies in Tourism 2009, Amsterdam, The Netherlands, pp. 173–183. Springer, Wien (2009)
26. Kabassi, K.: Personalizing recommendations for tourists. *Telematics and Informatics* 27(1), 51–66 (2010)
27. Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaza, M.: Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, Special issue: Transport Scheduling (2010) (under review)
28. Delling, P.: Engineering and Augmenting Route Planning Algorithms. PhD thesis, Fakultät für Informatik Universität Fridericiana zu Karlsruhe (2009)
29. Lourenço, H.R., Martin, O., Stutzle, T.: Iterated local search. In: Kochenberger, F.G., Glover, G. (eds.) *Handbook of Metaheuristics*, pp. 321–353. Kluwer Academic Publishers, Dordrecht (2003)
30. Ryan, R.: Google web toolkit architecture: Best practices for architecting your gwt app. In: Google IO Developer Conference 2009 (2009)

Automated Generation of Itineraries in Recommender Systems for Tourism

Pierpaolo Di Bitonto, Francesco Di Tria, Maria Laterza, Teresa Roselli,
Veronica Rossano, and Filippo Tangorra

Dipartimento di Informatica, Università degli studi di Bari "Aldo Moro",
Via Orabona 4, 70126 Bari, Italy
{dibitonto, francescoditria, marialaterza, roselli,
rossano, tangorra}@di.uniba.it

Abstract. Current recommender systems can support tourists in choosing travel products (accommodation, activities, means of transport, etc.), in planning long trips, and in profitably spending time in a specific geographical area such as a region (or a city). In the last case, the system should be able to construct itineraries suited to the tourist's interests. In this paper, a method for generating tourist itineraries in knowledge-based recommender systems is proposed. The method is based on a theoretical model that defines space-time relations among items of intangible cultural heritage (called events) and on transitive closure computation (of the relations), that is able to construct chains of events. The proposed method has been implemented in the T-Path recommender system, that suggests itineraries of cultural events occurring in the Apulia region.

Keywords: recommender systems, transitive closure, logical programming.

1 Introduction

Recommender systems are very useful tools for the tourism industry because they can support users in organizing long, medium and short trips, suggesting locations, accommodation, transport, and so on. For example, a tourist staying in the Apulia region could be interested in spending time visiting the main cities, landscapes or cultural events (festivals, processions, special markets, etc.). But planning itineraries is a complex problem and very difficult to face in an unknown geographical area. Thus, the tourist usually relies on maps, or guide books that suggest the main locations or attractions worth visiting. The improvements of technology are now offering the tourist more and more effective tools that are able not only to propose locations and/or attractions to visit, but also allow the tourist to select the most suitable itineraries for her/his interests. The research efforts in this domain have yielded numerous recommender systems aiming to support the tourist in making the best choices.

Most of the recommender systems presented in literature are able to suggest single items (flights, buildings, cities, etc.) on the basis of the tourist's requests. At this point, the tourist should create her/his own itinerary, choosing the items that fit her/his requirements. The newest recommender systems build itineraries automatically using

the Constraints Satisfaction Problem (CSP) approach [1]. This means that they generate a sequence of attractions or spots to be visited, filtering data according to the user's constraints (day of visit, cost, and so on) specified in the request.

Unlike the above approach, this paper proposes a method that builds tourist itineraries using transitive closure computation [2]. In particular, these itineraries are focused on intangible cultural heritage (called events) occurring in a geographical area. In other words, the main idea of our method is to detect a set of events (processions, festivals, special markets, etc.) with space and time relationships, in the sense that they are near both in space and in time.

Transitive closure computation aims at answering the following recursive query: "given an event e , what are all the possible paths starting from e ?". This computation is not done via Relational Calculus but via Predicate Calculus, as Aho and Ullman state in [3]: Relational Calculus does not suffice to express this kind of recursive query. Therefore, the itinerary planning process is generated by a logical program that uses a knowledge base. The proposed method has been implemented in the T-Path prototype, a knowledge-based recommender system that suggests tourist itineraries in the Apulia region.

The main contributions of our research work can be summarised as follows. Firstly, a novel method for generating itineraries using transitive closure computation is proposed. Secondly, the method is scalable. It can be used to generate itineraries at different grain sizes: the events can occur in regions, districts or cities (space relation), or in weeks, days, or hours (time relation). Because of this, the method could be implemented in any knowledge-based recommender system that uses an ontological representation of events and a logical program that computes transitive closure. Finally, the application domain is also a novelty in tourism recommender systems. In fact, these systems usually suggest items such as accommodation, flights, museums, etc. In our work the items suggested are itineraries composed of an ordered list of correlated events.

The paper is organised as follows: section 2 presents some related works about tourism recommender systems; section 3 presents the method for constructing tourist itineraries; section 4 gives an overview of the T-Path system architecture; section 5 shows a specific case study. Finally, some conclusions and future research directions are outlined.

2 Related Works

Current recommender systems for the tourism domain can be classified in two main groups: systems that suggest single items (or travel products) and systems that suggest a set of items. Examples of the first group are Triplehop's TripMatcher (www.ski-europe.com) and VacationCoach (www.travelocity.com) [4]. They select travel destinations by filtering the items collected in a virtual catalogue according to the user's needs and preferences. The systems that suggest a set of items (the second group) support tourists in planning their trips, arranging transportation, accommodation, and everything else needed. Examples of these kind of systems are Expedia (www.expedia.com), DieToRecs [5], and Trip@dvice [6]. In these cases the recommendation process works like a virtual travel agency: it suggests a pre-packaged offer,

or else it guides the user in the selection of the travel components (flight, accommodation, car, etc.) according to the user's needs and preferences.

Such recommender systems support tourists in building their own itineraries; systems that can automatically build itineraries starting from the tourist's requests are still few. Some authors state that this is "due to the complexity of the task and doubts concerning scalability to handle large travel databases" [7]. The construction of itineraries needs to generate a sequence of items to be visited. In literature, the problem of generating itineraries is dealt with by filtering data according to the user's constraints (day of visit, cost, and so on) specified in the request. The Electronic Travel Planner (ETP) prototype [7], for example, stores information (such as duration, cost, and availability, dates and times) about travel products in a relational database. The problem of building itineraries is expressed in terms of constraints (day of visit, arrival/departure time and location, preferred time of visit, cost, etc.) and an objective function. The itinerary suggested to the tourist satisfies all the constraints and maximizes the value of the objective function. The value of the objective function is the difference between the "enjoyment value" of the itinerary and its cost.

Other examples of systems that use the Constraints Satisfaction Problem (CSP) approach are: *INTRIGUE* [8] and Smart Tourist Agenda Recommender (STAR) [9], recommender systems that suggest itineraries around the city of Turin (in north Italy), and the tourist guide in [10] that support users visiting the city of Salamanca (Spain). Conversely, in [11] the problem of generating itineraries in the Macau system (that extends the previous MacauMap system [12]) is solved by using the A* graph search algorithm [13], using travel time as the distance value between tourist spots.

The method proposed in this paper and its application in a recommender system allows the tourist to receive a group of itineraries made up of a set of correlated events (procession, festival, special market, etc.). The chains of events are generated by means of transitive closure computation of the space-time relationships.

3 A Method for Generating Tourist Itineraries

In order to characterize the events considered (procession, festival, special market, etc.) and to determine whether two events have a space-time relationship, a set of functions and a space-time relation were defined as follows. Let E be a finite set of events e occurring in a geographical area (for example in the Apulia region). Details of defined functions are given below.

Let \mathcal{N} be the set of natural numbers, and L the finite set of all names of existing locations (at a specific grain size: district, city, region) in the geographical area considered. The functions *Start*, *End*, *Location*, and *Distance* were defined as follows:

- *Start*: $E \rightarrow \mathcal{N}$ is the function that associates to each event $e \in E$ a natural number that expresses the date when the event starts (for example the 1st February 2010 is the 32nd day of the year).
- *End*: $E \rightarrow \mathcal{N}$ is the function that associates to each event $e \in E$ a natural number that expresses the date when the event ends.
- *Location*: $E \rightarrow L$ is the function that associates to each event $e \in E$ the name of the location where the event occurs.

- *Distance*: $L \times L \rightarrow \mathcal{N}$ associates to each couple of locations the distance between them. If the distance between the two places of the couple is equal to 0, then the two places are coincident. If the distance between two places is lower than a threshold value ε (conveniently fixed), then the places are near in space, otherwise the places are distant.

Given the above-described functions, it is possible to define the space-time relation Δ between two events as follows: $\forall e_i, e_j \in E, e_i \Delta e_j \text{ iff } e_i \Delta_t e_j \wedge e_i \Delta_s e_j$, where:

- $i, j \in \mathcal{N}$
- $e_i \Delta_t e_j \Leftrightarrow \text{Start}(e_i) \leq \text{Start}(e_j) \leq \text{End}(e_i)$
- $e_i \Delta_s e_j \Leftrightarrow \text{Distance}(n_i, n_j) \leq \varepsilon$, whereas $\varepsilon \in \mathcal{N}$ is a fixed threshold, $\text{Location}(e_i) = n_i$, $\text{Location}(e_j) = n_j$.

Given a set of events, on the basis of this relation it is possible to build paths like e_1, e_2, \dots, e_n where e_i is correlated in space and time to e_{i+1} , for $i = 1, \dots, n-1$. For example, starting from event e_i , we need to know whether there is an event e_{i+1} such that e_i and e_{i+1} have a space-time relationship (represented by the symbol “ \rightarrow ”). If such an e_{i+1} exists, in a recursive way, we look for another event e_{i+2} , and so forth. When no other event satisfies the Δ relation, the transitive closure computation ends. The result is a path composed of a chain of events that have a pairwise space-time relationship ($e_i \rightarrow e_{i+1} \rightarrow e_{i+2} \rightarrow \dots \rightarrow e_n$). A deeper and more detailed discussion of the proposed method is reported in [14].

4 T-Path Architecture

The described method was implemented in the T-Path prototype, a knowledge-based recommender system that proposes tourist itineraries to see events occurring in different cities in the Apulia region. In the T-Path architecture it is possible to distinguish three logical layers (Fig. 1): (a) the data layer, that manages the data sources and builds an integrated knowledge base; (b) the application layer, that generates itineraries by transitive closure computation; and (c) the presentation layer, that displays the results of this computation, in other words the itineraries.

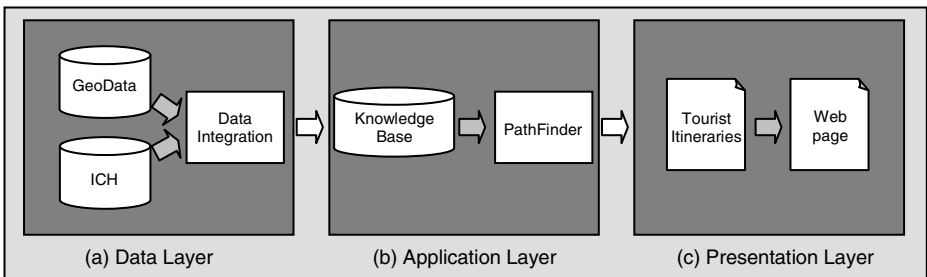


Fig. 1. T-Path architecture

4.1 Data Layer

The T-Path Data Layer is made up of two relational databases (Fig. 1 - a), namely GeoData and ICH (Intangible Cultural Heritage). The GeoData database stores information about the geographical structure of the considered territory. This database contains, for example, information such as “Bari is a city”, “Bari is located in Apulia”, and “Apulia is an Italian region”. Moreover, for each city the geographical coordinates (such as latitude and longitude) are also stored. In this way, it is possible to know the distance in kilometers between two cities. The ICH database stores data about cultural events (such as fairs, religious events, and musical concerts). Each event is classified in a predefined category and is described by a set of attributes. The attributes are used to define the space-time collocation of the events. Therefore, for each event, the name, the location, and the date are provided. For example, this database contains information such as: name of the event “Santa Lucia fairs”; location “Lecce (LE)”, and date: “from December 13 to December 24”. The geographical data and the ICH data are extracted from the source databases and loaded into a knowledge base through an integration process. The Data Integration module is a software component that dynamically generates the facts of the knowledge base, by mapping relational schemas to predicates. According to our mapping strategy, the integration process generates a fact in the knowledge base for each tuple in a relation.

4.2 Application Layer

The T-Path Application Layer is made up of the knowledge base and the PathFinder program (Fig. 1 - b). The knowledge base (see Table 1) contains the facts expressed using the Predicate Calculus. The predicates, derived from the functions of the method defined in section 3, are divided into two groups: the first group (first five predicates) describes the geographical knowledge; the second group (last two predicates) defines the space and time features of an event. The two groups of predicates are related to each other because the location of an event is a city.

The next examples aim to clarify the predicate instantiation.

Table 1. Knowledge base predicates

Predicate	Semantics
city(X, Y)	X is a city in province of Y.
province(X, Y)	X is a province of Y, where Y is a region.
region(X, Y)	X is a region of Y, where Y is a nation.
nation(X)	X is a nation, where X = 'Italy'.
near(X, Y)	The city X is near to the city Y.
location(e, Y)	The location where the event e happens is Y, where Y is a city.
day(e, G, A, Z)	The event e happens on the day G of the year A. Z is the duration of the event, in terms of days.

Example 1. The following facts state that Italy is a nation, Apulia is an Italian region, Bari and Lecce are Apulian provinces, and Bari is close to Lecce.

```
nation(Italia).
region(Apulia, Italia).
province(Bari, Apulia).
province(Lecce, Apulia).
city(Bari, Bari).
city(Lecce, Lecce).
near(Bari, Lecce).
```

Example 2. The following facts state that the festivity of San Nicola (Saint Nicholas) occurs recurrently in Bari on the 8th of May (that is the 128th day of 2009), and it lasts 3 days. The *location* predicate recalls the *Location* function, whereas the grain size is the city level of the geographical hierarchy. The *day* predicate recalls the *Start* and *End* functions. Thanks to these functions, both the starting date and the duration of an event are defined.

```
location('festivity of Saint Nicholas', Bari)
day('festivity of Saint Nicholas', 128, 2009, 3).
```

The PathFinder program builds a path (if it exists) via transitive closure computation using event $e \in E$ as an initial event. Then the PathFinder writes the result of the computation in an XML file that will be displayed to the tourist by the presentation layer. The pseudo-code of the program is shown below.

```
createXML(T)
for each  $e \in E$ 
     $P_e = tc(e)$ 
    PathXML = generatePath( $P_e$ )
    add(PathXML, T)
end for
storeXML(T)
```

The core of the PathFinder is a logical program able to execute recursive queries and to compute the transitive closure. The logical program has been created using the Prolog programming language [15] and the Swi-Prolog development environment [16]. The logical program provides an answer to the following recursive query: “*what are all the possible paths starting from a specific event e ?*”. In the logical program (shown below) $tc(e)$ is the goal and e is the starting event. The inferring rules recall the space-time relation of the method.

```
tc(e):-          location(A, X), rel(A, X).
rel(A, X):-      location(B, Y), spatial(X, Y),
                 temporal(A, B), print(A, B),
                 retract(location(A, X)), rel(B, Y).
spatial(X, Y):-  near(X, Y); near(Y, X).
temporal(A, B):- day(A, Sa, Aa, Da),
                 day(B, Sb, Ab, Db), A \= B, Aa == Ab,
                 Dura is Sa + Da, Sb >= Sa, Sb <= Dura.
print(X, J):-    write(X), write('•'), writeln(J).
```

The output produced (the chain of events) by the program is stored in an XML file, defined by the following Document Type Definition (DTD).

```

<!ELEMENT output (path*)>
<!ELEMENT path (stage+)>
<!ELEMENT event (name, location, date, brochure)>
<!ELEMENT location (city, province, region)>
<!ELEMENT event (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT province (#PCDATA)>
<!ELEMENT region (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT brochure (#PCDATA)>
<!ATTLIST stage id CDATA>
    
```

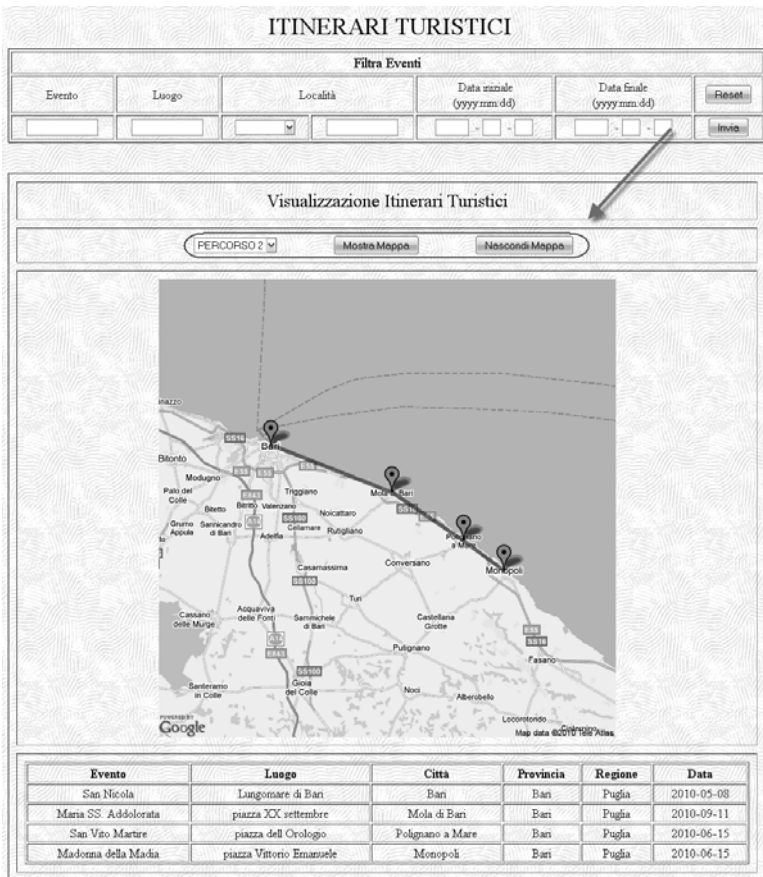


Fig. 2. A tourist itinerary computed by the Pathfinder

4.3 Presentation Layer

The T-Path Presentation Layer (Fig. 1 - c) aims at displaying the content of the XML file produced by the PathFinder. The information on the itineraries is visualized using both a geographical map, thanks to the integration of the Google Map [17] in the system, and a table that reports all the details of the events (Fig. 2). Moreover, it is possible to navigate the XML file in order to filter the contents on the basis of the tourist search criteria.

5 T-Path at Work

In T-Path, the most important religious events in the Apulia region occurring between the end of 2009 and the first months of 2010 have been collected. Apulia is a region in the South of Italy, being the easternmost region in our Nation. This region is located along the Adriatic Sea and down to the Ionian Sea. Its provinces are Bari, which is the chief town, Lecce, Brindisi, Foggia, and Taranto.

As examples, some of the events stored in the T-Path prototype are reported below (in Italian):

1. Fiera di Santa Lucia, from December 13 2009 to December 24 2009 in Lecce (LE).
2. Presepe vivente di Sanarica, from December 24 2009 to January 6 2010 in Sanarica (LE).
3. Presepe artistico, from December 24 2009 to January 11 2010 in Muro Leccese (LE).
4. Grande presepe artistico, from December 1 2009 to February 2 2010 in Salice Salentino (LE).
5. Focura per Sant'Antonio abate, January 16 2010 in Novoli (LE).
6. Presepe vivente di Bethlehem, from December 25 2009 to January 6 2010 in Grumo Appula (BA).
7. La notte dei Magi, January 5 2010 in Valenzano (BA).
8. Presepe Palazzo Marchesale, from December 8 2009 to January 6 2010 in Adelfia (BA).
9. Calata dei re Magi, January 6 2010 in Lizzano (TA).
10. Festa di San Ciro, from January 1 2010 to January 31 2010 in Grottaglie (TA).

For each city the latitude and longitude were also stored to know the distance (expressed in kilometres) between the cities. We fixed the ε threshold distance at 50 km. Let n_1 and n_2 be two cities, the criterion for the space relationship is

$$Distance(n_1, n_2) < 50, \text{ that is, } n_1 \text{ is near } n_2 \text{ iff } Distance(n_1, n_2) < 50.$$

As an example:

- $Distance(\text{Lecce}, \text{Sanarica}) = 32.78 < 50 \Rightarrow \text{Lecce is near to Sanarica}$
- $Distance(\text{Lecce}, \text{Adelfia}) = 131.34 > 50 \Rightarrow \text{Lecce is distant from Adelfia.}$

Of course, the *Distance* function is symmetric, i.e. $Distance(n_i, n_j) = Distance(n_j, n_i)$, for $i \neq j$, and reflexive, as $Distance(n_i, n_i) = 0$, $\forall i \in \mathfrak{N}$. Starting from these data the transitive closure computation will be computed. For the sake of simplicity, in this case study we refer to a specific event using the *code* field. Supposing that 8 is the start event, T-Path searches whether another event e exists such that $8 \rightarrow e$. Since:

- $Start(8) =$ December 8, 2009,
- $End(8) =$ January 6, 2010,
- $Start(6) =$ December 25, 2009,
- $End(6) =$ January 6, 2010,
- $Location(8) =$ Adelfia,
- $Location(6) =$ Grumo Appula,
- $Start(8) \leq Start(6) \leq End(8)$,
- $Distance(8, 6) = 13.56 \Rightarrow$ Adelfia is *near* Grumo Appula,

the T-Path system concludes that event 8 has a space-time relationship with 6, so it is possible to build the path $8 \rightarrow 6$.

In a recursive way, the T-Path system searches for another event f such that $6 \rightarrow f$:

- $Start(6) =$ December 25, 2009,
- $End(6) =$ January 6, 2010,
- $Start(7) =$ January 5, 2010,
- $End(7) =$ January 5, 2010,
- $Location(6) =$ Grumo Appula,
- $Location(7) =$ Valenzano,
- $Start(6) \leq Start(7) \leq End(6)$,
- $Distance(6, 7) = 15.03 \Rightarrow$ Grumo Appula is *near* Valenzano,

The T-Path system concludes that events 6 and 7 have a space-time relationship and builds the path $6 \rightarrow 7$. Since no other event has a space-time relationship with 7, the computation ends. The final result of this recursive query is path $A = \{8 \rightarrow 6 \rightarrow 7\}$. In the same way, T-Path finds the following further paths:

- $B = \{4 \rightarrow 5\}$,
- $C = \{4 \rightarrow 10 \rightarrow 9\}$,
- $D = \{4 \rightarrow 1 \rightarrow 2 \rightarrow 3\}$,
- $E = \{4 \rightarrow 1 \rightarrow 3 \rightarrow 2\}$.

where A, B, C, D and E are independent paths and each path is formed by an ordered set of events. The XML file of the path B is reported below.

```
<output>
  ...
  <path>
    <event id = 4>
      <name>Grande presepe artistico</name>
      <location>
        <city>Salice Salentino</city>
        <province>Lecce</province>
```

```

        <region>Apulia</region>
    </location>
    <date> December 1, 2009</date>
</event>
</path>
<path>
    <event id = 5>
        <name> Focura per Sant'Antonio abate
    </name>
        <location>
            <city>Novoli</city>
            <province>Lecce</province>
            <region>Apulia</region>
        </location>
        <date> January 16, 2010</date>
    </event>
</path>
    ...
</output>

```

The XML tree built is then displayed to the tourist using both a map where the path is depicted and a table showing all the details of the chain of events (Fig. 2). Moreover, the tourist can view the details of all the itineraries computed using a select box.

6 Conclusions

Up to now, the problem of automatically building itineraries to be proposed by recommender systems has been faced using the Constraints Satisfaction Problem (CSP) approach. The output is a sequence of attractions or spots to be visited, filtered according to the tourist's constraints (day of visit, cost, and so on) specified in the request.

This paper presents a method for building itineraries incorporated in a knowledge-based recommender system. The proposed method aims to find a chain of events using transitive closure computation on the basis of the space-time relations. This is possible thanks to the use of the theoretical model, that consists of a set of functions that characterize each single event, and a space-time relation that allows the events to be correlated. The theoretical model can be easily modified so as to correlate events at different grain sizes, both in terms of time and space. Moreover, it can be incorporated in any knowledge-based recommender system that uses an ontological representation of events and has a logical program able to execute recursive queries. The paper illustrates an implementation of the proposed method in T-Path, a recommender system prototype that suggests itineraries of cultural events (festivals, processions, special markets, etc.) in the Apulia region. Future developments will allow the chain of events to be tailored to the tourist's profile. For example, if the tourist is a religious man, the system will attribute high priority to the presentation of the whole chain of religious events (such as processions, historical parades, etc.).

References

1. Tsang, E.: *Foundations of Constraint Satisfaction*. Academic Press, London (1993)
2. Elmasri, R., Navathe, S.B.: *Fundamentals of Database Systems*, 5th edn. Addison-Wesley, Reading (2006)
3. Aho, V., Ullman, J.D.: Universality of data retrieval languages. In: 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp. 110–119. ACM Press, New York (1979)
4. Ricci, F.: Travel recommender systems. *J. IEEE Intelligent Systems* 17, 55–57 (2002)
5. Ricci, F., Fesenmaier, D.R., Mirzadeh, N., Rumetshofer, H., Schaumlechner, E., Venturini, A., Wöber, K.W., Zins, A.H.: DieToRecs: a case-based travel advisory system. In: *Destination Recommendation Systems: Behavioral Foundations and Applications*. CABI Publisher International, Wallingford (2006)
6. Venturini, A., Ricci, F.: Applying trip@advice recommendation technology. In: 17th European Conference on Artificial Intelligence, pp. 607–611. IOS Press, Amsterdam (2006), <http://www.visiteurope.com>
7. Dunstall, S., Horn, M., Kilby, P., Krishnamoorthy, M., Owens, B., Sier, D., Thiebaut, S.: An automated itinerary planning system for holiday travel. *J. Information Technology & Tourism* 6, 1–33 (2004)
8. Petrone, G., Ardissono, L., Goy, A.: INTRIGUE: personalized recommendation of tourist attractions for desktop and handset devices. *J. Applied Artificial Intelligence* 17, 687–714 (2003)
9. Goy, A., Magro, D.: Dynamic Configuration of a Personalized Tourist Agenda. In: *IADIS International Conference WWW/Internet 2004*, pp. 619–626. IADIS, Madrid (2004)
10. Corchado, J.M., Pavón, J., Corchado, E.S., Castillo, L.F.: Development of CBR-BDI agents: A tourist guide application. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS (LNAI), vol. 3155, pp. 547–559. Springer, Heidelberg (2004)
11. Biuk-Aghai, R.P., Fong, S., Si, Y.-W.: Design of a Recommender System for Mobile Tourism Multimedia Selection. In: *2nd International Conference on Internet Multimedia Services Architecture and Application*, pp. 1–6. IEEE Press, Los Alamitos (2008)
12. Biuk-Aghai, R.P.: MacauMap: Next generation mobile travelling assistant. In: *Proceedings of Map Asia* (2004)
13. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *J. IEEE Transactions on Systems Science and Cybernetics* 4, 100–107 (1968)
14. Di Bitonto, P., Di Tria, F., Laterza, M., Roselli, T., Rossano, V., Tangorra, F.: A model for generating tourist itineraries. In: *4th International Workshop on Semantics in Data and Knowledge Bases* (2010) (submitted)
15. Sterling, L., Shapiro, E.: *The Art of Prolog: Advanced Programming Techniques*, 2nd edn. MIT Press, Cambridge (1994)
16. SWI-Prolog, <http://www.swi-prolog.org>
17. Google Map, <http://maps.google.it>

A Method for Assessing Website Communicative Efficacy Using a Semantic Annotation Tool

Nadzeja Kiyavitskaya¹, Nicola Zeni², Cristina Coulleri², and Luisa Mich²

¹ Dept. of Information Engineering and Computer Science, University of Trento, Italy
nadzeja@disi.unitn.it

² Dept. of Computer and Management Sciences, University of Trento, Italy
{nicola.zeni,luisa.mich,crisrina.coulleri}@unitn.it

Abstract. Evaluation of the communicative efficacy of a website is important in the context of an organization communication strategy. To address this issue, we propose a systematic method and a tool that supports it. The tool is based on a general-purpose semantic annotation framework. The application and feasibility of the method are introduced through a case study where it was used to compare the communicative efficacy of a set of tourist destination websites.

Keywords: communicative efficacy, website content, communication strategy, tool support, Cerno.

1 Introduction

Communication has been investigated in different disciplines: rhetoric, psychology, mathematics, sociology, anthropology, and many others. Information theory [19] and communication theory as specific disciplines have drawn a widespread interest since the inception of new communication channels and devices. Thus, we have a large variety of areas and contributions related to communication principles and theories. In particular, computer science has been investigating communication issues in the context of such disciplines as artificial intelligence, natural language processing, human computer interface, knowledge management, addressing also human-side aspects apart from those considered first, that included, for example, coding. Among them, an important role is given to the *communication efficacy* (CE) that is the capacity of communicating to accomplish a desired goal. CE has been investigated, for instance, for communication in the context of collaborative interaction tools, where “the CE of the environment is determined based on how quickly participants are able to accomplish the task at hand” [5]. Other studies focused on the analysis of the pragmatics of communication [10, 28], more specifically, on the effect of messages in terms of actions following an utterance or linguistic act [1]. Websites are eminently communicative environments, and their content is critical to fulfill business goals of a company and for its communication strategy [24].

In order to identify a systematic way to evaluate the CE of a website, we first start from the fundamental principles of communication. In the classical rhetoric of Aristotle a discourse can be given to “docere, delectare, flectere”, that is to instruct the audience, gain their goodwill, and arouse their emotions [7]. The modern communication

theory identifies four fundamental objectives: *control, motivation, information, emotional expression* [18, 21]. Websites play a relevant role to support all these objectives, which are strongly related to on-line corporate communication, and in particular to marketing communication strategy [24]. Though there are a variety of studies and models for website quality (see, among the others, [13, 15, 25]), the evaluation process based on these models is not straightforward. Moreover, little methodological and tool support was developed so far in addressing the specific issue of CE. Also, from the business point of view, web analytics are used to measure the success of a web site, while our goal is to address CE as a design method, as an ex-ante issue. In this respect, CE requires as necessary conditions that: (a) the website functions correctly and (b) the website contains all the necessary information to support the goals related to the communication strategy. Our work focuses the second aspect of the CE, i.e., we develop a method to ensure that the website contains all the necessary information to accomplish the business model's goals in the context of the corporate communication. We assume the fundamental hypothesis of semiotics [2] which distinguishes three levels of the message analysis: syntax, semantics, and pragmatics; where the third level is a function of the others. In fact, in the first instance, the effect of a message depends on its correctness (syntax) and the content informativeness (semantics). To this end, the task that we address in this work is measuring the content informativeness dimension. In order to do this, we suggest adopting semantic annotation (SA) tools developed to identify relevant categories in text, video or audio content. We elaborate a systematic approach for evaluating the CE based on a SA framework called Cerno [9]. For a preliminary validation of the method we considered a set of 13 websites of local Tourist Boards in the Italian province of Trentino: Aziende di Promozione Turistica di ambito (A.P.T.). Some results of this study were published in [8], where we illustrated Cerno and its performances in terms of recall and precision. This work introduces the CE evaluation method, which steps have been defined based on the experience matured since then.

The paper is structured as follows: section 2 describes our approach of evaluating the CE of websites; section 3 illustrates the application of the proposed method on a case study analyzing the CE of tourist destination websites; and the conclusions of are drawn in section 4.

2 Our Approach

Our starting assumption for evaluating the CE of a website is that the efficacy of communication depends on the content informativeness, and in particular on the breadth of the knowledge coverage. This characteristic implies the need in identification of the pieces of information pertinent to the website goals in its content. This task can be addressed by SA of document text, which explicitly assigns semantics to text fragments relevant to a particular annotation goal.

Based on these assumptions, we suggest an evaluation process that consists of the following steps:

1. Analyze the business model of the company by identifying its mission and its management strategies.
2. Identify the role of the website in the business model.

3. Based on the identified objectives of the website, understand how the website should support this role in terms of its content.
4. Derive the conceptual model for SA using the list of content categories and concepts.
5. Fulfill the SA in order to find all content fragments related to the entities of the conceptual model.
6. Use the results of the SA to calculate some indexes to evaluate the CE of the websites.

The following subsections elaborate each step of the process.

1. *Identification of the business model.* The CE can only be estimated within a specific communication environment, because it requires understanding the goals of the communication. In other words, our first concern must be to identify the mission that an organization is trying to achieve and its business model.
2. *Identification of the website role in the business model.* Having derived the organization's strategies and business model, we need to understand the role of the website within this model, i.e., which goals is the website supposed to contribute to.
3. *Identification of relevant content categories.* To realize the goals derived from the business model of the organization, the website has to provide appropriate content, making the user take an action that the organization is interested in. Thus, our next step is to analyze the goals of the website and identify a list of key categories that must be present on it to contribute to its CE.
4. *Derivation of the conceptual model.* When the key categories are provided, the next step is to specify them by eliciting more specific concepts and defining a domain conceptual model. For example, the concept of Geography is a generic category and can be extended with a lot of related terms based a dictionary or a thesaurus like WordNet [29]. In order to limit the scope of related terms, we suggest asking the domain experts to filter candidate concepts by selecting only those relevant to the website strategy.
5. *Semantic annotation based on the derived model.* The conceptual model built on the previous step should guide the SA process. There are many SA approaches and tools available (a detailed classification of such tools can be found in [20]). However, some tools cannot be applied for the evaluation of websites. For example, machine-learning methods require essential amount of training data in order to provide reliable results, whereas it is not feasible to expect much training data provided from a single site. Wrapper induction methods are not applicable since they rely on the regularity of HTML format, on which the annotation rules are based. Therefore, we suggest using one of the rule-based approaches. In the perspective of this work, i.e., tourist destination websites, the domain knowledge corresponds to the business strategies materialized in the website. Starting from these strategies, a domain conceptual model must be derived. One of such model-driven annotation tools based on a comprehensive SA framework is Cerno [9]. In Cerno, the conceptual model obtained from domain experts must be then converted into an *annotation schema*, i.e., the list of semantic categories with their annotation rules. After that, the annotation process can be executed.

6. *Analysis of the annotation results based on the efficacy measures.* The results of the SA can be used as an input for calculation of quantitative measures of CE. The most basic index is to calculate the total number of annotations identified for each category and check which categories are well-covered and which are under-represented. This index can be then elaborated into more complicated metrics [3]:

- *The evaluation according to a hierarchy inside of the conceptual model.* Every concept can be assigned a weight (w) based on its hierarchical level of the conceptual model. The idea is to assign smaller weights to the deeper levels, while weight higher top level concepts. The final result is the sum of the number of annotations (Na_i) for each i -th concept multiplied by its corresponding weight.
- *The evaluation based on the website structure hierarchy.* This metric takes into account the hierarchy structure of the website. For example, the information found on its Home page can be given a greater weight compared to the information that is provided on the internal pages, assigning a weight according to the hierarchy level of the annotations.
- *The evaluation based on the number of keywords.* Instead of using a SA tool to provide the quantitative data on the content coverage, one can use a list of keywords related to the concepts of interest.
- *The evaluation based on the relevance of HTML tags.* When fulfilling the SA, one can take into account the information about HTML tags where the annotations were found. For instance, the heading like `<h1>`, `<title>`, can be given a higher score than the text found in `<p>` tags.
- *The evaluation based on the user profile.* This measure combines two things: the information on the profile of the user as well as the information on the needs of the user. One solution to obtain the profile data is to use socio-demographic reports, as for instance in [23].

Formulas for the first four metrics are represented by the sum of annotations which are multiplied by weights assigned according to the suggested evaluation aspect, i.e., position in the hierarchy of the conceptual model, position in the hierarchy of the website, relevance of the keywords, or relevance of the HTML tags:

$$CE = \sum_{i=1}^n \sum_{j=1}^m Na_{ij} w_j$$

To express the last metric we can use the following function:

$$CE = f(\text{user profile}, \text{user needs})$$

where a user profile could be for example defined in the set {student, manager, teacher, employed, unemployed, elderly, family}; user needs could be defined as Accommodation = {hotel, camping, B&B, residence, apartment}. In this case, weight distribution depends on the profile and needs of the users.

All these measures require a preliminary selection of weights. This selection must be realized by domain experts together with web engineers and is not trivial.

3 Case Study of Alpine Tourist Destination Websites

This case study was run within a larger project of the eTourism group of University of Trento [6] which aim was to investigate the tourism models of Trentino (<http://www.provincia.tn.it/>) as an alpine destination. The economy of Trentino – which is part of the Italian autonomous region Trentino Alto-Adige Suedtirol – is mainly based on tourism, and the territory is organized in fifteen agencies responsible for local administration and promotion [11]. We choose the Local Tourist Boards in Trentino because they are highly uniform as regards their mission and goals unlike agencies of promotion at higher level (national and regional). Moreover, we found a reasonable number of them, i.e. 15 (this number refers to the situation until year 2008), which allows carrying out an accurate analysis of the sites. The following subsections illustrate application of the CE evaluation method.

1. *Business model identification.* Tourism is a principal source of income in many areas of the Trentino Dolomites, and therefore, the mission of the Tourist Boards in this area is marketing and promotion of local tourist destinations. In particular, the eTourism group was interested in evaluating the efficacy of the Tourist Boards in promoting these alpine destinations among international visitors. To address this task, we need to assess the quality of the communication realized by the Tourist Boards to accomplish this goal.
2. *Identification of the website role in the business model.* The communication for promoting the alpine destinations is realized through the official websites of 15 Tourist Boards. Considering the international target of tourism offer, one of the main requirements of our investigation was the presence of English versions of the websites. Only 13 websites satisfy such a requirement. Thus, we have to extract the English-language content of the websites and analyze how comprehensively the strategic goals of the Tourist Boards are represented there.
3. *Identification of relevant content categories.* In order to derive a list of key categories that must be covered by a Tourist Board website to be effective, we referred to the notion of *tourist destination*, which is defined as a place of travel that tourists wish to visit due to its natural or artificial attractions. A tourist destination is characterized by the following factors [13]:
 - a well-defined geographic area with specific borders and a territorial identity;
 - presence of a variety of operators with different visions and objectives which requires a shared strategy in promoting a local offer;
 - understanding of the nature of a potential demand for the tourist products offered;
 - awareness of the need to balance tourist use of local resources according to ecological, environmental and community regulations.

Among the operators involved in generation of the tourist product are: Hosting structures, Restaurants, Commerce, Handicrafts, Agriculture, Entertainment, Cultural institutions, Sports activities, Free-time, Public boards, Transportation, Public services.

All these factors and operators embrace a list of key categories important for promoting a tourist destination. The nature of the destination is another issue to take into account. There exist many various types of destinations, and concepts relevant for each type differ or need to be defined at different level of details [14]. In the context of our case study, alpine destinations are characterized by specific sport activities (skiing, snowboarding, hiking, mountain biking) and lodging facilities (hotel, B&B, garni, gasthaus) that visitors are interested in. Whereas for a religious destination, for instance, sports do not play an important role and an accommodation could be arranged in a convent. Therefore, in the next step we elaborate the key categories based on our destination type.

4. *Derivation of the conceptual model.* In order to develop a conceptual model for CE of an alpine destination, we asked the tourist experts to select a subset of high-level categories and more specific concepts relevant for its strategies. The resulting list includes the following concepts: accommodation (hotels, campings, apartments for rent), catering places (restaurants, local food), sports (possibility to do various sport activities, competitions, courses, facilities), transportation (how to reach a destination by any transportation means, timetables, terminals), culture and history (artistic heritage, places to visit, cultural events, local traditions, holidays, costumes), and medicine (medical services and treatments of the resort). See the extended description of these categories in Table 1.

Table 1. Topics related to the CE of Tourist Board Websites for an alpine destination

Category	Key concepts	Description
Geography	Climate, Weather predictions, Land Formation, Lakes and Rivers, Landscape	Comprises characteristics of the landscape (mountain, lakes, plateaus), or geologic features (type of the rocks), characteristics of the environment (natural resources, parks, protected zones, biotopes) and climate (temperature, number of sunny days, precipitations, quality of the air, altitude).
Local products	Local handcrafting, Agricultural products, Gastronomy	Concerns all aspects that differentiate the production of one destination from the others: typical products of local agriculture, farming, and gastronomy (wine, cheese, sausage), or products related to forest – (wood, mushrooms, flowers). Local handicraft products are also important (sculptures, weaving, embroidery, etc.). Includes also markets and fairs.

Table 1. (Continued)

Culture	Traditions and customs, Local history Festivals, Population, Cultural institutions and associations, Libraries, Cinemas, Local literature, Local prominent people	Description of historical aspects related to the origins of the local population, ancient institutions of local governing (laws, municipal government, etc.) Other characteristics are folklore manifestations, customs and traditions related to particular periods of the year (e.g., carnival, Christmas). It includes also cultural associations (choruses, bands, etc.), libraries, reading-rooms, cinemas. Historical and contemporary personalities of the area, such as writers, photographers, mountain climbers.
Artistic Heritage	Places to visit: museums, castles, Tickets, fees, guides	Artistic heritage involves all that a tourist can visit or see in a destination; among these are: churches, castles, museums, ruins, castles, etc.
Sport	Sporting events, Sport infrastructure, Sport disciplines	Usually, it is distinguished between winter and summer sports. Among winter sports there are all types of snow activities (e.g. downhill, cross country skiing, snowboarding, etc.). Summer activities include excursions, climbing, Nordic walking, jogging, mountain bike, paragliding, etc. Other sports relate to this category, such as those practicable using special infrastructure (swimming pools, gold fields from golf, etc.). Relevant information is rental stations of sport equipment, training possibilities, enrollment to specific initiatives (alpine guides). Includes sport manifestations as well (e.g., marathon races).
Accommodation	Places to stay, How to book, How to arrive, Prices, Availability	Comprises all types of accommodation (hotel, pension, residence, bed and breakfast, rooms, campings, hostels, garni, hospitality in family). Apart from this, relevant information is modalities to make a reservation, offers, discount packages, prices must be indicated.
Food and refreshment	Places to eat, Dishes, Degustation, Time tables, How to book	Concerns various eating structures (restaurant, pizzeria, bar, malga), typical dishes, initiatives related to tasting of local products. Relevant information includes restaurants' opening hours, how to reach the place, how to make a reservation of a lunch, how to get to a restaurant.
Wellness	Wellness centers, Wellness services	Often, the wellness structures are present in hotels or thermal centers. Those include Turkish bath, sauna, massages, diets, gymnastics, beauty treatments.
Services	Transport, schedules, Information offices, Terminals, stations, airports, Travel agencies	Means of transports available in the region: buses, shuttles from the airport, ski-buses, trains; routes, timetables, parking areas, the distances from highways, railway and coach stations. Tourist information offices, Tourist boards and travel agencies.

5. *Semantic annotation based on the derived model.* Having derived a list of concepts of interest, we fulfilled the SA of the English-language content.

In order to populate the components of Cerno with annotation rules, the initial schema provided by the domain experts was expanded semi-automatically using definitions and synonyms provided by WordNet [29] database and the on-line Thesaurus [16]. The total number of annotation rules collected was 507.

To download and save the website content, we used an automatic tool called WebExtractor [27]. After that, we converted the websites from their format (HTML, PHP, or others) into textual form using Detagger [4], a tool which removes markups and scripts from the documents. As a result, we obtained a set of 11742 paragraphs of plain text and performed the SA of these paragraphs using Cerno. See a fragment of the annotated content of a website in Fig. 1. The technical details of the annotation process setup and evaluation can be found in [8].

```
<FoodAndRefreshment>Bread and wine snack in the shade of an elegant
park.</FoodAndRefreshment>

7.00 p.m.

<FoodAndRefreshment>Dinner at the "La Luna Piena" restaurant,
consisting of the "Il Piatto del Vellutaio" </FoodAndRefreshment>

9.00 p.m.

<ArtisticHeritage>Museo del Pianoforte Antico: guided visit and
concert proposed within the "Museum Nights" programme on the 3, 10,
17 and 24 of August.</ArtisticHeritage>
```

Fig. 1. Example of the annotated content of the website

6. *Analysis of the annotation results.* When the annotation process was completed, we calculated the total number of all the identified phrases relevant to varied categories, shown in Table 2. To provide a qualitative diagram of these results, Fig. 2 displays the coverage of different categories by all websites. From the both figure and table, we observe that the best total score was estimated for the website of A.P.T. Altopiano di Pinè e valle di Cembra. Moreover, this website also covers all the categories with at least one text fragment, which means that the CE was high both in terms of overall breadth and depth of relevant information provided.

The results revealed that the best-covered category among all others was Geography. The total sum of Geography-related annotation was 185. In fact, all of the considered websites include a page presenting the destination, which normally contains many geography-related terms, like "mountain", "valley", "alpine" and others. The highest scores were demonstrated by A.P.T. dell'Altopiano di Pinè e valle di Cembra (26) and A.P.T. San Martino di Castrozza, Primiero e Vanoi (24). Another well-represented category was Sport, whose total sum of annotations was 78. This result is also not surprising, given that the major focus of promotion of Alpine destinations is different kinds of mountain sport activities. The least covered category was Wellness. Its total count of annotations was 9. In fact, only few websites provided the information about wellness facilities, like beauty

centers, fitness courses, or massages. The site that contained most information about this category was A.P.T. Terme di Comano-Dolomiti di Brenta (4). Other under-represented category was Food and Refreshment, where the total count was 31 and only 10 of 13 websites provided related information. As the matter of fact, the information about places to eat was not covered by those versions of the website.

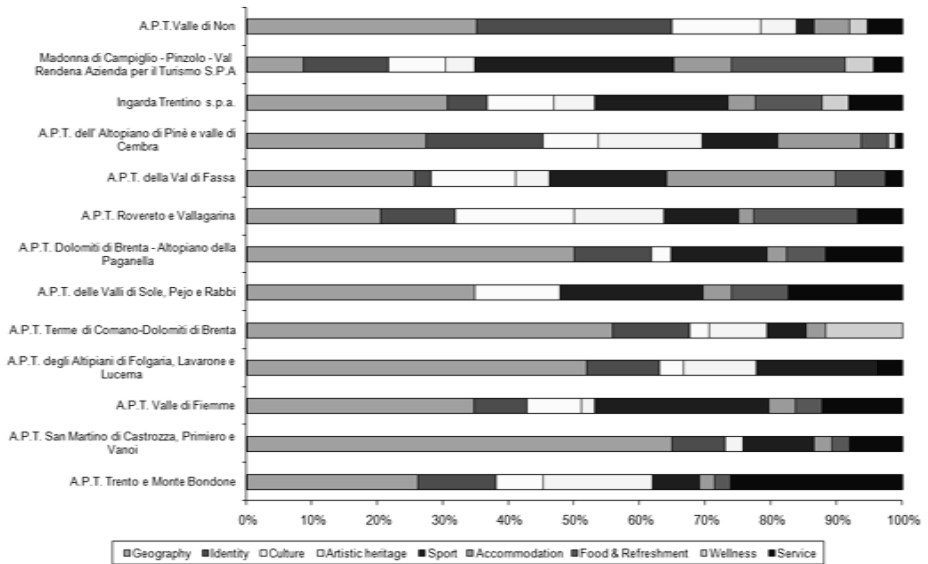


Fig. 2. Diagram of coverage by all categories by 13 websites

The results showed that the site of A.P.T. dell' Altopiano di Pinè e valle di Cembra had the best CE; it not only has the greatest total count of annotations, but also represented all of the categories. The lowest CE in terms of both breadth of knowledge coverage and extent to which this knowledge is elaborated (see section 2) was shown by the following sites: (a) A.P.T. delle Valli di Sole, Pejo e Rabbi with a total count equal to 23 and three categories are not represented at all; (b) A.P.T. degli Altipiani di Folgaria, Lavarone e Lucerna with a total count equal to 27 and three categories that are not represented. One website also demonstrated a low total score equal to 23 (Madonna di Campiglio – Pinzolo – Val Rendena Azienda per il Turismo S.P.A.); however, all the categories are represented by this site at least once. Consequently, this website demonstrated good breadth coverage of related information, but poor elaboration of this information.

The obtained results can be used to provide some recommendations to the Tourist Boards regarding the ways to improve their website content in order to achieve a better CE. For example, the importance of several categories, especially Wellness, Food and Refreshment, and Accommodation, was underestimated by the website developers. Therefore, more information related to these categories

must be provided. The overall representation of relevant information was not balanced. We found out that the majority of websites contained too much information related to Geography category compared to other categories (see the extreme left-hand bars of the diagram in Fig. 2). The best coverage in terms of both breadth and depth of detail was demonstrated by A.P.T. dell' Altopiano di Pinè e valle di Cembra. Moreover, this website also largely outnumbered all other sites obtaining a total count of 95 annotations. In comparison, the second best result demonstrated by two websites – of A.P.T. Valle di Fiemme and Ingarda Trentino S.p.a. – was only 49 annotation counts. Therefore, other Tourist Boards could take this result as a best practice example in order to improve their CE.

Table 2. Evaluation summary for the Tourist Board Websites

Agency (A.P.T.)	Category	<i>Geography</i>	<i>Identity</i>	<i>Culture</i>	<i>Artistic heritage</i>	<i>Sport</i>	<i>Accommodation</i>	<i>Food& Refresh.</i>	<i>Wellness</i>	<i>Service</i>	<i>Total</i>
Trento e Monte Bondone		11	5	3	7	3	1	1	0	11	42
San Martino di Castrozza		24	3	0	1	4	1	1	0	3	37
Valle di Fiemme		17	4	4	1	13	2	2	0	6	49
Altipiani di Folgaria		14	3	1	3	5	0	0	0	1	27
Terme di Comano, Brenta		19	4	1	3	2	1	0	4	0	34
Valli di Sole, Pejo e Rabbi		8	0	3	0	5	1	2	0	4	23
Dolomiti di Brenta, Pagan.		17	4	1	0	5	1	2	0	4	34
Rovereto e Vallagarina		9	5	8	6	5	1	7	0	3	44
Val di Fassa		10	1	5	2	7	10	3	0	1	39
Altopiano di Pinè, Cembra		26	17	8	15	11	12	4	1	1	95
Ingarda Trentino s.p.a.		15	3	5	3	10	2	5	2	4	49
Madonna di Campiglio		2	3	2	1	7	2	4	1	1	23
Valle di Non		13	11	5	2	1	2	0	1	2	37
Sum		185	63	46	44	78	36	31	9	41	

4 Conclusion

In this work, we emphasized the need in evaluating CE of websites. To address this issue, we proposed a systematic approach for estimating a website's CE where the key role is played by its content. Moreover, we suggested a tool to automate this evaluation, i.e., a SA framework Cerno, and showcased our method on a realistic case study. The study was fulfilled for 13 websites of Tourist Boards of the Italian autonomous province Trentino. The results of the study, though carried out using the basic metric for CE, allowed us to identify strong and weak points for each site and derive a number of useful recommendations for improving the communication between providers of tourist services and their target sectors of consumers. These problems were also confirmed after a manual check of the site content.

From an application viewpoint, this case study demonstrates how advances in modern information technologies and, in particular, SA and domain models can yield benefits to both providers and users of tourist services [12, 26].

Our future work will address the limitations of the first applications of the method in order to: (a) include a deeper investigation of the CE testing the metrics introduced in section 2 applying other modules of Cerno [31], so that to analyze the structural information of a document and XML tags [30] (step 6); (b) take into account information contained in multimedia content of websites [17] (step 5); and (b) validate the

method with different CE metrics on different types of web sites. Furthermore, we are interested in developing a toolset to support construction of a conceptual model (steps 3 and 4). From the research viewpoint, we aim to investigate what types of problems can be identified by applying different CE metrics. This will allow for a fine-tuning of our CE evaluation method to the conditions or expectations of a particular case study.

Acknowledgments. This work was partially sponsored by the PAPHYRUS project (ICT-215874).

References

1. Austin, J.L.: *How to Do Things With Words*. Harvard University Press, Cambridge (1962)
2. Chandler, D.: *Semiotics: The Basics*. Routledge, London (2002)
3. Coulleri, C.: *Semantic annotation of websites*, Master's thesis, University of Trento (2006)
4. Detagger tool description, <http://www.programmi.com/detail.asp?id=6311>
5. Doerry, E.: Evaluating distributed environments based on communicative efficacy. In: Katz, I., Mack, R., Marks, L. (eds.) *Proc. CHI 1995*, pp. 47–48. ACM, New York (1995)
6. eTourism research group, <http://www.etourism.economia.unitn.it>
7. Herrick, J.A.: *The History and Theory of Rhetoric. An Introduction*. Allyn & Bacon, Boston (1990)
8. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Text Mining through Semi-Automatic Semantic Annotation. In: Reimer, U., Karagiannis, D. (eds.) *PAKM 2006. LNCS (LNAI)*, vol. 4333, pp. 143–154. Springer, Heidelberg (2006)
9. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Cerno: Lightweight tool support for semantic annotation of textual documents. *Journal of Data and Knowledge Engineering* 68(12), 1470–1492 (2009)
10. Lefons, E., Paziienza, M.T., Silvestri, A., Tangorra, F., Corfiati, L., De Giacomo, P.: An algebraic model for systems of psychically interacting subjects. In: Dubuisson, O. (ed.) *Proc. IFAC Workshop Information & Systems, Compiègne, France*, pp. 155–163 (1977)
11. Local tourist boards, http://www.visittrentino.it/en/chi_siamo
12. Maedche, A., Staab, S.: Applying Semantic Web Technologies for Tourism Information Systems. In: Wöber, K., Frew, A., Hitz, M. (eds.) *9th Int. Conf. for ICT in Tourism (ENTER 2002)*, Innsbruck, Austria, pp. 311–319 (2002)
13. Mich, L., Franch, M.: Instantiating Web Sites Quality Models: an Ontologies driven Approach. In: *Proc. IWWOST 2005, Porto, Portugal* (2005), <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-153/paper5.pdf>
14. Mich, L., Franch, M., Martini, U.: A modular approach to quality evaluation of tourist destination website: The quality model factory. In: *Proc. Int. Conf. for ICT in Tourism (ENTER 2005)*, Innsbruck, Austria, pp. 555–565 (2005)
15. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. *IEEE Multimedia* 9(4) (2002)
16. On-line thesaurus, <http://thesaurus.reference.com>
17. Paci, G., Pedrazzi, G., Turra, R.: Wikipedia based semantic metadata annotation of audio transcripts. In: *Proc. of 11th Int. Work. on Image Analysis for Multimedia Interactive Services (WIAMIS)*. IEEE Computer Society, Los Alamitos (2010)
18. Robbins, S.P.: *Organizational Behavior – Concepts, Controversies, Applications*, 4th edn. Prentice-Hall, Englewood Cliffs (2004)

19. Shannon, C.E., Weaver, W.: *The Mathematical Theory of Communication*. University of Illinois Press, Urbana (1949)
20. Siorpaes, K., Simperl, E.: *Human Intelligence in the Process of Semantic Content Creation*. In: *World Wide Web*. Springer, Netherlands (2009)
21. Scott, W.G., Mitchell, T.R.: *Organization Theory: a Structural and Behavioural Analysis*. Concord, Irwin (1972)
22. Spohrer, J., Riecken, D. (eds.): *Introduction to Special issue on Services science*. *Communications of the ACM* 49 (7) (July 2006)
23. Stanca, L.: *Information society in Italy (2004) (in Italian)*, http://www.innovazione.gov.it/ita/normativa/pubblicazioni/rapport_stat_04_presentazione.shtml
24. Stuart, H.: *Towards a definitive model of the corporate identity management process*. *Corporate Communications* 4(4), 200–207 (1999)
25. Triacca, L., Bolchini, D., Botturi, L., Inversini, A.: *MiLE: Systematic Usability Evaluation for E-learning Web Applications ED Media 2004*, Lugano, vol. 1, pp. 4398–4405 (2004)
26. Walchhofer, N., Fröschl, K.A., Dippelreiter, B., Pöttler, M., Werthner, H.: *Semamo: An Approach to Semantic Market Monitoring*. *Information Technology & Tourism* 11(3), 197–209 (2009)
27. WebExtractor, <http://sourceforge.net/projects/webextractor>
28. Winograd, T., Flores, F.: *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley Professional, Reading (1987)
29. WordNet lexical database, <http://wordnet.princeton.edu>
30. Zeni, N.: *The Cerno Framework for Semantic Annotation: Extensions and Applications*, PhD thesis, University of Trento (2008)
31. Zeni, N., Kiyavitskaya, N., Mich, L., Cordy, J.R., Mylopoulos, J.: *A lightweight approach to semantic annotation of research papers*. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) *NLDB 2007*. LNCS, vol. 4592, pp. 61–72. Springer, Heidelberg (2007)

A Process Framework for Semantics-Aware Tourism Information Systems

Olawande J. Daramola

Department of Computer and Information Sciences,
Covenant University, Ota, Nigeria
jodaramola@covenantuniversity.com

Abstract. The growing sophistication of user requirements in tourism due to the advent of new technologies such as the Semantic Web and mobile computing has imposed new possibilities for improved intelligence in Tourism Information Systems (TIS). Traditional software engineering and web engineering approaches cannot suffice, hence the need to find new product development approaches that would sufficiently enable the next generation of TIS. The next generation of TIS are expected among other things to: enable semantics-based information processing, exhibit natural language capabilities, facilitate inter-organization exchange of information in a seamless way, and evolve proactively in tandem with dynamic user requirements. In this paper, a product development approach called Product Line for Ontology-based Semantics-Aware Tourism Information Systems (PLOSATIS) which is a novel hybridization of software product line engineering, and Semantic Web engineering concepts is proposed. PLOSATIS is presented as potentially effective, predictable and amenable to software process improvement initiatives.

Keywords: Product line, Web engineering, Software Process Improvement, Tourism Information System, Semantic Web, Software product development.

1 Introduction

The advent of new technologies has promoted thoughts of new intelligent possibilities in Tourism Information Systems (TIS) which has also amplified the complexity of the tourism requirements landscape. According to [1], as a result of emerging new technologies like the Semantic Web and mobile computing, the modern tourist is prone to a number of dynamic characteristics, which are: 1) become more mobile and critical; 2) become less loyal and frequently change their product preferences; 3) look for more specialized products and ask for better services; 4) want more and better information; 5) compare more products in more detail; 6) have fast changing needs and belong to different niches at the same time; and 7) tend to make more but shorter vacations. The dynamic characteristics of tourism consumers' behaviour have aggravated the complexity of functional requirements of TIS which portend a critical challenge for providers of e-tourism services [1], [2].

Although a number of attempts have been made to tackle this problem, none of the approaches so far reported in literature has been centred on a thinking that seeks to

solve this problem from a software development process perspective. This scenario raises a pertinent research question: 1) how can developers of TIS effectively respond to the trends in consumer behaviour from a product development perspective? Or in another form: Is there a product development approach or methodology that could be engaged to tackle dynamic requirements in tourism? This is the thinking that must guide the development of the next generation of TIS.

In this paper, a software process framework called Product Line for Ontology-based Semantics-Aware Tourism Information Systems (PLOSATIS) is proposed as a novel hybridization of software product line engineering, ontology engineering and semantic computing for developing next generation TIS. Specifically, the next generation of TIS are expected among other things to: enable semantics-based information processing, exhibit natural language capabilities, facilitate inter-organization exchange of information in a seamless way, and evolve proactively in tandem with dynamic user requirements [2], [3]. PLOSATIS leverages concepts and practises from ontology development because ontology is a fundamental component for achieving the Semantic Web. Ontology has the capability to solve a number of problems in tourism. This includes: 1) enabling interoperability of heterogeneous platforms; 2) standardization of business models, business processes, and knowledge architectures; and 3) serving as a model of knowledge representation for the generation of knowledge-based information services [4].

PLOSATIS also leverages the Software Product Line Engineering (SPLE) paradigm because of SPLE's ability to engender systematic and strategic reuse of software artifacts based on identifiable common domain requirements so as to facilitate rapid market-entry and flexible response to dynamic user requirements [5]. Hence, SPLE-based PLOSATIS is designed not only to enable the production of next generation TIS product families with semantic-awareness, but also, to facilitate the proactive evolution of such TIS in response to dynamic user requirements. Another vital aspect of PLOSATIS is semantic computing, because of the need to equip next generation of TIS with semantic processing capabilities that will engender meaning-based execution of information retrieval and information processing tasks such as semantic searching, semantic querying, and semantic browsing unlike conventional dumb portals that are currently prevalent in the e-tourism landscape [6]. Hence, PLOSATIS is designed to enable the development of TIS that will exhibit Semantic Web and semantics-aware information processing capabilities that is mostly non-existent in many e-tourism platforms.

The rest of the paper is organised as follows: Section 2 is a review of related work from literature, while Section 3 describes the PLOSATIS framework in detail. In section 4, an experimental application of PLOSATIS is presented and the results obtained, also a discussion on the preliminary assessment framework used to evaluate PLOSATIS is given. In Section 5 the paper is concluded with a brief note.

2 Related Work

Thus far, relatively few approaches for modeling of Semantic Web applications have been reported in literature. Some of these include: Semantic Hypermedia Design Method (SHDM) [7], [8] which is a model-driven approach to designing web applications using five steps: Requirements Gathering, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. SHDM also embraces the use of

ontological definition languages such as DAML+OIL and OWL, for expressing advanced aspects such as constraints (restrictions), enumeration and XML Schema datatypes. The OntoWebber system [9] is an ontology-based approach to website management. It facilitates the design, creation, generation and maintenance of Web sites using a set of software tools. It also enables the personalization of Web site views based on individual users. Another notable approach is the Hera project [10], which is a methodology that supports the design and engineering of Web Information Systems (WIS) using Semantic Web technology. The main focus of the Hera project is to support Web design and implementation particularly hypermedia aspects.

However, these aforementioned approaches are quite generic and do not address the peculiarities of specific application domains. Also, although they leverage Semantic Web technology to exhibit some forms of intelligence (personalization - OntoWebber, hypermedia presentation - Hera), they were not setup primarily to facilitate semantic-awareness in WIS as envisioned by PLOSATIS. Additionally, PLOSATIS, unlike others, is based on a product line perspective in order to facilitate proactive evolution of TIS (which are specializations of WIS for tourism).

3 Description of the PLOSATIS Framework

PLOSATIS is more amenable to a context where the following obtains: 1) All developed TIS belong to the same organization or a consortium of collaborating organizations; 2) the requirements of different variants of TIS products within the product family are well known and can be predetermined in advance; 3) the process description and tools for developing specific kinds of TIS product can also be predetermined. In essence PLOSATIS fits more into the context of a software developer organization.

- *The PLOSATIS Lifecycle:* The flow of activities in PLOSATIS (see Figure 1) is sequential, but also iterative. The breakdown of the activity workflow is as follows:
 - i) *Requirements and Process Analysis:* This consist of activities that provide the necessary managerial guide and organizational control that complements the technical aspects of ontology engineering, domain engineering, and Web product engineering of the PLOSATIS framework. Its main sub-activities are as shown in Figure 1. *Configuration management* ensures that changes that need to be made to products by way of upgrade and versioning are carefully planned in a way that makes them technically realizable without disrupting the design of the product line, while *critical evaluation* ensures that periodical evaluations are carried out at specific points, notably after ontology engineering, domain engineering and Web product engineering to determine whether the process should proceed based on cost incurred so far and rate of productivity.
 - ii) *Ontology engineering:* This comprises ontology modelling, ontology design, and ontology development activities, that are required for the realization of the reusable knowledge artifacts needed for the execution of the PLOSATIS process.
 - iii) *Domain engineering:* During domain engineering, the reference architecture for the product line is created that consists of the core components (reusable software assets that are needed for building the variant TIS products) of the product line. The components are constructed from scratch or sourced as Commercial off

the Shelves (COTS) components, which are then tested and certified for reuse in the product line. Also, semantic computing capabilities are introduced into the domain components using appropriate middleware tools and technologies, or through the implementation of appropriate algorithms for natural language understanding. The key sub-activities of domain engineering are: domain analysis, domain design, domain realization and domain testing.

iv) *Web product engineering*: This is concerned with the creation of specific TIS products through the reuse of core assets created in domain engineering. The operations at this level are typically the weaving together of customized domain components and the creation of hypermedia contents. The focus of Web product engineering is more toward integration and customization rather than construction based on reuse of existing domain components. Semantic capabilities can also be introduced at this level depending on the peculiarity of functionality required by a product. The core activities of product engineering are product analysis, product design, product realization, and product testing.

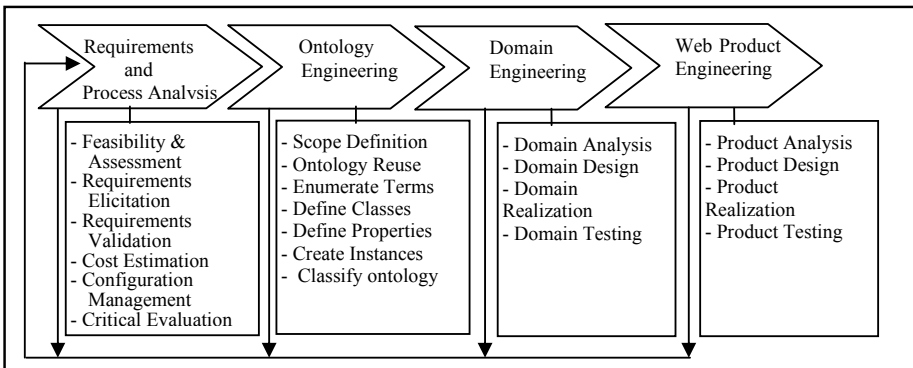


Fig. 1. PLOSATIS process Lifecycle

4 Experimenting with PLOSATIS

An empirical study with the PLOSATIS framework is being undertaken by the SEAI group of Covenant University, Nigeria. This entailed the creation of a prototype National Tourism Semantic Web Portal (NTSP) in order to archive, and provide web-access to Nigerian tourism data in a much more meaningful via the web and mobile platforms.

- *Requirements and Process Analysis in PLOSATIS*: PLOSATIS was initiated with the requirements and process analysis activity which probed into the expected functional characteristics of the NTSP. Important domain components and relevant ontologies for the NTSP were identified at this stage.
- *Ontology Engineering in PLOSATIS*: A suite of ontologies was developed to serve as the semantic backbone for prospective products of the PLOSATIS framework. Specifically, four tourism OWL ontologies were developed from scratch. This was

preferred to using existing tourism ontologies in order to evolve customized ontologies that comprehensively model the Nigerian tourism terrain, such that specific data on several tourism aspects such as destination, accommodation, restaurant, events, travel that pertain to Nigeria can be captured. The ontologies were designed to store contextual information about tourist locations and their operational time characteristics such that the high mobility of the modern tourist and the dynamic changes that can occur during a trip can be catered for, thus enabling the generation of personalized and context-aware travel information services. All the ontologies were implemented as OWL ontologies using the Protégé Ontology development tool. The ontologies are:

i) *Nigeria Destination Ontology (NDO)*: The NDO is a semantic representation of information on specific attributes of Nigerian destinations. The concept of a destination is conceived to consist of three types: *City*, *Town* and *Village*; each destination type has six attributes namely: *Tourism Asset*, *Weather Temperature*, *Scenery*, *Volume of Traffic*, *Crime Rate*, and *Status*. Also, Villages and Towns are conceived as extensions of specific City destinations and thus are related to specific destination subclasses instances using ‘*PartOf*’ association. *Tourism Asset* is an enumeration of the types of tourism artifacts that are available in a destination (10 different categories were identified i.e. beaches, landforms, conservation, recreational parks etc.); each category has equivalent *datatype* property (e.g. *hasLandforms*, *hasBeaches* etc.) which is used to capture the details (full name and location address) of all tourism assets belonging to that type (a *maximum cardinality* of 25 is imposed). Thus the NDO models specific destinations as an agglomeration of available tourism assets, and associated contextual information.

ii) *Nigeria Accommodation Ontology (NAO)*: The NAO is a semantic representation of the attributes of the various types of tourism accommodation. Seven specific attributes of accommodation types (e.g. hotel, guest house, hostel, chalet etc.) were considered. These are 1) *Services* - the description of kinds of services rendered; 2) *Gastro* - profile of eateries, cuisines or restaurant nearby; 3) *Attraction*- special attractions within or nearby; 4) *State*: province or region where it is located; 5) *Facilities* - physical facilities available; 6) *Location*- specific location of the resource and 7) *Time* – the working time period of the establishment.

iii) *Nigerian Events Ontology (NEO)*: The NEO was created to capture information about the events that are considered significant for Nigerian tourism. The NEO consists of five disjoint subclasses namely: *Location*, *State*, *Sponsor* (the organization that sponsors the event), *Time* (month of the year), and *Event* (a classification of the event into specific event category). Seven types of event category were identified. These are: Cultural, Social, Educational, National, Political, Religious, and Sports.

iv) *Nigerian Restaurants Ontology (NRO)*: The NRO is a semantic representation of information on Nigerian restaurants. The NRO consists of four disjointed classes namely: *Restaurant*, *State*, *Location*, and *Time*. The class *Restaurant* was defined as consisting of 26 subclasses which correspond to 26 standard types of restaurants to which a typical restaurant can belong. All the ontologies import the 3WC Time [11] and the 3WC Geo [12] ontologies in order to model user’s time and location context.

iv) *Nigerian Travel Ontology (NTO)*: The NTO is an integral part of the NTSP architecture though it is not yet implemented. It is conceived as a composition of all important knowledge artifacts relevant to the Nigerian travel industry. Aside importing the NDO, NAO, NEO and NRO, the NTO will also contain knowledge about other tourism objects such as transportation (land, and local air travel).

- *Domain Engineering in PLOSATIS*: The domain engineering activities entails the identification, design, and implementation of the core components of the National Tourism Semantic Portal (NTSP) architecture. The NTSP has a layered architecture consisting of the five layers (See Figure 2). The layers are described as follows:
 - i) *Client Layer*: This layer is comprised of client devices through which the services of the NTSP are requested. Typical clients include PDAs, web browsers (through Laptop and PC), Web services and Mash-up applications.
 - ii) *Protocol Layer*: This layer defines the implementation platform for all the services in the NTSP. This can be WAP for WAP-enabled mobile applications, SMS-based applications or HTTP for Web clients. This layer is responsible for the determination of appropriate communication protocol depending on the nature of requesting client and the provision of context-aware services.
 - iii) *Portal Services Layer*: The portal services layer consists of the class of all services that are available in the NTSP architecture. These services are rendered by specific domain components that leverage semantic information stored in the underlining ontology layer. Services can be broadly classified as 1) information services such as semantic query processing, semantic searching, and semantic browsing; and 2) Transaction services which are basically recommendation services rendered by different recommender systems, for destination, accommodation, restaurant, and events.
 - iv) *Semantic Middleware Layer*: This layer encapsulates the set of middleware infrastructures that enables the NTSP components with Semantic Web (to read, and process facts stored in ontologies) capabilities, and enables semantics-aware processing of user queries. Relevant middleware components, tools, and algorithms (such as: stemming, part-of-speech tagging, content summarization, and query reformulation, data mining) are deployed at this layer in order to trigger semantic reasoning, semantic searching, semantic mining, semantic browsing, and semantics-based recommendation services that are available on portal services layer of the NTSP.
 - v) *Data and Ontology Layer*: This layer represents the semantic backbone for all services in the architecture. The components of this layer are the suite of knowledge representation ontologies that are designed to facilitate intelligent personalized services in the NTSP. This layer also contains the set of database abstractions that stores the tourism information contents that are exploited by the NTSP.
 - *Implementation of Domain Components*: The implementations of domain components were based on Java technology, using the Jena Semantic Web development framework and the NetBeans 6.5 Java IDE. The Sun Application Web Server 9.0 was used as middleware for deploying server components. The Web GUI and functionalities were implemented using Macro

Media Flash and Dream Weaver web design tools, and Java Server Pages (JSP). The recommender systems were implemented as Enterprise Java Beans (EJB) components embedded in the web interface. Each of the EJBs references the specific ontology to which they were mapped using the Jena ontology APIs [13] to trigger ontology querying and reasoning capabilities. The Pellet 2.0 Descriptive Logics (DL) reasoner [14] was used as the reasoning engine for the ontologies, while JESS was used as the rule engine. All the recommender system components in the NTSP are knowledge-based RS. Each of the ontology in the NTSP is mapped to a corresponding recommender system, the recommender systems then generates list of top nearest neighbourhood recommendations using a content-based filtering approach. So far two of these recommender systems have been realized which are the destination recommender and the Accommodation recommender system. The other recommender systems are scheduled to be realized in the later phases of the project, So far, a first prototype of the NTSP has been built which is an extension of the work reported in [4].

Fig. 2. A layered schematic architecture of the NTSP

- *Web Product Engineering in PLOSATIS:* During this phase, the domain components were customized with specific functionalities, ontologies populated with specific data instances, and creation of hypermedia contents (web graphics, animations, sound etc.) in order to realize the TIS products based on the feature model of the product line. The feature oriented domain analysis method (FODA) [15] was used. Based on the feature model, the destination recommender, accommodation recommender, and semantic query engine are the mandatory components in TIS product line, while other components are optional. Thus far the compulsory components have been realized in the first prototype of the NTSP.

4.1 Evaluation Framework for PLOSATIS

Three of the seven key goals of a software process as proposed in [16] are considered as most relevant to the context of PLOSATIS as a proposed software process. These are *effectiveness*, *predictability*, and *improvement*. Hence the choice to evaluate PLOSATIS along these three dimensions.

- *Determining Effectiveness of PLOSATIS*: The effectiveness of a software process is solely determined by its ability to deliver the right product that is, the extent to which its output is valuable to the user. A potentially viable approach for evaluating the effectiveness of PLOSATIS is the Goal Question Metric (GQM) paradigm [17]. The GQM is a kind of software measurement that offers a systematic approach for tailoring and integrating goals with models of software processes, products and quality perspectives of interest, based on the specific needs of an organisation [17]. It is a 3-level process comprising of: the identification of the key goals of a process, the formulation of goal-specific assessment questions, and the definition of metrics to generate data to answer the goal-specific questions in a quantitative way. For instance, in the NTSP project, the adopted GQM model to determine the effectiveness of PLOSATIS is presented as follows:

The PLOSATIS Effectiveness Model:

Let:

$M1_c$ = user rating for M1; $M2_c$ = user rating for M2; $M3_c$ = user rating for M3;
 $M4_c$ = Confidence Factor; $M5_c$ = user rating for M4

Let $\{PQ_i\}$ be set of Weighted Product Quality per User [each $PQ_n = (M1_c * M4_c * M5_c)$]

Where Maximum $PQ_n = 125$ [for a perfect user rating], $n \leq i$

Let $\{AF_i\}$ be set of Product Appreciation Factor per User

[each $AF_n = [PQ_n / (M2_c * M3_c)]$]

Where Maximum $AF_n = 25$ [for a perfect user rating], $n \leq i$

Interpretation Algorithm for PLOSATIS Effectiveness Model

[Where 62.5 is a benchmark value that corresponds to average rating (PQ_n) of PLOSATIS by a user]

If 70% or more of $\{PQ_i\} \geq 62.5$ then PLOSATIS is perfectly effective process

If 60% - 70% of $\{PQ_i\} \geq 62.5$ then PLOSATIS is very effective process

If 50% - 60% of $\{PQ_i\} \geq 62.5$ then PLOSATIS is mostly effective process

If 45% - 50% of $\{PQ_i\} \geq 62.5$ then PLOSATIS is barely effective process

If less than 45% of $\{PQ_i\} \geq 62.5$ then PLOSATIS is not effective process

Figure 3 gives a brief overview of questions and metrics of data collection for the draft GQM.

Using the draft GQM model in Figure 3, a usability experiment can be undertaken using a population of qualified users of the system as participants to determine the effectiveness of the PLOSATIS process with respect to the specific problem scenario at hand. Questionnaire based on the developed GQM model can be given to each of the participants to capture their evaluations. The collected data could subsequently be analysed statistically to determine the measure of effectiveness of PLOSATIS. A first-cut GQM evaluation conducted by our group revealed PLOSATIS as a mostly effective software process (details not provided due to space constraint).

- question 1*: Can you characterize your experience with Tourism Information System (i.e. e-Tourism websites)?
- etric 1*: Subjective rating per person [0-5]; 0 (lowest) – 5 (highest)
- question 2*: How strong is your desire to obtain intelligence and semantic-awareness from Tourism portals?
- etric 2*: Subjective rating per person [0-5]; 0 (lowest) – 5 (highest)
- question 3*: Can you characterize your perception of the level of semantic-awareness of tourism portals based on experiences from the past?
- etric 3*: Subjective rating per person [0-5]; 0 (lowest) – 5 (highest)
- question 4*: What is your assessment of the quality of semantic-awareness of PLOSATIS products?
- etric 4*: Confidence factor = (number of agreeable outcomes) / (number of trials)
- etric 5*: Subjective rating per person [0-5]; 0 (lowest) – 5 (highest)

Fig. 3. Questions and Metrics used for the GQM

- **Determining the Predictability of PLOSATIS:** A software process is deemed predictable if it is possible to reasonably estimate the cost of developing products by using the process. According to [16], a good software process is one that layout clearly the steps of development such that planning for new products, and proper allocation of resources of both time and people, can be done ahead of time. PLOSATIS satisfies this requirement because it has clearly defined activities, from which specific cost parameters can be derived for project cost predictions. A cost estimation model that is based on augmentation of the SPL estimation model given by [18] has been adopted for PLOSATIS. The cost model which computes efforts in Person Months is given as follows:

$$E_{plosatis} = E_{req} + E_{onto} + E_{dom} + E_{ontoupdate} + N * (E_{reusewith} + E_{uniquewith} + J * E_{updatewith})$$

Where

E_{req} , E_{onto} , E_{dom} , $E_{ontoupdate}$ - are efforts expended on: requirements engineering, ontology engineering, domain engineering, and periodic update and maintenance of ontologies used in the project, respectively;

$E_{reusewith}$, $E_{uniquewith}$, $E_{updatewith}$ - are efforts expended on: Web product engineering for the reuse of existing core assets, the manual adaptations of core assets after creation, and the update of product-related core assets in the core asset base, and creation of additional core assets that are unique to a product respectively;

N - The number of TIS products in the product line; and

J - The average planned number of content update cycles for one TIS product.

E_{onto} and $E_{ontoupdate}$ can be estimated based on the ONTOCOM model as proposed in [19], while the other metrics of the estimation model can be based on COCOMO II [20]. This derived effort estimation model for PLOSATIS therefore provides a basis for a priori cost prediction of future projects using relevant historical data in the instance of the adoption of PLOSATIS by an organization.

- **Amenability to Software Process Improvement of PLOSATIS:** Amenability to software process measurement and improvement is a vital quality of a good software process. However, the hybrid nature of PLOSATIS ensures that none of the existing software assessment frameworks such as: SPICE, CMMI, BOOSTRAP and Goal Question Metric (GQM) [21] is perfectly suitable for its assessment as a software process. Therefore a 32-questions template that spans the four aspects of the PLOSATIS development lifecycle has been formulated which could be used as basis for Software Process Improvement (SPI) initiative. The questions (see Table 1) were based on best-known practices and rules that govern the conduct of software product line initiatives [21], Ontology engineering activities [4], and Web engineering [7], [8], [10].

Table 1. Questions Template for Software Process Assessment of PLOSATIS

Requirements and Process Analysis	
Question 1.	Does the organization's management in support of the pursuit of the software product line initiative and subscribe to its principles?
Question 2.	Is the influence of the management team on the execution and overall success of the software product line initiative considered significant?
Question 3.	Is adequate risk assessment undertaken in order to compare the expected investment with accruable benefits that can be gained from the pursuit of software product line before commencement of projects?
Question 4.	Is a comprehensive domain analysis undertaken prior to product development in the software product line?
Question 5.	Is there a requirements engineering scheme for the software product line that ensures that the product line requirements are well documented, defined, analyzed, verified and managed?
Question 6.	Is there a reference architecture that fully captures the base requirements of the software product line and clearly identify the possible variability points in the software product line outputs.
Question 7.	Is there a configuration management scheme in place to address the configuration management issues that arise in the software product line?
Question 8.	Is a cost-benefit analysis conducted at the end of each software product line project to evaluate the pay-off of such efforts?
Question 9.	Is the cost benefit ratio or ROI of the software product line in agreement with the organization's financial projections?
Question 10.	Are the integral activities of software product line development executed iteratively?
Ontology Engineering Assessment	
Question 11.	Is the ontology engineering effort intensive and considered highly relevant?
Question 12.	Is the schedule for ontology development well documented?
Question 13.	To what extent are scheduled ontology development tasks executed as planned?
Question 14.	Is the conceptualization of the ontology in line with the specification of the product line architecture?
Question 15.	Are all ontologies consistent with the scope of the software product line?
Question 16.	Are adequate provision made for the timely update of the ontology with a documentation of update schedules?
Question 17.	Does the product line scope allow for the addition of new ontologies or integration of existing ontologies with other relevant ontologies as requirements evolve?
Question 18.	Are there provisions for ontology support activities such as acquisition, evaluation, integration, merging, and alignment?
Domain Engineering	
Question 19.	Are all of the domain components within the software product line core asset base and the eventual TIS products consistent with the software product line scope?
Question 20.	Are the variability points in all domain components in the core asset base well defined such that they can be properly customized for variant TIS products composition?
Question 21.	Is there a scheme in place to ensure that the core asset base gets updated regularly through the addition of new domain components as the TIS product line evolves?
Question 22.	Is there a version control management system in place to keep track of the domain components development and reuse history?
Question 23.	Do all the third party components (COTS) present or added into core asset base satisfy the cost benefit ratio for the organization?
Question 24.	Is the degree of domain components reuse very high relative to the number of TIS products developed within the software product line?
Web Product Engineering	
Question 25.	Is there a reference architecture that is shared by all TIS products within the software product line?
Question 26.	Is the degree of commonality among TIS products generated by the software product line very high?
Question 27.	Does the variation among TIS products well-known ahead of time and fully accommodated within the definition of the software product line scope?

Table 1. (Continued)

Question 28.	Does every TIS product generated from the software product line represent a valid business case for the organization?
Question 29.	Does the software product line produce a many variant TIS products, or at least more two?
Question 30.	Is there sufficient automated support for customization activities and domain component integration activities to realize variant TIS?
Question 31.	Is there support automated tool support for hypermedia creation activities that would aid navigation and presentation of information in TIS products?
Question 32.	Does every TIS product released from the software product line certified as suitable by the organization?

The formulated assessment template (see Table 1) though not exhaustive could provide adequate basis for the application of standard SPI frameworks such as SPICE, CMMI, or GQM to PLOSATIS. For example, using the questions of the assessment template, coupled with the definition of appropriate goal-specific metrics, relevant quantitative data can be generated that would facilitate successful GQM evaluation of PLOSATIS along specific software process quality dimensions for improvement. Hence, PLOSATIS is quite amenable to software process improvement in the instance of its adoption by a TIS developer organization.

4.2 Discussions

The experience so far with PLOSATIS has revealed its potential to enable proactive evolution of TIS in tandem with emerging dynamic requirements. For example, the first set of prototypes that have been implemented does not yet cover the full scope of the reference architecture of the TIS product line, but there remain sufficient opportunities for future extensions and products evolution based on the predefined versioning scheme, which is to revise the products every two years. In order to realize other variants of the existing TIS product all that will be required is to add one or more of optional domain components that were already specified in the reference architecture. While mobile versions of all TIS products can be produced by customizing some of the exiting domain components for the mobile platform. Also, other future additions could be made to the reference architecture based on the dynamics of user requirements within this specific domain which will in turn provide a basis for new product variants that possess additional features and more advanced functionalities to evolve.

5 Conclusion

In this work, the PLOSATIS process framework has been proposed as a novel and viable software process for the development of semantics-aware TIS. It offers a product development platform for the emergence of next generation TIS that are capable of exhibiting higher level of intelligence compared to existing TIS. It also facilitates the proactive evolution of such TIS in tandem with emerging user requirements in the tourism domain. Initial experiment through a case study in product line development gave credence to the viability of the notion of PLOSATIS. PLOSATIS is currently an ongoing work, and so in future, industrial case studies of the application and adoption of PLOSATIS will be conducted to further validate it.

References

1. Steinbauer, A.: Consumer Behaviour in e-Tourism, Doctoral Thesis, Department for Information Systems and e-tourism Ludwig-Franzens-Universität Innsbruck (2002/2005)
2. Werthner, H., Klein, S.: Information Technology and Tourism—A Challenging Relationship, p. 23. Springer, New York (1999)
3. Staab, S., Werthner, H., Ricci, F., Zipf, A., Gretzel, U., Fesenmaier, D.R., Paris, C., Knoblock, C.: Intelligent systems for tourism. *IEEE Intelligent Systems* 17(6), 53–66 (2002)
4. Daramola, O., Adigun, M., Ayo, C.: Building an Ontology-based Framework for Tourism Recommendation Services. In: ENTER 2009, Amsterdam, Netherlands, pp. 135–147 (2009)
5. Clements, P., Northrop, L.: Software Product Lines, Practices and Patterns. The SEI Series in Software engineering. Addison-Wesley, Reading (2002)
6. Maedche, A., Staab, S.: Applying Semantic Web Technologies for Tourism Information System. In: Proceedings of the 9th International Conference for Information and Communication Technologies in Tourism (ENTER 2002), Innsbruck, pp. 113–124 (2002)
7. Lima, F., Schwabe, D.: Application modeling for the Semantic Web. In: Proceedings of the 1st Latin American Web Congress, Washington, DC, USA, pp. 93–102. IEEE Computer Society, Los Alamitos (2003)
8. Schwabe, D., Szundy, G., Silva de Moura, S., Lima, F.: Design and Implementation of Semantic Web Applications. In: WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, USA (May 2004), CEUR-US.org
9. Jin, Y., Xu, S., Decker, S., Wiederhold, G.: Managing Web Sites with OntoWebber. In: Proceedings of the 8th International Conference on Extending Database Technology, Prague, Czech Republic, pp. 766–768. Springer, Heidelberg (2002)
10. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. *Journal of Web Engineering (JWE)* 2(1-2), 3–26 (2003)
11. W3C Time, <http://www.w3.org/TR/2006/WD-owl-time-20060927/> (accessed: 12/12/2010)
12. W3C Geo, <http://www.w3.org/2003/01/geo/> (accessed: 12/12/2010)
13. <http://jena.sourceforge.net/> (accessed: 08/10/2009)
14. <http://clarkparsia.com/pellet/> (10/11/2009)
15. Kang, K., Cohen, J., Novak, W., Peterson, S.: Feature-oriented domain analysis feasibility study. Carnegie Mellon University, Software Engineering Institute, Tech. Rep. CMU/SEI-90-TR-21, pp. 1–82 (1990)
16. Tyrrell, S.: The Many Dimensions of the Software Process. *ACM Crossroads*, 22–26 (Summer 2000)
17. Basili, V., Caldiera, G., Rombach, H.D.: Goal Question Metric Approach. In: *Encyclopedia of Software Engineering*, pp. 528–532. John Wiley & Sons, Inc., Chichester (1994)
18. Bockle, G., Clements, P., McGregor, J.D., Mathig, D., Schmid, K.: Calculating ROI for Software Product Lines. *IEEE Software* 21(3), 23–31 (2004)
19. Paslaru-Bontas Simperl, E., Tempich, C., Mochol, M.: Cost estimation for ontology development: applying the ONTOCOM model. In: Abramowicz, W., Mayr, H.C. (eds.) *Technologies for Business Information Systems*, pp. 327–339. Springer, Netherlands (2007)
20. Boehm, B., Abts, C., Winsor Brown, A., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: *Software Cost Estimation with COCOMO II*, Upper Saddle River, N.J. (2000)
21. Ahmed, F., Capretz, L.F.: A Framework for Process Assessment of Software Product Line. *Journal of Information Technology Theory and Application (JITTA)* 7(1), 135–157 (2005)

Use of Hypermedia Tools for End-User Development

Sebastian S. Ortiz-Chamorro^{1,4}, Gustavo Rossi^{1,2}, and Daniel Schwabe³

¹ LIFIA, Universidad Nacional de La Plata, Argentina

² CONICET, Argentina

³ Departamento de Informática, PUC-Rio, Brazil

⁴ Departamento de Electrónica e Informática, Universidad Católica de Asunción, Paraguay
{sortiz,gustavo}@lifia.info.unlp.edu.ar, dschwabe@inf.puc-rio.br

Abstract. Software development tools aimed at end-users tend to employ various forms of visual programming because these users find textual programming very difficult to learn. However, visual programming has known scalability issues. As an alternative, we propose hypertextual programming; a technique that represents the program as hypertext and allows the user to both browse it and manipulate it mainly by using navigation. This technique leverages the users' ability to navigate in hyperspace, a widely available skill, to edit the program under development. In order to reap the benefits of this technique, adequate hypertextual editors must be built. Many of the lessons learned in the web engineering area can be used to deal with this problem. This paper discusses the state of the current research efforts behind this novel programming technique.

Keywords: hypertextual programming, hypermedial programming, end-user development, interfaces for end-user development, domain-specific languages, web engineering.

1 Introduction

Textual programming languages force the user to “learn the arcane syntax and vocabulary conventions of the language” [2]. Learning a text-based language constitutes a difficult and undesirable challenge for end-users doing software development.

To mitigate this problem, visual programming techniques [4] have successfully been used in end-user development [3]. However, visual programming is known to have scalability problems [5].

The potential length and complexity of the end-user's programs call for a representation of the program code as a set of manageable pieces that the user can both inspect and manipulate.

Hypermedia (a.k.a. hypertext¹) systems [6] provide interactive environments where users can navigate and manipulate a potentially vast network of interconnected pieces of information. This paper discusses the current research efforts on a programming technique based on the use of hypermedia development environments.

¹ We adopt the definitions in which these terms are synonyms.

1.1 Research Objectives

Research Question: How could hypermedia representations of programs be used to build better end-user oriented development tools?

General Objective: explore the potential use of hypermedia-based techniques in the creation of end-user oriented development environments.

Specific Objectives:

- Select, elaborate and integrate a set of concepts, models and tools to be used in the design and implementation of end-user oriented development environments that both use and provide adequate support to the representation of computer programs as hypermedia systems.
- Perform a detailed description of the fundamental features that a hypermedia-based programming environment must have and define adequate vocabulary to refer to these elements, consistent with the concepts used in specialized literature.
- Perform a critical comparison of the developed techniques against other editing and program visualization techniques.
- Select and adapt existing hypermedia development methodologies to be used in the construction of hypermedia-based development environments.
- Construct functional hypermedia-based development environments.
- Study the use of these development tools to assess their efficacy.

1.2 Research Methodology

A technique that aims to construct hypermedia-based software development tools has to be both adequately defined and validated.

In order to define this technique, we will provide: i) a formal definition of hypertextual programming, and ii) a theoretical analysis that compares textual, visual and hypertextual programming. This analysis aims at providing a clear distinction between these techniques.

In order to validate this technique, we will provide two hypertextual development environments that use hypertextual programming in different end-user oriented domains. One of the applications will be used for building general-purpose mashups. The second application, Benefit Catalog, uses hypertextual programming to build interactive rule-driven questionnaires to automatically generate health-care insurance policies.

These applications offer two different kinds of empirical evidence to assess the effectiveness of hypertextual programming. Benefit Catalog is a real-world application used by health insurance domain-experts. The mashup tool will be used in laboratory experiments with end-users. Using different domains will help to assess how general this technique is and will also provide guidelines to identify areas where it may be more effective.

2 Hypertextual Programming

Hypertext has been used in programming before. We reviewed three representative examples [10,11,12]. In general, all of these tools and techniques assume that there are one or more underlying programming languages and use hypertext to rearrange and/or link the potentially different program sources with other documents and products of the

software engineering process [1]. Although these development environments qualify as hypertext systems, they do not use navigation as a means to edit the source code.

2.1 Definition

We defined *hypertextual programming* (HTP, a.k.a. *hypermedial programming*) as a form of programming that uses navigation as the primary tool to inspect and edit the application code, and is supported by a computer system, called a *hypertextual editor* (a.k.a. *hypermedial editor*), that: i) represents the entire program source code as hypertext; and ii) allows all the possible finite language instances to be generated as navigation paths through it [1].

OOHDM [7], a general Web engineering technique, can be used to construct a hypertextual editor for a domain-specific language [1]. Other design guidelines like Web usability [8] and hypermedia design patterns [9,10] can be leveraged as well.

2.2 Comparison with Visual and Textual Programming

HTP does not force the end-user to learn a textual programming language. At the same time, HTP does not appear to have the vertical scalability problems that plague visual programming because large amounts of code can be organized in various interconnected views of the program code, if necessary, at different levels of abstraction [1].

3 Benefit Catalog

As a first example application, we participated in the development of Benefit Catalog, a hypertextual editor that is currently being used by end users to program dynamic health-care insurance policy configuration in a company in the United States [1].

In this development tool, the various program components such as product templates, questions, answers, cost sharing components, benefit options and benefit service-levels are represented as navigational objects. Programs are developed by creating, modifying and removing these objects.

Benefit Catalog is a hypertextual editor that has been used to effectively build complex programs by end-users in the health-care industry. This application is particularly important because the programs built by the end-users are very large and complex. This provides empirical evidence to substantiate the claim that this technique may be used to build scalable developing environments.

We are currently in the process of making a detailed study to quantify the size and complexity of these programs, but our preliminary findings show that the size of these programs –as measured by units of instructions comparable to lines of code– is above 10,000 units. We know of no other documented cases of programs developed by end-users that have this size and complexity.

4 Conclusions and Future Research

Thinking of a Web application as a development tool that both represents and enables the modification of an underlying programming language constitutes a novel application of Web technology.

Throughout this research we introduced the concept of hypertextual programming and provided a formal definition. We also provided guidelines to help construct development environments that employ this technique [1].

We have compared hypertextual programming against visual and textual programming. HTP does not force the user to learn the syntax of a text-based language and also does not suffer from the vertical scalability problems that plague visual programming. However, a more detailed discussion of the differences between these techniques and how they draw on different human abilities to support programming activities is needed.

The proposed research methodology has provisions to substantiate our claims around this technique both on theoretical and on empirical grounds. In the case of empirical evidence, we currently have a significant real-world example of this technique in the form of the Benefit Catalog application. A second hypertextual editor for building mashups and code-named Mashcraft is currently under development. We expect to perform laboratory experiments with this tool. Using these diverse forms of empirical validation will allow us to assess the effectiveness of this technique both under the view of domain-experts that need a practical solution and also under the controlled environment provided by laboratory experiments.

Millions of users navigate the World Wide Web. Hypertextual programming leverages this widely available end-user skill to facilitate the construction of computer programs.

Acknowledgements

Creating, studying and validating this technique constitute the core of a Ph. D.² thesis carried out by Sebastian S. Ortiz-Chamorro at the National University of La Plata in Argentina. The advisors for this thesis are Gustavo Rossi, Ph. D. from the same university and Daniel Schwabe, Ph. D. from PUC-Rio. Developing a hypermedia-based mashup development tool is a sub-part of this investigation that is being carried-out as a research project at the Catholic University of Asuncion, Paraguay. This project has received funding from Conacyt³.

References

1. Ortiz-Chamorro, S.S., Schwabe, D., Rossi, G.: Hypertextual Programming for Domain-Specific End-User Development. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 225–241. Springer, Heidelberg (2009)
2. Cypher, A. (ed.): Watch What I Do: Programming by Demonstration. MIT Press, Cambridge (1993)
3. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 1–8. Springer, The Netherlands (2006)

² Carrera de Doctorado en Informática.

³ Conacyt, Consejo Nacional de Ciencia y Tecnología (National Council of Science and Technology) is a Paraguayan government agency.

4. Shu, N.: Visual Programming. Van Nostrand Reinhold, New York (1988)
5. Burnett, M.M., Baker, M.J., Bohus, C., Carlson, P., Yang, S., van Zee, P.: Scaling up Visual Programming Languages. *IEEE Computer* 28(3), 45–54 (1995)
6. Conklin, J.: Hypertext: an introduction and survey. *Computer* 20(9), 17–41 (1987)
7. Schwabe, D., Rossi, G.: An Object Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems* 4(4) (1998)
8. Nielsen, J.: *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis (1999)
9. Rossi, G., Schwabe, D., Garrido, A.: Design reuse in hypermedia applications development. In: *Proceedings of Hypertext 1997*, pp. 57–66 (1997)
10. Hypermedia Design Patterns Repository,
<http://www.designpattern.lu.unisi.ch/index.htm>
11. Østerbye, K.: Literate Smalltalk Programming Using Hypertext. *IEEE Transactions on Software Engineering* 21(2), 138–145 (1995)
12. Anderson, K.M., Taylor, R.N., Whitehead, E.J.: Chimera: hypermedia for heterogeneous software development environments. *ACM Transactions on Information Systems* 18(3), 211–245 (2000)
13. Garg, P.K., Scacchi, W.: ISHYS: Designing an Intelligent Software Hypertext System. *IEEE Expert: Intelligent Systems and Their Applications* 4(3), 52–63 (1989)

A Document-Centric Approach to Open Collaboration Processes

Nelly Schuster, Christian Zirpins, and Stefan Tai

Karlsruhe Institute of Technology (KIT),
76131 Karlsruhe, Germany
{firstname.lastname}@kit.edu

Abstract. Individuals collaborate with each other in groups in order to solve complex tasks or to jointly develop new solutions. The Web offers them a huge amount of data, information and services to be used as input to their collaboration. In our research we examine such creative and emergent open collaboration processes and their relationship to documents as a communication and collaboration vehicle. Our goal is to better support coordination and integration in such collaborations. From a technological perspective, we aim to support open collaborations by putting a service-oriented document model and middleware in the center of a collaboration. As a first step, we provide a concrete design of a document collaboration system representing documents as rule-based mashups of RESTful services provided by humans, enterprise systems or the Web. As a second step, we aim to analyze communication of resources in open collaboration processes. We strive to use this analysis data in order to a) provide end-users of our collaboration system with a visual method in order to help them understanding the evolution of their document collaboration ecosystem and processes and to b) allow detecting potential inconsistencies in rule-based process definitions which can be caused by the open ecosystem of resources. In this paper we give a short overview of the current state of our research and outline future research directions.

Keywords: document mashups, open collaboration processes, document engineering, rule-based mashup.

1 Research Context

Creative collaborative design of software systems or research activities in teams are examples of so called emergent knowledge processes [1]. These processes denote the engineering of documents involving different individuals, activities and resources, which might be changed in an ad-hoc way. These open collaboration processes require lightweight coordination support. While structured or semi-structured processes can be modeled in advance, open collaboration processes emerge during runtime often in unforeseen ways. For instance, at the beginning of such a collaborative process it is not clear, which resource is affected or needed at which time and who participates when in the collaboration. Thus, methods and tools for open collaborations need to be able to adapt to changing participants and resources. In addition, there need to be mechanisms to capture and specify dependencies between activities.

Besides coordination support, open collaboration processes require content consolidation support. The Web offers a steadily growing amount of distributed resources providing content or functionality, e.g. linked open data or translation services. These content or functionality services can be used, reused and exchanged in open collaborations. In order to allow easy integration of these sources into open collaboration, new kinds of tools are required which are able to cope with this evolving ecosystem of autonomous and heterogeneous Web resources.

Various approaches exist to support flexible design and execution of collaborative processes [2, 3]. However, they do not focus on processes targeted at collaboratively engineering documents. On the other hand, Web-based collaboration technologies like online collaborative writing tools have improved the effectiveness as regards information consolidation and communication, but they do not explicitly drive the coordination of collaborations and information integration.

New kinds of software and architectures can be built from compositions of Web resources [4]. For example, mashups of services allow end-users to build composite applications for situational purposes [5]. We claim that this new kind of end-user-driven compositions based on service-oriented computing technology can be leveraged to support open collaboration processes.

In the following Section 2, we describe the general goals and planned contributions of this research. In Section 3, we outline the state of our current approach for supporting open document collaborations and describe two ideas for future work. Related work for our approach and the planned future work is described in Section 4. A short conclusion is given in Section 5.

2 Goals and Contributions

As a first goal, we aim to integrate the mashup approach with document models in order to support open collaboration processes from a technological perspective. We represent documents as *document mashups* of human- or software-provided content delivery, transformation or publication services. Coordination of activities is supported by user-defined rules which are part of the mashup. This should enable end-users to coordinate their document collaborations and to more easily integrate Web resources as contributions to their collaborations. As a concrete result, we provide a model of document mashups, and the design and implementation of a middleware and collaboration system.

Second, we aim to analyze communication of services in open collaboration processes. The emerging nature of the processes and the ecosystem of autonomous services make it hard for coordinators of a collaboration to keep track of the collaboration and to detect and resolve potential inconsistencies. Thus, we strive to collect data exposed by services (e.g. changes of underlying resources), mashups (e.g. service compositions) and rules (e.g. execution state) and analyze and prepare this data in order to a) provide collaboration coordinators with a visual method helping them to understand the evolution of their collaborations and used services and to b) allow detecting potential inconsistencies in the specified rules which can be caused by the autonomous nature of the services. The idea is to analyze the data in real time to be able to support the coordinator with the information needed in a certain situation. We intend to integrate these features into our document mashup tool.

3 Approach

In the following sections, we describe our document mashup approach. In addition, we describe our ideas for supporting end-user-engineering of our document mashups based on analyzing data of open collaborations.

3.1 Model and Tool for Open Collaboration Process Support

We developed a document mashup model that defines documents as artifact-centric collaboration processes [6].

Fig. 1 shows the logical collaboration structure of a document mashup supporting the authoring of a scientific publication. The example mashup includes five structural nodes, e.g. for the abstract and the main part of the publication. Nodes are coordinated by human coordinators who can select activities which should be performed on the node or its subtree.

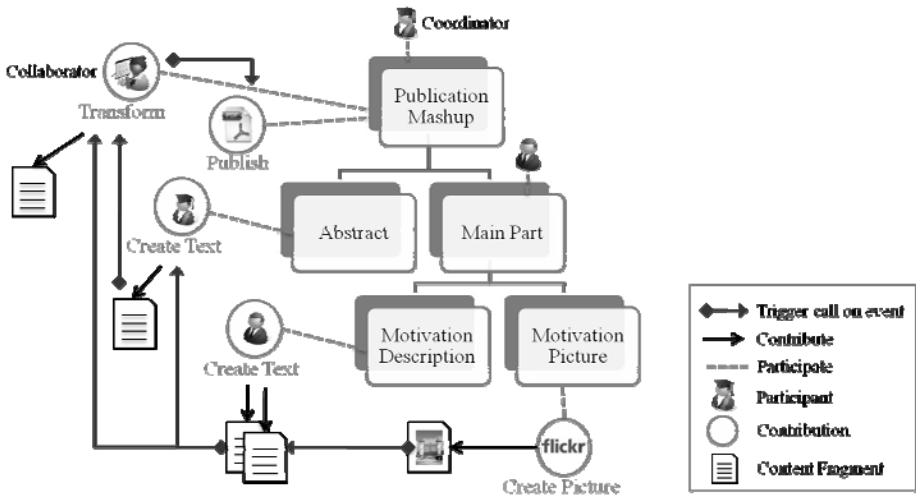


Fig. 1. Document mashup example: scientific publication

Activities of a collaboration are mapped to content creation, transformation and publication services. Examples are writing the “abstract” of a paper, proofreading or publishing the mashup to a Weblog. We assume that all services follow a uniform create-retrieve-update-delete interface. Services can be provided by humans, Web services or enterprise systems. Results of the activities are integrated into the graph structure and represent contributions to the collaboration.

Each service might expose events indicating creation, updates or deletion of its underlying content fragments. On the one hand, the document mashup is updated accordingly. On the other hand, the events are consumed and reacted upon by interaction rules which are part of the mashup and regulate and control activities. For instance in Fig. 1, the update of a picture provided by flickr triggers an update request to a human who writes a text about the picture.

During collaboration the mashup graph structure as well as included services and interaction rules can be adapted by the coordinators as required.

Based on the document mashup model, we have designed and implemented a prototype infrastructure for event-based interaction between document mashups and services which follow the REST architectural style. Furthermore, we have implemented a Web-based collaboration environment for end-users leveraging the infrastructure which supports collaborative mashup authoring and service provisioning use cases.

3.2 Monitoring and Analyzing Open Collaborations

In open collaboration processes, involved or available providers, resources and activities are changeable and heterogeneous in their nature. In addition, the collaboration itself evolves over time. The development of next-level collaboration systems thus requires making such collaborations and the evolution of the resources understandable. We aim to monitor and prepare data exposed by services as well as mashups in order to be able to analyze their behavior.

Precisely, we want to monitor creation, update and deletion events emitted by services when underlying resources change. Furthermore, we response times of certain services can be measured. We aim to use this information for the analysis of service lifecycles. When observing events of services of a specific mashup, the collaboration flow could be reconstructed. It might also be possible to detect implicit dependencies, e.g. if the update of a service always is followed by the update of another service although no rule is specified. Furthermore, influences of a service could be analyzed, e.g. if changes to it have a high impact on the whole mashup it might be a central part of the mashup.

Observing the rule execution in a document mashup could be used to find blocking rules, e.g. rules which wait for a second event and thus block the execution of other services. Furthermore, potential inconsistencies could be analyzed, for instance, if a service call was triggered by a rule but not yet answered.

When observing these behaviors for a series of mashups, frequent compositions of services, service types or providers or typical collaboration flows could be detected.

We believe that this knowledge about the evolution of services and mashups can be used to enhance collaboration platforms in various ways, from end-user support to system architecture adaption. Following, we present two ideas how the data analysis can be leveraged for better end-user support.

3.3 Data Visualization

Representing information in a visual form often is used to make complex information better understandable. Thus, we propose to develop visualizations of selected service and mashup data for end-users in order to support them in their open collaborations. This might be advantageous in various use cases.

As mashup coordinators need to select services to add to their mashup, they might be interested in the lifecycle or evolution of a service in order to make the decision about including the service. Thus, a visualization of the evolution of a service might accompany the textual or semantic description. For instance, dots on a timeline indicating

updates of the underlying resource could show how the dynamicity of a service developed over time.

In an advanced scenario, a list of services could be shown to the coordinator which seemed to be useful in similar situation or collaboration context, e.g. they were frequently used with a service which the coordinator just added to the mashup.

In order to support traceability and understandability of the collaboration, browseable visualizations could show the flow of the collaboration. A playback mechanism could be applied to show which contributions were made to the mashup at which point in time. In order to show the evolution of a mashup without presenting the contents the history flow visualization [7] could be adopted. This visualization shows how parts of a document are developed by various authors over time. This or other kinds of visualizations could also help in detecting blocking or boosting activities or showing influences of a certain event on the collaboration.

3.4 Rule Recommendations

Dependencies between services in document mashups are specified by the mashup coordinator through event-condition-action (ECA) rules. For instance, the rule engine retrieves an update event from a picture service and triggers as action a request to a text service. Input to such rules are events representing service requests and responses, resource updates, deletions and creations. Rules can be specified based on existing resources. As resources might change over time or might disappear, rules need to be adapted to this changing environment.

Large and long running mashups potentially produce large rule bases. These rule-bases need to be kept up to date and consistent by the mashup coordinator. Thus, we aim to use the service and mashup data in order to find potential inconsistencies, deadlocks or useful or needless rules. Based on the findings the system could make recommendations or indications to the mashup coordinator. In order to support rule recommendations, we first need to determine critical events and the possible inconsistencies they could cause.

An example for an inconsistent rule is, if a rule is waiting for an update event of a deleted service. This might also cause a deadlock. Contrariwise, if the action of a rule calls a deleted service, it is inconsistent. If the system detects such rules, it either removes them automatically or it notifies the mashup coordinator who then can change or delete the rules.

Rules could be recommended to the mashup coordinator if implicit dependencies between services are detected or certain rules are very common in similar mashups.

3.5 Planned Evaluation

We intend to evaluate our document mashup approach through end-user experiments in different domains, namely collaborative authoring of student project reports and project proposals. The overall goals are to evaluate the applicability of the general document mashup approach in these scenarios as well as the usability of our collaboration tool. The experiments will also be used to monitor service and mashup data for our intended extensions to the approach. In a second end-user case study experiment, we will evaluate the usefulness of these extensions.

4 Related Work

Our work adopts concepts of different fields of research like service mashups, open document collaboration, as well as software evolution analysis and visualization. We partially use approaches from these fields as entry points or additions to our work.

Mashup technologies aim at enabling end-users to compose situational applications from Web-based content and services. Several mashup research tools and products exist which offer a graphical user interface. These tools range from very specific tools which are restricted in the operations and formats they support to general purpose mashup tools. For instance, Yahoo! Pipes (<http://pipes.yahoo.com>) is a powerful tool to compose and filter data feeds. Another example is the IBM Mashup Center (<http://www-01.ibm.com/software/info/mashup-center/>) that lets users compose widgets referencing various types of services and data. Widgets can be composed both presentation-wise and through simple rules which route data between them. However, we did not find a mashup approach facilitating (document) collaboration of humans while allowing the integration of contents and services from the Web.

As regards *open document collaboration*, various research prototypes and products exist. Several Web 2.0 applications like TypeWith.me (<http://typewith.me/>) or Google Docs (<http://docs.google.com/>) allow collaborative authoring of rich text documents. However, these tools do not support coordination mechanisms or the integration of content or services from the Web. Another related technology is Google Wave (<http://wave.google.com/>). A wave is a collaboration of participants based on XML documents consisting of wavelets. This is similar to the composition of document services in a mashup. Waves might include automated robots that are comparable to our services which are not provided by humans. However, Google Wave focuses more on communication than on collaborative evolution of a document. Also, there is no way to define interaction rules based on events and to re-use wavelets in other waves.

Related work to our planned extension to visualize service and mashup data is described [8], where the evolution of data on programmableWeb.com is analyzed and visualized. The programmableWeb.com API gives access to a large database of Web API and mashup descriptions. However, this work focuses on the general evolution of the mashup ecosystem, but not on the evolution of specific mashups or services.

Furthermore, related work can be found in the field of *analysis and visualization* of software or software evolution. There exists work based on mining of different kinds of repositories, like bug databases or versioning systems [9]. There is also work on the evolution characteristics of Web services [10]. However, we did not find any approach which has collaboration processes in mind when analyzing software evolution.

With respect to analyzing processes, related work exists in the field of structured or semi-structured processes as well as the mining of processes for patterns [11], even for ad-hoc processes [12]. However, none of these approaches focus on documents as communication and collaboration vehicle.

5 Conclusion

Our work provides a unique and novel approach to support open document-centric collaboration processes. In particular, it firstly provides a model, infrastructure and

collaboration tool for document mashups to support teams in collaboratively authoring documents like scientific publications, EU-proposals or software documentation. The solution allows coordination of collaborations through representing collaboration activities as autonomous services which can be interconnected through rules reacting on events of these services. Furthermore, content and functionality services from various sources, i.e. humans, software systems or the Web, can be integrated into the evolving document.

Secondly, in future work we aim to capture and analyze data of such collaborations. This potentially allows visualizing service lifecycles or collaborations in order to help end-users understanding the evolution of services and collaborations and thus helping them to choose the services, steps or compositions they need at hand. Furthermore, coordination rules in a mashup could be checked for inconsistencies. Based on detected inconsistencies, the system could recommend rule base adaptations to the mashup coordinator. Using real time data emitted by services and mashups could improve the user experience of collaboration systems and enable new application areas involving more challenging collaboration scenarios or less experienced users.

References

1. Markus, M., Majchrzak, A., Gasser, L.: A Design Theory for Systems that Support Emergent Knowledge Processes. *MIS Quarterly* 26(3), 179–212 (2002)
2. Ceri, S., Daniel, F., Matera, M., Raffio, A.: Providing Flexible Process Support to Project-Centered Learning. *IEEE Trans. on Knowl. and Data Eng.* 21(6), 894–909 (2009)
3. Dustdar, S.: Caramba – A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams. *Distrib. Parallel Databases* 15(1), 45–66 (2004)
4. Shaw, M.: Architectural requirements for computing with coalitions of resources. In: *Proceedings of the First Working International Federation for Information Processing Conference on Software Architecture* (1999)
5. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. *IEEE Internet Computing* 12(5), 44–52 (2008)
6. Schuster, N., Zirpins, C., Tai, S., Battle, S., Heuer, N.: A Service-Oriented Approach to Document-Centric Situational Collaboration Processes. In: Reddy, S. (ed.) *WETICE*, pp. 221–226. *IEEE Computer Society, Los Alamitos* (2009)
7. Viégas, F.B., Wattenberg, M., Dave, K.: Studying cooperation and conflict between authors with history flow visualizations. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2004*, pp. 575–582. *ACM, New York* (2004)
8. Yu, S., Woodard, C.: Innovation in the Programmable Web: Characterizing the Mashup Ecosystem. In: *Service-Oriented Computing – ICSOC 2008 Workshops*, pp. 136–147 (2009)
9. Voinea, L., Telea, A.: Visual data mining and analysis of software repositories. *Computers & Graphics* 31(3), 410–428 (2007)
10. Treiber, M., Truong, H.-L., Dustdar, S.: On Analyzing Evolutionary Changes of Web Services. In: *Service-Oriented Computing – ICSOC 2008 Workshops*, pp. 284–297 (2009)
11. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering* 47(2), 237–267 (2003)
12. Truong, H.L., Dustdar, S.: Online Interaction Analysis Framework for Ad-Hoc Collaborative Processes in SOA-Based Environments. *Transactions on Petri Nets and Other Models of Concurrency* 2, 260–277 (2009)

Description-Based Mashup of Web Applications

Junxia Guo and Takehiro Tokuda

Department of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1-W8-71, Meguro, Tokyo 152-8552, Japan
{guo,tokuda}@tt.cs.titech.ac.jp

Abstract. The World Wide Web has become an almost limitless source of information. Mashup combines information or functionality from two or more existing Web sources to create a new Web page or application. Unfortunately, it is difficult for users without programming experiences to build mashups with existing Web applications, especially when they want to transfer information between Web applications which are used to build mashup. In this paper, we propose a description-based mashup approach for personal use which allows users without programming experiences to build mashup applications with existing Web applications as well as to transfer information between Web applications. This approach is based on information extraction, information transfer and functionality emulation methods. Our implementation shows that general Web applications can be used to build mashup applications easily without programming.

Keywords: Web application, description-based mashup, information transfer, information extraction.

1 Introduction

With the development of World Wide Web, more and more information has become available on the Web. The Web can be considered as an infinite source of information. In recent years, mashup applications have brought new creativity and functionality to Web applications by combining data or functionality from two or more existing Web sources. The Web sources for building mashup applications can be Web applications and Web services. Although Web services are most commonly used Web sources of mashup, many existing Web sites do not provide Web services, such as the BBC Country Profiles [4]. Thus, it is not easy to build mashup applications with existing Web applications especially for the users without programming experiences.

Methods to help users without programming experiences to build mashups easily and efficiently have been improved rapidly in recent years. Most of the existing mashup methods build mashups only with Web services and do not support the use of Web applications.

Some mashup methods can integrate parts of Web applications without Web services, but they have certain limitations. For example, C3W [2] can reuse text and functional parts of Web pages by clipping, connecting and cloning to build

mashups. However, C3W is not straightforward to general users because of a special browser to clip Web content from Web pages and a special container named DerivationPad for the clipped Web content.

Florian Daniel et al. proposed an approach in [1] that allows users to compose mashup applications using UI components. But it has a premise that component developers (IT specialists) need to componentize UI components from the Web applications and publish them on the Web.

Dapp Factory [5] can use information extracted from Web pages and RSS to build mashups, but it cannot extract dynamic Web content from Web pages such as the clock part of Localtimes.info [6]. Here we use the term static Web content to represent the content shown on the Web page that can be found directly in the HTML source file, for example text and image. And we use the term dynamic Web content to represent the content that is created by client-side scripts dynamically, which is shown on the Web page but cannot be found directly in the HTML source file.

2 Problem Statement and Research Objectives

As we mentioned in Section 1, most of the existing personal-use-oriented mashup tools for users without programming experiences cannot address two problems. One problem is that they cannot use dynamic Web content of Web pages as mashup components. The other problem is that they do not support the information transfer between the mashup components. By transfer here we mean the information extracted from mashup component A is given to mashup component B as the input. And in this paper, the mashup component is the component that is generated from existing Web applications.

In this paper, we present an approach that allows users without programming experiences to build mashup applications with any parts of existing Web applications and to transfer information between mashup components. We mainly have two objectives.

- Extract not only static Web content, but also dynamic Web content from Web applications and use the extracted results as elements to build mashups.
- Implement information transfer between mashup components extracted from Web applications without programming.

3 Research Methodology

To achieve above objectives, we need to address two main problems. One is how to wrap part(s) of a Web application into a mashup component. The other one is how to transfer information between the mashup components. The technique that we use to wrap part(s) of a Web application into a mashup component is based on information extraction and functionality emulation methods. We record the information of mashup components into a description file, which is written by notation called Mashup Component Description Language file (MCDL file) to

configure the locations and scopes of target parts of Web applications. To transfer information between mashup components, we also record the information about the data that will be delivered to other components in MCDL file. Then, we obtain the data using information extraction method and transfer the data using the program hidden in the client-side mashup application.

The components that we use to build mashup application are extracted from general Web applications. To describe the necessary information for the extraction, emulation and information transfer, we use Mashup Components Description Language. It is XML-based file, where following items are defined for each mashup component. At present we have to write each item manually. We plan to generate the following description automatically.

- id: We specify the unique name of each mashup component.
- type: We specify the type of the mashup component. It should be one of {Application, Feed, SOAP and REST}. In this paper, we focus on the "Application" type.
- startPageURL: We specify the URL of a Web page where we can submit the searching request to get the target Web page.
- inputType: We specify the type of searching method on the start page. It should be one of {LinkList (anchor list), OptionList (drop-down option list in selectbox) or InputBox (text input field) and None}. We just support these four basic input types by now. We plan to do experiments about other input types, for example radio button, as future work.
- keywordFrom: We specify how to get the searching keyword of emulation. It could be the input value from user interface, or the value transferred from other component. When the searching keyword is from other component, we use the "id" of the component to describe it.
- inputArea: We specify the path of the input Web content on the start page. If there is other content with the same "inputType" in the start page, we need to specify which one is going to be used. Users can give a value of "null" instead of the path, which will be treated as the path of *body* node by default. The path that we use here and other items are XPath-like expressions. The value of path can be gotten easily by using the tool we supply.
- targetContent: We specify the Web content to be extracted by path. Here we support multiple Web contents extraction.
- targetContentType: We specify the data type of the target Web content. It could be dynamic content (mashup, flash or function part) or static content (body, bullet, bullet list, definition list bullet, definition list, div, form, heading, input, label, text, image, link, object, select, paragraph, span, table or text area).
- newStyleforContent: We specify the new style information of target Web content in the mashup resulting page. However, the style can only be changed for static Web contents. According to the type of target Web content, different style attributes can be specified. We process this based on the attribute of different tags defined in HTML. The "newStyleforContent" information is optional information, and when it is "null", extracted Web content will be

shown in its original style. For dynamic Web contents, the extracted parts are shown in their original styles and the "newStyleforContent" value is "null".

- valueToOtherComponent: We specify the information that will be transferred to other component by path.

The above ten items describe how to get a component. A MCDL file describes a series of components from different Web sources, like a batch file. Each "component" defined in the MCDL file will be shown in one container. In the above ten items, "type", "inputType", "keywordFrom", "targetContentType" and "newStyleforContent" are selected from the predefined list. "startPageURL" is copied from browser. "inputArea", "targetContent" and "valueToOtherComponent" can be copied from our Path-Reader Tool [3], which can generate path strings of target Web content by GUI. Users can get the paths easily by clicking the target parts and then COPY the path into MCDL file. Users do not need to read the HTML source code manually.

As shown in Figure 1, the whole process consists of the following steps.

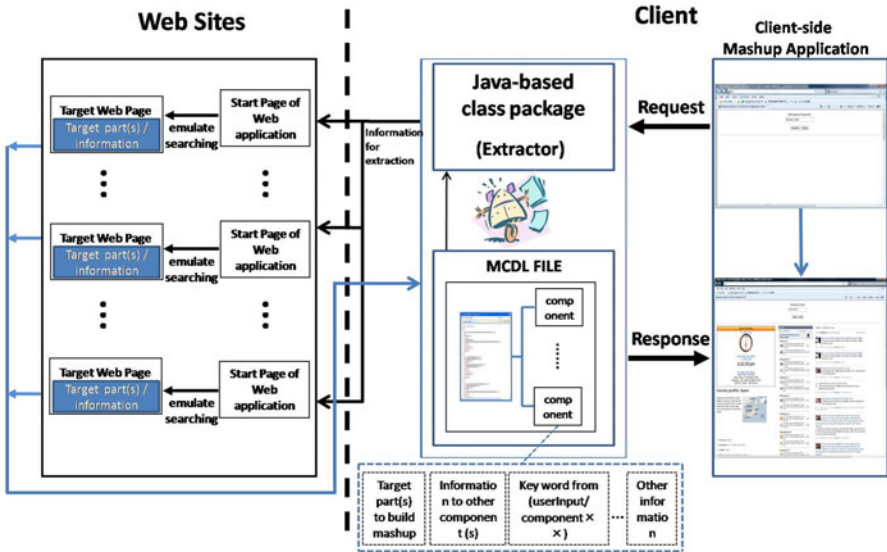


Fig. 1. Overview of Our Approach

Step 1: Users create a MCDL file which includes the information of mashup components that will be used to build mashup application and the information that will be transferred between the components. We provide an application to ease the creation of the MCDL file.

Step 2: Users send requests from the "Client-side Mashup Application" to the "Extractor".

Step 3: The "Extractor" reads the MCDL file, then according to the given keyword searches and extracts the target parts which are specified in the MCDL file.

Step 4: We integrate the extracted components and arrange the layouts as the response result.

4 Expected Contributions and Plan for Future Work

In this paper, we have presented a novel approach that allows users to integrate any parts from any Web applications without programming and allows the information transfer between mashup components to realize the consecutive queries for personal use. Our approach uses a description language (MCDL) to describe the information about mashup components which is selected to be extracted from Web applications and the information about information transfer between mashup components.

In future work, we would like to explore more flexible ways of integrating Web applications and Web services. At present we support one path to describe the location of "targetContent". We would like to support multiple paths to describe the location of "targetContent" and allow the users to decide how many paths they would like to use according to the accuracy they want. Furthermore, we would like to explore an approach to help the users without programming experiences to build mashup applications with SOAP type Web services easily and efficiently without restriction. We also would like to supply a tool to generate the MCDL file easily and automatically. In addition, the information about mashup components and the information about mashup application logic is described in one file at present. It is not convenient for reusing the components. We would like to separate the two aspects.

References

1. Daniel, F., Matera, M.: Turning Web Applications into Mashup Components: Issues, Models, and Solutions. In: Proceeding of the 9th International Conference on Web Engineering, ICWE 2009 (2009)
2. Fujima, J., Lunzer, A., Hornbak, K., Tanaka, Y.: C3W: Clipping, Connecting and Cloning for the Web. In: Proceeding of the 13th International Conference on World Wide Web (2004)
3. Guo, J., Han, H., Tokuda, T.: A New Partial Information Extraction Method for Personal Mashup Construction. In: Proceeding of the 19th European-Japanese Conference on Information Modelling and Knowledge Bases (2009)
4. BBC Country Profiles, http://news.bbc.co.uk/2/hi/country_profiles/default.stm
5. Dapp Factory, <http://www.dapper.net/>
6. Localtimes.info, <http://localtimes.info/>

iSemServ: Towards the Engineering of Intelligent Semantic-Based Services

Jabu Mtsweni^{1,2}, Elmarie Biermann², and Laurette Pretorius²

¹ SAP Research Center Pretoria/SAP Meraka UTD, Faerie Glen, Pretoria, South Africa
jabu.mtsweni@sap.com

² University of South Africa, Pretoria, South Africa
pretol@unisa.ac.za; bierman@xsinet.co.za

Abstract. The emergence of Semantic Web Services is stimulating the need for modern enterprises to efficiently and rapidly develop and deliver machine-processable and machine-interpretable value-added services in order to automate a variety of tasks on the Web. However, semantic-based services are scarcely adopted and utilised as there are few real-life examples that demonstrate the possibilities and benefits of such services. Furthermore, there is a lack of service creation frameworks and technical platforms that purport to guide and promote simple, flexible, rapid, and unified engineering of semantic-based services. In addition, current semantic service platforms do not support the construction of semantic services that are intelligent beyond the application of ontologies. In this position paper, preliminary efforts that seek to address the challenges of simplifying and speeding-up the engineering process of intelligent semantic services are presented. The goal of the work presented in this paper is about providing service providers, designers, and consumers with simple, unified, and yet simple tools that can aid in the technical implementation of intelligent semantic-based services. The main contributions envisioned from this research is a conceptual service creation framework called *iSemServ* and a technological service creation platform, which is intended to simplify and support the phases of building intelligent semantic services in an integrated manner. The proposed research adopts a quantitative approach with the main focus on model-building, prototypes, laboratory experiments, and computer-based simulations.

Keywords: Intelligent Semantic-based Services, Service Creation Framework and Platform, Ontology, Intelligent Agents.

1 Introduction

The need to expeditiously deliver value-added services on the Web is becoming a business imperative. From the current research literature, it is clear that Web Services (WS) have transformed the World Wide Web (WWW) from being a repository of data and information to a platform of distributed services [1]. However, WS only focus on the syntactical description of services; making it a challenge for services to be automatically discovered, selected, composed, invoked, and executed without or with little human intervention. Consequently, Semantic Web Services (SWS) are emerging to address such challenges [2].

In general, SWS can be defined as the convergence of Semantic Web (SW) and WS concepts in order to enable machine-processable and machine-interpretable services. SWS strives for WS that cannot only be processed by humans but by machines or software programs as well [3]. This is accomplished by leveraging WS with semantic annotations derived from domain and service ontologies to facilitate and enable automation of various aspects of WS such as discovery, selection, invocation, and execution [4, 5]. Ontologies as core to SWS refers to the conceptual knowledge representation that can be used to annotate services [6].

Although the SWS domain promises services that are machine interpretable and processable, this domain still suffers from a number of challenges. Studer [6] notes that “SWS are still searching for their ‘killer’ application”. According to Agre *et al.* [3], SWS and related applications are scarcely adopted and utilised as there are few pilot applications that fully demonstrate the possibilities and benefits of semantic-based services, and related applications. This might be further attributed to the fact that currently a combination of disconnected software tools are used to design and develop such services. This can however lead to undesirable consequences such as long service development time, high development costs, lack of services reuse, and even lack of semantics reliability and reusability.

It should be noted though that there are various software tools (e.g. WSMO studio [7] and OWL-S IDE [8]) emerging that seek to promote the development of SWS in general. However, most of these tools are not directly integrated with existing service development platforms. Moreover, these tools and platforms do not support the engineering of semantic-based services that are intelligent beyond the use of ontologies.

Furthermore, the issue of simple, rapid, and integrated engineering of semantic-based services tends to receive little exploration within the service-oriented research community. de Cesare *et al.* [9] affirm that limited work has been conducted with regard to the practical implementation of semantic-based services within the service engineering community.

In this position paper, preliminary efforts that seek to address the challenge of simplifying and speeding-up the process of engineering *intelligent semantic services (IsS)* in a unified manner are presented. In the context of the work presented in this paper, an *Intelligent semantic-based Service (IsS)* extends and leverages SWS with intelligent features adopted from intelligent agents such as autonomy (i.e. ability to act without or with little intervention and control own actions and state[10]) and pro-activity (i.e. ability to show goal-directed behavior and be able to take initiatives where possible [10]). Intelligent agents can play an important role in the adoption and usage of SWS [11], specifically in handling the automation of services’ aspects and high level of co-ordination between service functionalities on the Web. According to Blois *et al.*[12], intelligent agents can also be quite useful for the development of SWS and related applications, particularly for purposes of reasoning about ontologies linked to services. Intelligent agents can also be beneficial in providing consumers with context-aware services, eliminating the issue of irrelevant service provisioning [13]. In this paper, an IsS is defined as *a semantically-rich software unit representing some business activity, and is capable of being: (1) autonomous, (2) pro-active and reactive, (3) interoperable, (4) composable, and (5) reusable.* Since IsS augments SWS with intelligent agents’ capabilities, it should be noted that an IsS possesses all the concepts of WS, SWS and ontologies.

The remainder of this paper is structured as follows: In Section 2, research motivations and objectives that guide this study are detailed. The applicable use-case scenario is presented in Section 3. Section 4 briefly describes the solution approach adopted for the proposed study. The initial iSemServ architecture framework is briefly presented in Section 5. In Section 6; the main contributions from this study are explained. The state-of-the-art related to the semantic service engineering platform is analyzed and presented in Section 7. In Section 8, a brief description of the work done to date is given including current and future plans for the proposed study. Finally, in Section 9 the position paper is concluded with a summary.

2 Motivation and Objectives

The preliminary research work presented in this position paper is motivated by the need to minimize and simplify the technical complexities (e.g. ontology development [6]) experienced by service designers and developers when implementing and deploying semantic-services. Secondly, service designers, and providers value user-friendly, simple and yet unified technical platforms that are capable of assisting to rapidly deliver value-added services for consumption on the Web. Nevertheless, such platforms are hardly available in the SWS domain, and this in turn result in a tedious and error-prone process of engineering intelligent semantic-services, leading to services that are of low quality and performance.

The research proposal presented in this position paper is thus based on the following research question: *How could the process of engineering intelligent semantic-based service (IsS) be simplified and accelerated with the use of an open integrated platform?* In order to satisfactorily deal with the main research question, the following research objectives are identified. The first objective is to *identify and characterize a set of fundamental building blocks that make up an IsS*. The notion of IsS is emerging, and with this particular objective we attempt to understand and provide solutions as to what an IsS is, how is an IsS distinct from WS and SWS, and what components (building blocks) make up an IsS from the technical perspective. Furthermore, within this objective the goal would be to understand how the identified building blocks will be interoperated to further improve and promote simplicity when it comes to the actual development of IsS. The second objective is about *devising a service creation framework*. To engineer (i.e. model, build, and advertise) a functional IsS within an integrated platform; a number of essential components need to be identified, and analysed. The objective in this regard is to review and propose components that forms part of the service creation framework that will serve as a blue-print for engineering IsS with an integrated service creation platform. The framework will be designed based on principles of *simplicity, rapidness, complexity hiding, reusability, uniformity, and intelligence*. Thus, guiding and supporting engineering of services through simple ontologies development, intelligence wrapping of services to promote service dynamism and degree of automation, and development of semantic-based services in a cost and time-efficient manner.

The third objective is about *exploring, selecting, and/or developing software tools that can contribute to the simplification and acceleration of the process when engineering IsS* for different purposes. The *development of an integrated service creation*

platform using deliverables from the third objective forms part of objective number four. In this objective, a technical platform would be delivered for purposes of testing and validating the suggested framework. The other objective of this proposed study is the actual development of prototypes based on identified use-case scenarios.

3 Use-Case Scenario

One of the use-case scenarios that have already been identified for this study is an intelligent e-procurement scenario adapted from [6]. In this scenario, we look into demonstrating the implementation of a web-based application composed of intelligent semantic services developed by retail suppliers. This is purported to also illustrate how intelligent semantic services can minimize the hurdles experienced by small scale traders in the rural areas of South Africa who travel long distances on regular basis back-and-forth to urban cities to procure stock for their retail shops. In this scenario, suppliers develop and make available intelligent semantic services using our suggested service creation platform. These services are capable of co-ordinating with intelligent agents in assisting small scale traders in the rural areas to seamlessly “shop around online” using their mobile devices or desktop computers. For example: a small scale trader can use his/her device to simply draw up a digital shopping list. After that, the intelligent agent(s) cooperating with semantic services take over, and with little human intervention shops around from available suppliers and considers users requirements.

Once the “shopping” is completed, a preliminary basket with a variety of deals or quotations is sent back to the user for confirmation. The users chooses the preferred option, and the agent(s) in co-ordinating with semantic-services residing at relevant suppliers places an order, and forwards the final invoice to the shop owner. Depending on the business model, once the payment has been received, the supplier interacts with the shop owner to finalize delivery of ordered goods. The main aspects that are to be addressed in our work, and are implicitly illustrated in this scenario are that of simplifying the process of modelling semantic service functionality using techniques such as the Eclipse Modelling Framework (EMF) [14]; seamless transformation of service model into partial semantic service functionality using converters such as UML2Java; manual and guided extension of service functionalities from partial translations, automatic generation of syntactic service descriptions, seamless translation of service descriptions into partial ontological specifications (e.g. WSDL2WSMO [15]), visual and/or textual building of domain ontologies for suppliers, and service ontologies for different services (e.g. product catalogue services) using for example Web Service Modelling Toolkit (WSMT [16]); visual and automatic annotation of services; and lastly intelligence wrapping of semantic services.

4 Solution Approach

The proposed research adopts a quantitative research approach with the main focus on model-building, prototypes, laboratory experiments, and computer-based simulations. The primary research methodology adopted for this study is that of modelling, where

a service creation framework/model is devised and tested based on functional and non-functional characteristics as deduced from literature and use-case scenario as briefly described in Section 3. The service creation framework (see Section 5), is developed as a base for the technical platform that could make it possible to simplify the process of engineering semantic services.

Using at least two use-case scenarios; domain specific services and applications demonstrating the practicality and utility of the service creation framework and the integrated service creation platform will be produced via the prototyping research approach. Laboratory experiments will be conducted with an objective to gain deep insight into the service creation framework and to note the effects of the framework and the platform when developing IsS as compared to currently existing models and platforms. Through the experiments; simplicity (ease-of-use), usefulness, and uniformity of the service creation framework and the unified engineering platform will be tested and measured through simulations and usability test with sampled service developers. In the proposed research, computer-based simulations are applied mainly for demonstrating philosophical properties of an IsS such as *intelligence* and *context-awareness*.

5 Preliminary Solutions

In this section, a preliminary *iSemServ* architecture framework is presented as one possible approach for accelerating and simplifying the process of engineering intelligent semantic-based services that can autonomously accomplish Web-based tasks on behalf of its users. The architecture-framework is structured into the *service layer*, *semantic layer*, and *intelligence layer*. The service layer deals with the activities related to the actual design and development of syntactic services.

In the semantic layer, ontologies are generated using WSMO [7] and semantic annotations are performed using various techniques adopted from WSDL-S [17] and ServFace¹. As the main contribution from the main research of this position paper, the intelligence layer enables semantically-annotated services to be intelligent, by enabling semantic services to be autonomous, proactive, context-aware, and highly automatic.

The architecture framework follows a SOA-based and an Agent-based approach. Furthermore, the *iSemServ* architecture framework is based on these functional requirements as per research literature and challenges addressed in the overall study (see Section 1). In the following subsections, the main constituent components of the *iSemServ* architecture framework are briefly explained.

5.1 Visual Modeller

This component deals with service modelling. In order to satisfy the requirements of *simplification* and *rapidness*, VUML [18], a view-based design method using UML is adopted for the *iSemServ*. This is to enable users to model the service using mainly notations and diagrams. VUML is essential for *iSemServ* as it can ease the service engineering process by automatically generating relevant service source code once a visual service model has been created.

¹ <http://www.servface.eu>

5.2 Service Composer/Importer

It is possible for the required IsS to be constructed from existing WS or SWS as provided by external providers. Hence, the *service composer/importer* component deals with the activities related to the composition of external services and/or importing of single services for use during the development process. External services can be imported or composed using service descriptions (e.g. WSDL) from external registries. The *flexibility and rapidness* requirements of the framework are partially satisfied by this component.

5.3 Service Editor

This component deals with the actual coding and extensions of the functionalities related to the required service using either the VUML generated code or service descriptions of imported or composed services. Eclipse plug-ins such as WSDL2Java could be used for reverse-engineering purposes, with regards to the imported services. In order to satisfy the *complexity hiding* requirement, other aspects of service development such as service description are hidden from the developer as service descriptions are generated automatically.

5.4 Service Descriptor

This component is mainly about formally describing constructed syntactic services. Although this process is envisioned to be automatic and semi-transparent to the user, it is worth explaining. This process is mainly about syntactically describing the functional and non-functional properties of the constructed service. These descriptions will be automatically generated from the service source code as generated and made available during the service construction phase. Java2WSDL, an Eclipse plug-in can be used for automatically generation service description file.

5.5 Ontology Editor

This component deals mainly with the construction of service, domain, and context ontologies for the service produced in the service layer.

5.6 Agent Platform

This component deals specifically with the development and/or reuse of agents that can envelop semantic-based services to produce IsS. The agent platform also facilitates the mapping of agents with semantic-service capabilities, service and domain ontologies, and context knowledge. This is done to simplify and automate the process of reasoning and processing ontologies services and agents interaction.

5.7 Intelligence Wrapper

The *intelligence wrapper* component is responsible for wrapping a developed semantic-based service with *intelligence* to materialize a concrete IsS. This activity is semi-automatic and is achieved through the use of the agent platform component called JADE [19].

5.8 IsS Validation, Deployment, and Publication

Depending on the complexity of the constructed IsS, the validation process could be undertaken to ensure that the service behaves and executes as desired. The final stages of the proposed *iSemServ* architecture framework deal with the internal deployment and publication of a functional IsS into an environment where users can access and execute available intelligent semantic-based services.

6 Main Contributions

The main scientific contributions from the proposed research to the field of Service and Web engineering respectively is a multi-layered service creation framework and an open integrated service creation platform that purport to simplify and accelerate the process of engineering intelligent semantic-based services by introducing visual service design and development tools, user-friendly and integrated ontology development tools, seamless intelligence wrapping of the service, and finally improve level of automation in services through the use of software agents.

On the other hand, the social contributions are on aiding both traditional and modern enterprises in designing and developing intelligent semantic-based services and applications that could promote their competitiveness and growth on the Web, in a cost-effective manner. Moreover, from the proposed study, the following research contributions are also envisaged: (1) provide a suitable test-environment for semantic-based services and related applications, (2) promote the uptake of semantically-enriched services and applications by supporting semantics reusability and interoperability in heterogeneous semantic-based applications, and (3) minimize the time and costs required for engineering semantic-based services and related applications. The limitations that are envisaged in this study are that of limited evaluation and validation. Intelligent semantic-services would not be deployed in a real-life environment, but mostly tested in a lab environment.

7 Related Work

ODE-SWS, a SWS development environment by Corcho *et al.* [20] focuses on developing SWS in a language independent approach. Various semantic description languages can be used within this platform. The framework is integrated within WebODE, an ontology engineering workbench, responsible for exporting provided ontology into other ontology languages [21]. WebODE is considered in the semantic layer of the proposed *iSemServ* framework for flexibility and interoperability purposes. On the down side, ODE-SWS environment does not consider intelligence aspects for services as included in our proposed research work

One of the frameworks that claim to be the first toward SWS engineering is called INFRAWEB [22]. It focuses on constructing semantics data for existing and new services. INFRAWEB is made up of different units (i.e. SWS creation, monitoring, selection, discovery, composition, and conversion), which are essential to the actual development and implementation of semantic-based services. INFRAWEB is limited in a sense that it is tightly bound to a specific ontology language (i.e. WSMO); hence it is not flexible and does not use software agents for achieving high degree of automation.

Internet Reasoning Service (IRS-III) is a comprehensive framework and a platform for creating WSMO-based SWS [23]. IRS-III is promoted as a development framework for SWS. According to Domingue *et al.* [24], the main goal of IRS-III is to support capability-based discovery and invocation of semantic services. Its other main role is to mediate between service providers and service consumers using ontologies to further enhance interoperability and collaboration. However, IRS-III does not necessarily provide an environment where new semantic-based services can be visually engineered based on users' requirements.

Finally, one of the recent research endeavours that is closely related to the work of this paper is that of Srinivasan, Paolucci, and Sycara [25]. The authors proposed and developed an integrated development environment (IDE) called CMU's OWL-S Development Environment (CODE) for developing, deploying, and consuming semantic-based services. CODE adopts and extends existing WS and ontology engineering tools (i.e. OWL-S editor, WSDL2OWL-S converter, and so on) in order to support developers with the process of developing, deploying, and consuming semantic-based services [26]. It is embedded within the Eclipse environment and is purely based on OWL-S and Java; and follows a multi-methodological approach by applying both code-driven and model-driven methodologies. The CODE platform supports various SWS activities such as discovery, invocation, and execution. However, on the down side, the platform does not cater for flexibility, interoperability, and intelligence, which is one of the main contributions from this paper. It should be noted that some of the components within the proposed *iSemServ* framework are adopted and extended from the CODE platform.

8 Current Status and Future Directions

The theoretical investigation on technologies such as WS, SWS, ontologies (e.g. WSMO and OWL-S), and service engineering has been completed. The deliverables from the investigation are three draft literature review chapters. The task of identifying and characterizing the basic building blocks that constitute an IsS has been completed. Currently, preliminary experiments are being conducted to operationalize and analyse the identified basic building blocks. Furthermore, the components that are envisaged to form part of the service creation framework are being identified, analysed, and selected.

A preliminary *iSemServ* architecture framework is presented in this paper (see Section 5). The framework has been theoretically evaluated through comparison analysis, against other service creation frameworks such as INFRAWEBBS [3], CODE [25], and IRS-III [24]. Above that, the service creation platform will then be developed based on the *iSemServ* framework, and use case scenarios will be refined for prototypical demonstrations and experimentation purposes.

9 Conclusion

The emergence of machine-interpretable and machine-processable Web services promises much improved service provisioning and consumption. However, without

robust, simple, and mature platforms; the wide spread adoption and application of semantic-based services could always be limited. Furthermore, enterprises facing challenges of software costs, development costs, and technical skills will also lag behind. Hence, it is essential for the research community to timely provide guidelines, techniques, user-friendly tools, and open platforms that can enable businesses and Web users to produce and consume semantic-based services in a real-world setting. In this position paper, a proposal was presented on how these challenges can be addressed. The main contributions envisaged from the proposed study are a service creation framework and a technological service creation platform intended to simplify and support the phases of building a functional intelligent semantic services and related applications in an integrated manner.

Acknowledgment

The support of SAP Research Centre Pretoria and SAP Meraka UTD (CSIR) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the authors and not necessarily to be attributed to the companies mentioned in this acknowledgement.

References

1. Alonso, A., Casati, F., Kuno, H., Machiraju, V.: *Web Services: concepts, architectures, applications*. Springer, Heidelberg (2004)
2. Lu, J., Zhang, G., Ruan, D. (eds.): *E-Service Intelligence: Methodologies, Technologies and Applications*, vol. 37. Springer, Heidelberg (2007)
3. Agre, G., Marinova, Z., Pariente, T., Micsik, A.: *Towards Semantic Web service engineering Workshop on service matchmaking and resource retrieval in the semantic Web (SMRR 2007)*, CEUR (2007)
4. Corcho, O., Silvestre, L., Benjamins, R., Bas, J.L., Bellido, S.: *Personal ebanking solutions based on semantic web services*. *Studies in Computational Intelligence (SCI)* 37, 287–305 (2007)
5. Bensaber, D.A., Malki, M.: *Development of semantic web services: model driven approach*. In: *8th international conference on new technologies in distributed systems*. ACM, Lyon (2008)
6. Studer, R., Grimm, S., Abecker, A. (eds.): *Semantic web services: concepts, technologies, and applications*. Springer, Berlin (2007)
7. Dimitrov, M., Simov, A., Momtchev, V., Konstantinov, M.: *WSMO Studio - a semantic web services modelling environment for WSMO*. In: *4th European Conference on the Semantic Web: research and applications*. Springer, Innsbruck (2007)
8. Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., Senanayake, R.: *The OWL-S editor – a development tool for semantic web Services*, *The Semantic Web: Research and Applications*, pp. 78–92. Springer, Heidelberg (2005)
9. de Cesare, S., Holland, G., Holtmann, C., Lycett, M.: *Semantic-based systems development*. In: *22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion OOPSLA 2007*. ACM, Montréal (2007)
10. Jennings, K., Wooldridge, M.: *Software Agents*. *IEEE Review*, 17-20 (1996)

11. Garcia-Sanchez, F., Valencia-Garcia, R., Martinez-Bejar, R., Fernandez-Breis, J.T.: An ontology, intelligent agent-based framework for the provision of semantic web services. *Expert Systems with Applications* 36, 3167–3187 (2009)
12. Blois, M., Escobar, M., Choren, R.: Using agents and ontologies for application development on the semantic web. *Journal of Brazilian Computer Society* 13, 35–44 (2007)
13. Aziz, Z., Anumba, C., Ruikar, D., Carrillo, P., Bouchlaghem, D.: Semantic web based services for intelligent mobile construction collaboration. *ITcon* 9, 367–369 (2004)
14. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: *Eclipse Modeling Framework*. Addison Wesley Professional, Reading (2003)
15. El Bouhissi, H., Malki, M., Bouchiha, D.: A reverse engineering approach for the Web Service Modeling Ontology specifications. In: *Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008)*, Cap Esterel, pp. 819–823 (2008)
16. Kerrigan, M., Mocan, A., Tanler, M., Fensel, D.: The Web Service Modeling Toolkit (WSMT)- an integrated development environment for Semantic Web Services. In: *4th European Conference on the Semantic Web: Research and Applications*. Springer, Innsbruck (2007)
17. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Verma, A.S.K.: *Web Service Semantics - WSDL-S. W3C* (2005)
18. Nassar, M., Anwar, A., Ebersold, S., Elasri, B., Coulette, B., Kriouile, A.: Code generation in VUML profile: A model driven approach. In: *IEEE/ACS International Conference on Computer Systems and Applications*, pp. 412–419. IEEE, Rabat (2009)
19. Bouhissi, H.E., Malki, M., Bouchiha, D.: Towards WSMO ontology specification from existing Web Services (2006)
20. Corcho, O., Gomez-Perez, A., Fernandez-Lopez, M., Lama, M.: ODE-SWS: A semantic web service development environment. In: *1st International Workshop on Semantic Web and Databases (SWDB 2003)*, Berlin, Germany (2003)
21. WebODE: WebODE ontology engineering platform. Vol. 2009 (2003)
22. Nern, J., Agre, G., Atanasova, T., Marinova, Z., Micsik, A., Kovács, L., Saarela, J., Westkaemper, T.: INFRAWEBs semantic Web service development on the base of knowledge management layer. *International Journal on Information Theories and Applications (IJITA)* 13, 161–168 (2006)
23. Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: a broker for Semantic Web Services based applications. 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. 201–214. Springer, Heidelberg (2006)
24. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: IRS-III: A broker-based approach to semantic Web services. *Web Semantics: Science, Services and Agents on the World Wide Web* 6, 109–132 (2008)
25. Srinivasan, N., Paolucci, M., Sycara, K.: CODE: a development environment for OWL-S Web services. Robotics Institute, Carnegie Mellon University (2005)
26. Srinivasan, N., Paolucci, M., Sycara, K.: Semantic Web Service Discovery in the OWL-S IDE. In: *39th Annual Hawaii International Conference on System Sciences*, vol. 6, pp. 109–118. IEEE Computer Society, Kauai (2006)

Sustaining High-Availability and Quality of Web Services

Erbin Lim and Philippe Thiran

PRECISE Research Centre
Faculty of Computer Science
University of Namur

Abstract. In this work, we study how to sustain the high availability and quality of Web services by using communities of Web services. Availability is the probability that a Web service is in functioning condition at a specific time. Communities of Web services gather Web services that provide the same functionality, but not necessary with the same quality. Within this framework, an available Web service can be selected or even substituted by another one when it fails. In this research, we aim at investigating how to manage the community to sustain not only the high availability but also to guarantee the quality of service expected by the user.

1 Introduction

Web services are intensively used for developing loosely-coupled, inter-enterprise business processes. Ideally, a Web service is required to accept all the requests of the users. However, a Web service can get busy with an overloaded demand and thus, cannot offer a good quality of service to the number of users that exceeds its capacity. This situation would cause inefficiency in offering service quality and consequently lead to a drop in the satisfaction value of the Web service that is assigned by the users.

1.1 Availability

Availability is the probability that a Web service is in a functioning condition at a specific time. Sustaining high availability of a Web service is a challenging but crucial issue. For a Web service, failing to respond with an acceptable quality of service would cause a negative satisfaction of the users, and hence, drop the future number of requests.

Traditionally, the most common solution to sustain availability is the use of *replicas* which are managed according to specific replication strategies and are used when the original Web service fails [10, 11, 16]. Another solution is to group similarly-functional Web services into *communities* to improve their overall performance and availability [17]. A community of Web services *selects* an available Web service or even *substitutes* the selected Web services when it fails.

1.2 Quality of Service

Replicas or Web services of a community can differ with their quality of service. Ideally, replication, selection or substitution should guarantee the quality expected by the user, and thus his satisfaction. The user satisfaction must be maintained so that a user request can be rejected if the offering service quality is not sufficient. However, rejecting a user request can also impact the user satisfaction. Hence, we need to measure and analyze the tradeoff between availability and user satisfaction.

Quality of Service can be divided into two categories [1], the first category includes objective parameters while the second category includes subjective parameters. Objective parameters are defined as parameters that can be measured quantitatively. For example, *number of requests*, *number of successfully handled requests*, and *uptime* can be considered as objective parameters; while *user-perceived availability*, *attractiveness* can be considered as subjective parameters.

1.3 Scope of the Thesis

The objective of the thesis is to sustain high-availability and quality of services. In the thesis, we use the communities of Web services which are proposed by [17]. The main contribution of this paper lies on the QoS issues in the selection and substitution process within the communities.

2 Community of Web Services

Benatallah et al. [5] define a community as a collection of Web services with a common functionality, although these Web services have distinct non-functional properties like different providers and different QoS parameters. In this work, we consider a community as a means for providing a common description of a desired functionality (e.g., FlightBooking) without explicitly referring to any concrete Web service (e.g., EKFlightBooking) that will implement this functionality at run-time [4].

Figure 1 represents the architecture of communities of Web services. The components of this architecture include Web services providers (WSP), Universal Description Discovery and Integration (UDDI) registries, and communities. A community is dynamic by nature. It is established and dismantled according to specific scenarios and protocols. UDDI registries receive advertisements of Web services from providers.

Two communities of Web services are shown in the figure. They could for example offer AirfareQuotation and HotelBooking functionalities, respectively. A *Master Web service* (MWS) always leads a community. The other Web services are denoted as *slaves* (SWS). Slave Web services offer the same functionality but with possibly different qualities of services.

One of the responsibilities of the master Web service is to attract Web services to sign up in the community it heads using rewards. As a result, the master Web-service interacts with the UDDI registries on a regular basis, so it gets to know the latest changes in the content of these UDDI registries such as advertisements of new Web services. Additional responsibilities of the master Web service are:

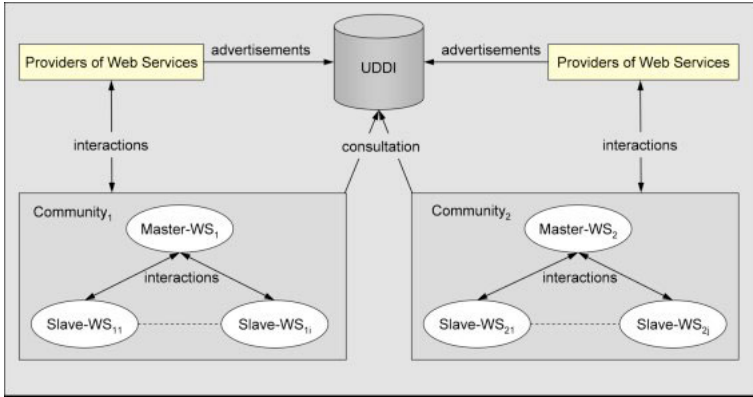


Fig. 1. Community of Web Services

- to accept or invite new Web services to be members of the community and to exclude or reject existing Web services;
- to nominate the Web service that will receive the user request.

The MWS exists in a community as a figurehead to control the SWS and the management of requests. The selection of the MWS is a conceptual one.

3 Problem Definition

The research question can be formulated as follows:

How can we sustain high-availability and the expected quality of Web services using communities of Web services?

This research will look into the research issues when using similarly functional Web services with different QoS. The problems can be divided into two different categories:

- *Member management*, which involves all problems related to the evaluation of the number of SWS and their quality;
- *User request management*, which includes all problems related to the nomination (selection or substitution) of the SWS that will receive the user request.

3.1 Member Management

In a community of Web services, different SWS provide concrete Web services with different QoS. The number and quality of SWS within the community can directly impact the number of participation requests and the user satisfaction.

This drives the MWS to regularly check the *size* of the community and the quality of services provided by the SWS. As a result, a MWS oversees all the events in a community such as arrival of new Web services, departure of some Web services, sanctions on Web services due to unavailability, misbehavior, etc.

If the MWS notices that first, the number of SWS is less than a certain threshold and second, the number of requests that arrive from users over a certain period of time is high than another threshold, the MWS should invite new Web services to join the community [13]. Conversely, when the number of SWS in a community is high but the number of user requests is low, this could lead the MWS to eject some Web services with a low level of QoS and availability from the community and to invite other Web services with better performance to join the community.

In a community, SWS compete to increase their participation. MWS nominates the available SWS that can answer efficiently user request. Thus, an excellent user satisfaction can drive the MWS to prefer certain SWS. This can lead to two undesirable situations:

- Overloading of the efficient SWS that could decrease the offering service quality;
- Leaving of SWS as their participation rate in the community is insufficient.

3.2 User Request Management

A user request is handled by the MWS that selects the most efficient SWS that will answer it. Sometimes the selected SWS can be overloaded, and consequently becomes unavailable. If the best selected SWS becomes unavailable, MWS would substitute it by asking the second best SWS in terms of similar QoS, user satisfaction (and so on).

Finding the most efficient SWS for a user request is not a trivial task. Users have different requirements in terms of the expected QoS. Moreover, they rate differently their satisfactions about the received quality and responsiveness.

Moreover, a MWS must reserve his most efficient SWS for certain users. Indeed, all user requests do not necessary require a high QoS. Hence, these requests can be handled by low efficient SWS so that the most efficient SWS are kept available for future requests.

4 Research Methodology

The first part of the research will introduce the framework of a community of Web services. This also includes the definition and evaluation of QoS metrics of each individual SWS. This provides the basis to define the QoS metrics for the community as a whole.

The next step will be to define the member management of the community. This process involves creating mechanisms and heuristics to determine how to manage the members in the community. The mechanisms are used to determine the optimal number and quality of members within a community.

The user request management comes next. Mechanisms are to be used to select the best available SWS to serve a user request. In line with the goal of achieving high availability, it is also crucial to define a substitution mechanism to handle the situation where a SWS fails. Both the selection and substitution mechanisms use the QoS metrics defined in the first part.

An experiment or prototype would be critical in determining the effectiveness of the proposed mechanisms. This can be achieved by running experiments to adjust the parameters in the proposed mechanisms to determine if they achieve the desired result.

5 Related Work

The use of community of Web services to sustain High-availability was mentioned in [17]. But the authors do not mention the need for sustaining a certain level of QoS. To determine the size and type of a community, the level of QoS of the community needs to be calculated. Similarities between calculating the QoS of a community of Web services can be found with calculating the level of QoS in a composition of Web services [7,6].

The selection process when a request is handled by the community is similar to the selection process within a composition of Web services [18,3,12]. The substitution process is the main contributing factor to sustaining high-availability and QoS in Web services. Similar work has also been done in the area of web services discovery in [15]. In [2,14], the authors select the web service based on QoS metrics and client preferences.

In [8], the authors present a tool that sustains the service quality within workflow management systems. They use mathematical models to meet specific goals for performance and availability due to outages of individual servers. This is similar to what we are proposing because we aim to sustain the quality of service of the web service even if the original service provider were to fail.

Most literature consider *availability* and QoS parameters as a deterministic value. However, as mentioned earlier, we consider *availability* and various QoS parameters to be based on a probability. This approach was also mentioned in [9].

6 Work in Progress

To date, we have reviewed the literature of community of Services and its QoS metrics. We have also started to define the QoS metrics to be used in our research. We have compared the framework of a community of Web services with current more researched frameworks, for example, a composite of Web services.

7 Conclusion

In this paper we have outlined the project proposal for a method of sustaining high availability and quality of Web services. This research creates a framework to allow communities of Web services to achieve this goal.

We believe that our research contributes to the area of Web engineering by providing a new approach to sustain high availability and quality of Web services.

References

1. Zeitham, V.A., Parasuraman, A., Berry, L.L.: Quality counts in service too. *Business Horizons* (May-June 1985)
2. Al-Masri, E., Mahmoud, Q.H.: Discovering the best web service. In: *WWW 2007* (May 2007)
3. Ardagna, D., Pernici, B.: Global and local qos guarantee in web service selection. In: *First International Workshop on Business Processes and Services* (2005)
4. Bentahar, J., Maamar, Z., Benslimane, D., Thiran, P.: An argumentation framework for communities of web services. *Intelligent Systems* 22(6), 75–83 (2007)
5. Sheng, Q.Z., Benatallah, B., Duman, M.: The self-serv environment for web services composition. *IEEE Internet Computing*, 40–48 (January-February 2003)
6. Canfora, G., Di Penta, M., Esposito, R., Luisa Villani, M.: A lightweight approach for qos-aware service composition. In: *Proc. 2nd International Conference on Service Oriented Computing (ICSOC 2004)* (2004)
7. Cardoso, J., Miller, J., Sheth, A., Arnold, J.: Modeling quality of service for workflows and web service processes. *VLDB Journal* (2002)
8. Gillmann, M., Weikum, G., Konner, W.: Workflow management with service quality guarantees. In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 228–239 (2002)
9. Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the qos of web-services-based workflows. *Science Direct* (2007)
10. Osrael, J., Frohofer, L., Goeschka, K.: What service replication middleware can learn from object replication middleware. In: *Proceedings of the Workshop on Middleware for Service Oriented Computing held in conjunction with The ACM/IFIP/USENIX 7th International Middleware Conference*, pp. 18–23 (2006)
11. Salas, J., Perez-Sorrosal, F., Patino-Martinez, M., Jimenez-Peris, R.: Ws-replication: A framework for highly available web services. In: *Proceedings of the 15th International World Wide Web Conference (WWW 2006)*, pp. 357–366 (2006)
12. Liu, Y., Ngu, A.H.H., Zeng, L.: Qos computation and policing in dynamic web service selection. *ACM, New York* (2004) 1-58113-912-9/04/0005
13. Maamar, Z., Subramanian, S., Thiran, P., Benslimane, D., Bentahar, J.: An approach to engineer communities of web services - concepts, architecture, operation, and deployment. *International Journal of E-Business Research* 5(4) (2009)
14. Maximilien, E.M., Singh, M.P.: *Toward autonomic web services trust and selection* (2004)
15. Ran, S.: *A model for web services discovery with qos*. ACM, New York (2003)
16. Ye, X., Shen, Y.: A middleware for replicated web services. In: *Proceedings of 2005 IEEE International Conference on Web Services, ICWS 2005*, p. 638 (July 2005)
17. Benslimane, D., Maamar, Z., Sheng, Q.Z.: Sustaining web services high-availability using communities. In: *2008 Third International Conference on Availability, Reliability and Security* (2008)
18. Zeng, L., Benatallah, B., Dumas, M., Kalaganam, J., Sheng, Q.Z.: Quality driven web services composition. In: *Proceedings of the 12th International Conference on World Wide Web*, pp. 411–421 (2003)

Client-Side Adaptation: An Approach Based in Reutilization Using Transversal Models

Sergio Firmenich¹, Silvia Gordillo^{1,2}, Gustavo Rossi¹, and Marco Winckler³

¹ LIFIA, Facultad de Informática,
Universidad Nacional de La Plata and Conicet Argentina

² CiCPBA

{sergio.firmenich,gordillo,Gustavo}@lifia.info.unlp.edu.ar

³ IRIT, Université Paul Sabatier, France
winckler@irit.fr

Abstract. Improving navigability in Web applications is essential for applications success. Our research aims to improve the user experience by applying novel techniques such as concern-sensitive navigation. Concern-sensitive navigation allows enriching Web pages with content related to the context in which they are accessed. In previous works, we showed how this technique can be applied during the development process; at present we are working in a client-side adaptation approach to apply concern-sensitive navigation to existing applications. This approach is open to other adaptation kinds which are illustrated in this position paper. We also outline a set of tools we are developing to simplify the process of adaptation.

Keywords: Concern-sensitive navigation, User experience, Client-side adaptation.

1 Introduction and Motivation

The Web is in constant evolution. Social sites like Facebook or LinkedIn, interactive encyclopedias like Wikipedia or multimedia servers such as Flickr have changed our way to access information. As a final result of this evolution, we could say that the user experience has been improved in a remarkable way, though we still need to struggle to make popular applications more “adaptable”. A constraint in Web engineering approaches to adaptation is that the user’s information which is used to adapt an application is many times restricted to a single application’s boundary. In this way, very often the user is not considered as accessing many applications at the same time, and this has as a result that the user’s experience is not accordingly improved.

In previous papers we presented concern-sensitive navigation (CSN) [2] as a conceptual tool to improve navigation by enriching or adapting a link’s target node according to the user’s concern in the link’s source node. In this way, when the user navigates from node A to node B, his concern in A is used to adapt B. On the other hand, when the user navigates from C to B, B is adapted using the concern in C; therefore the contents and links in B slightly change according to the navigation source. If both nodes A and B belong to the same application, we say that it is an *intra-application* adaptation,

otherwise, it is an *inter-application* adaptation. We are interested in client-side (CS) adaptation of existing applications; this means that the adaptation is not only performed in the client, but that adaptation engineers might be third parties (potential users instead of developers of the adapted application).

This position paper presents a novel realization of CS adaptations. First, we focus on CSN adaptations, achieving reuse of adaptation code by using transversal models between different applications. We outlined our tool to ease the development process.

The structure of the paper is as follows. In Section 2, we explain how we build CSN at the CS, and how these adaptations can be reused. In section 3 we show new types of adaptations that can be realized at the CS. In section 4 we conclude and present some ideas for further research.

2 Concern-Sensitive Navigation: A Client-Side Approach

In [2] we presented a novel realization of CSN, by performing the adaptation in the CS of existing applications, using scripts (not necessary developed by the designers of the adapted application). The adaptation process is facilitated by a weaving engine; in [2] we used the Grease Monkey (GM)¹ weaving engine.

To make these scripts stable we extensively used the concept of Modding Interface (MI) [1]. A MI is a high-level application model aimed to create an abstract layer over the DOM. A Modding Interface contains concepts and concepts' properties to abstract the real DOM elements. Two files are used to define a Modding Interface: an OWL to define the model and a XSLT for mapping elements between model and the DOM. In this way, the scripts access DOM elements indirectly by manipulating concept instances. In [1] it is the application developers' responsibility to publish the MI, as well as to maintain the matching between models and underlying DOM elements. Our approach uses this MI as an abstraction mechanism to achieve reusable adaptations and easier maintenance. Note that a developer who wants to create an adaptation must define the MI for the target application. In [2] we presented our MI development tool; later we show how we generalized the idea and propose improvements for the tool.

2.1 Applications Families and Reusable Adaptations

In our research we have seen that many applications may share similar MI models, therefore we introduced the concept of application family. An application family is a set of applications sharing several model's concepts. Since adaptation scripts access the model's concepts and not the DOM, we can reuse an adaptation across several applications (in the same family). In Figure 1 we show an application family example. Note that the same model is shared between Amazon.com and BarnesAndNoble.com. At left, we show a MI model containing the concept "Result" with two properties: the resulting product's name, and the product's price. At right, we show how each property is mapped to different DOM's elements in each application, consequently an adaptation accessing to "Result" instances could be reused in both applications.

¹ GraseMonkey: <https://addons.mozilla.org/en-US/firefox/addon/748>

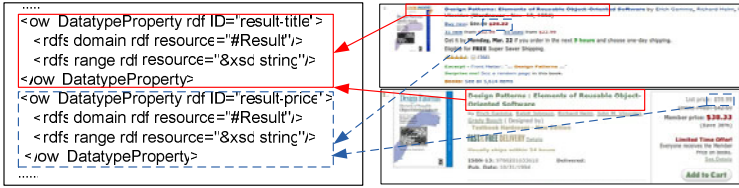


Fig. 1. Search results in different e-commerce applications

2.2 Developing CSN Adaptations

Developing a CSN adaptation using this approach is feasible but can be difficult even for an experienced developer. The first step to solve this problem is to raise the abstraction level in MI construction process. As part of this research we have developed a tool, which helps developers to define MI's concepts by highlighting DOM elements. In this way, developers are relieved from low-level technical issues. The tool also supports the definition of a CSN adaptation by selecting the source and target application, and choosing what information from the source node will be used.

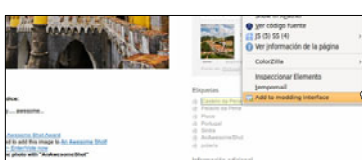


Fig. 2.1. Adding concepts to the Modding Interface

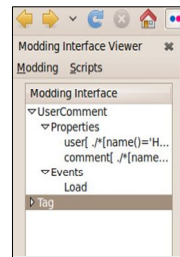
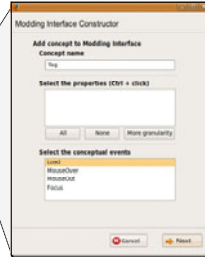


Fig. 2.2. MI Viewer

In Figure 2 we show the first step in this process, Figure 2.1 shows how the MI's concepts are defined, while in Figure 2.2 the concepts already defined are presented.

When the concepts are defined, our tool supports the CSN adaptations construction by selecting source and target nodes. In this process, the adaptation developer must specify which concept instances will be stored to use them in the target application.

3 Generalizing Client-Side Adaptation

CSN adaptations are a useful way to improve the user experience. We have done experiments with users, and they have appreciated the improvements achieved with CSN enhancements. In broader terms, we could say that CS adaptation is gaining followers. Users are really interested in adapting existing Web applications. GM is really used around the world, some scripts have been downloaded million of times. However, not all possible CS adaptations are related with the navigational path followed by the user. Moreover, the typical GM scripts under utilizes the CS adaptations possibilities. With this in mind, we present our classification of kinds of CS adaptations:

- Static adaptations: this adaptation kind will be applied every time that the application’s page loads. This is a very important adaptation because with them we can achieve, for example, an improvement in the application’s accessibility. An adaptation could hide instances of concepts or reorganize UI elements to do more efficient the reader’s work.
- CSN adaptations: adaptations which improves the user’s experience by adapting applications considering the user’s concern. Several examples of this adaptation kind can be found in [2].
- Navigational history adaptations: adaptations based in the user’s navigation. This kind of adaptation was initially described in History-Based [3]. However, when this technique is used dynamically at the CS, the possibilities increase considerably since we can use the navigation history through many applications and not only into an application’s scope.
- Task-aware adaptations: adaptations which take into account the user’s current activity, e.g. while he works with several applications. For example, in Figure 3, we can see a common Web usage scenario enriched with task-aware adaptation. In Figure 3.1 the user is reading a Wikipedia’s article. After that, the user opens a new tab and searches for “adaptivity” in Google. At the end, in Figure 3.3, when the user returns to Wikipedia, the page has been adapted by adding a link to the Wikipedia’s article about “adaptivity”.

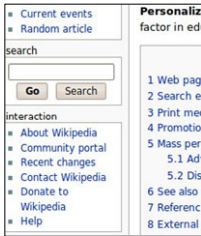


Fig. 3.1. User is reading a Wikipedia’s article

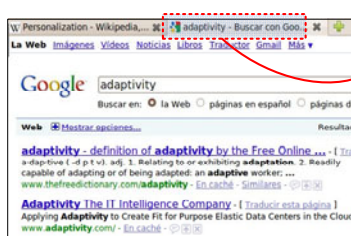


Fig. 3.2. User opens another tab to search for “Adaptivity” in Google



Fig. 3.3. Wikipedia is adapted by a context-aware adaptation

Note that this kind of adaptations can’t be achieved easily using well-known Web development methodologies. In comparison with CS adaptations, their adaptation approaches have two disadvantages: 1) the adaptations are restricted by application developers, 2) users are not taken into account as users of other applications.

Security issues are important here; with our CS approach, we can know all about the user’s activity. Obviously, this must be done in a secure way and we must be sure that the information shared by applications is only used in the adaptation code. A way to do it is using credentials when an adaption does a request which require privileges. Performance is important too. In our tests we didn’t perceive a lost in this, after all, and for now, the adaptations are only JavaScript code which is very usual nowadays.

3.1 Improving Support for Client-Side Adaptations

The adaptation kinds presented above imply new features not contemplated in our previous work and which GM can't offer. Because GM scripts don't have privileges, they can't know when another page is loaded, and they are not able to share data among different applications. On other hand, when GM is used, it is the developer who must coordinate different tasks: create and export models, create and export adaptations scripts, install scripts in GM and store the MI in an accessible site. Therefore, we choose to develop a tool which will work both with GM and a GM script which will be the connection between the adaptations and our tool. We aim to improve this process by making the tool responsible of: (1) constructing and storing transversal models, (2) responding to adaptation requests about context and data.

The following list enumerates new features of the extended tool we are building:

- Features to make the information accessible in different context: saving and restoring contextual data is very important when the adaptation is developed in an inter-application way.
- Support for task-aware adaptations: these adaptations will strongly depend on context changes; for example, the user is accessing a specific application, or even with more granularity, the user is accessing a specific functionality in an application. The tool must have an API to register changes in context elements and create corresponding events for these changes. In this way, the adaptations that are pending of these changes can be performed when the event occurs.

4 Concluding Remarks and Further Works

Providing Web users with appropriate information and functionalities according with his actual concern or preferences is an interesting research topic. The adaptation approaches of mature Web methodologies have usually two main restrictions: (1) adaptations are defined by application developers; (2) generally, adaptations are restricted to a single application, so users are not considered as users of other applications. Our approach solves these restrictions by enabling users to adapt applications in a reusable and straightforward way. As a proof of concept, we have outlined our client-side realization of CSN presented in [2]. We are now working in a broader classification of client-side adaptation kinds and we are extending our tool to support them.

References

1. Diaz, O., Arellano, C., Iturrioz, J.: Layman tuning of websites: facing change resilience. In: Proceeding of WWW 2008 Conference, Beijing, pp. 127–128. ACM, New York (2008)
2. Firmenich, S., Rossi, G., Urbieto, M., Gordillo, S., Challiol, C., Nanard, J., Nanard, M., Araujo, J.: Engineering Concern-Sensitive Navigation Structures. In: Concepts, tools and examples, JWE (accepted 2010)
3. Brusilovsky, P.: Adaptive Navigation Support. In: The Adaptive Web: Methods and Strategies of Web Personalization, pp. 263–290. Springer, Heidelberg (2007)

QuEF (Quality Evaluation Framework) for Model-Driven Web Methodologies

F.J. Domínguez-Mayo, M.J. Escalona, and M. Mejías

Department of Computer Languages and Systems, University of Seville, Spain
{fjdominguez,mjescalona,risoto}@us.es

Abstract. QuEF (Quality Evaluation Framework) is an environment for the assessment of Model-Driven Web Engineering (MDWE) methodologies. This approach is oriented towards the evaluation, through objective measures, of the quality of MDWE methodologies in a specific environment. Given the high number of methodologies available and proposed over recent years, it has become necessary to define objective evaluation tools to enable organizations to improve their methodological environment and help designers of web methodologies to design new effective and efficient tools, processes and techniques. Since methodologies are constantly evolving, the need may arise not only to evaluate the quality but also to find out how it can be improved and how the quality improvement process could be optimized in order to reduce costs.

Keywords: Model-Driven Web Engineering, Quality, Methodologies, Approaches, Model-Driven Engineering.

1 Introduction, Problem and Motivation

In recent years, the growing interest in the internet has led to the generation of a high number of Model-Driven Web Engineering (MDWE) approaches which offer a frame of reference for the Web environment [4]. On the other hand, there are a high number of approaches without standard consensus, [9], [7], [3] a lack in the use of standards, and scarcity of both practical experience and tool support. In the face of this situation, an important need to assess the quality of existing methodologies arises. In this paper, QuEF (Quality Evaluation Framework), an environment for the quality evaluation of Model-Driven Web methodologies, is proposed.

This doctoral consortium paper is organized into the following sections. In Section 2, the aims and objectives of the proposed research is presented. Section 3 presents the research methodology, the work carried out to date, and the tentative plans for future work. In Section 4, the main contributions of the research to Web Engineering are explained.

2 Aims and Objectives of the Proposed Research

The main goal of this research is to lay the basis of an environment for the assessment of MDWE methodologies that facilitates a quality assessment and continuous improvement

of different methodological proposals under some objective criteria in order to improve these methodologies although the framework could be extended to other specific area or domain. Hence, there is a need for the suitable design of MDWE methodologies and effective tools. To this end, our work concentrates on evaluating and comparing existing proposals although the framework could be extended in the future to other areas. On the other hand, the software development process has a direct influence on the quality and cost of software development and, therefore, the use of an MDWE methodology and its influence on the final product quality must be considered.

In this work, an approach, or Methodology, is a Model-Driven proposal for the development of web applications. It may provide a set of guidelines, techniques, processes and/or tools for the structuring of specifications, which are expressed as models. Only web modelling approaches which are based on MDA in the framework are considered. In addition, a framework in this work is a basic conceptual structure composed of a set of elements used to evaluate, in this case, MDWE methodologies although it could be extended to other areas or domains. QuEF is a quality evaluation framework with a set of elements based on existing literature as shown in Figure 1, where four components for the evaluation of the quality of MDWE methodologies can be seen:

- *Quality Model component*: this includes the basis for the specification of quality requirements with the purpose of evaluating quality. It specifies each element and its purposes.
- *Thesaurus & Glossary component*: this includes all the necessary terms to improve the standardization of the access channel and communication between users of different MDWE methodologies.
- *Approach Characteristic Template component*: this includes the description templates of the input methodology characteristics to be evaluated. It depends on the Quality Model description.
- *Quality Evaluation Process component*: this includes the definition and specification for carrying out the quality evaluation process.

3 Research Methodology, Work Carried Out to Date, and Tentative Plans for Future Work

An action research methodology is going to be followed. Therefore, the first step then is to examine the idea carefully in the light of the means available, this work is divided into several phases in order to develop a first version of the thesis project: Analysis of the State of the Art, Identification, Definition of Concepts, Framework Design and Framework Web 2.0 Tool Support implementation. In next step, more fact-finding about the situation is required. If this first period of planning is successful, two items emerge: namely, “an overall plan” of how to reach the objective and secondly, a decision in regard to the first step of action. Usually this planning has also somewhat modified the original idea. The next step is ‘composed of a circle of planning, executing, and reconnaissance or fact finding for the purpose of evaluating the results of the second step, and preparing the rational basis for planning the third step, and for perhaps modifying again the overall plan’. This research methodology is oriented to problem-solving in social and organizational settings, in our case are the MDWE research community.

3.1 The Quality Model Component

The Quality Model in QuEF is a set of characteristics, subcharacteristics and metrics, quality factors, quality attributes and the relationships between them, which provides the basis for specifying quality requirements and evaluating quality in a specific domain (in this case, MDWE). In Figure 2, the Quality Model metamodel is shown together with the relations between the different elements in the Quality model, whose elements have already been described and explained in previously published papers [1], [2].

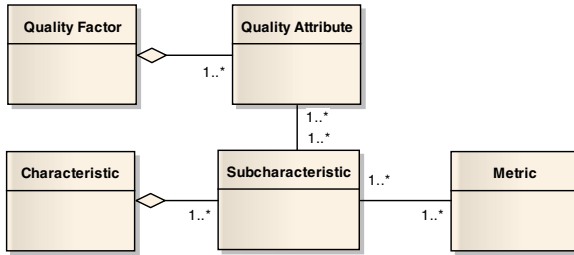


Fig. 1. Quality Model metamodel

In order to define a *Quality Model*, the metamodel contains *association links* between the *subcharacteristics* and the *quality attributes*. These *association links* represent the dependencies between *subcharacteristics* and *quality attributes*. They show quality attributes which are affected by *subcharacteristics* or the areas of the methodology that will be significantly affected if the approach is changed. *Association links* may be based on proven and real-world experience. The impact of each *subcharacteristic* on *quality attributes* must be demonstrated and the requirements determined by real case-study applications to a number of real projects. This should be supplemented by reference to published literature. Furthermore, *subcharacteristics* have to define quantitative or qualitative *metrics* which may be used to measure each *subcharacteristic*. Otherwise it would be necessary to define a set of indicators from reference values which may be set to a prescribed state based on the results of measuring or on the occurrence of a specified condition. Hence, a quality factor has various quality attributes and a characteristic has various subcharacteristics, as is shown in Figure 2. A weight is used to define the importance of a metric in the value of a subcharacteristic. Similarly, a weight is also used to define the importance of a quality attribute in the value of a quality factor and the influence importance in association links between subcharacteristics and quality attributes. The tasks for the definition of the Quality Model have already been described in previous papers, and are: The identification quality factors; The identification of quality attributes for each quality factor; The identification of characteristics; The identification of subcharacteristics and metrics for each characteristic; and, The specification of association links between the subcharacteristics and the quality factors.

3.2 The Thesaurus and Glossary Component

An important element for QuEF is the thesaurus component. A thesaurus is a list containing the "terms" used to represent concepts, themes or contents of documents in order to standardize the terminology which improves the access channel and communication between users of different MDWE methodologies. We consider it necessary to carry out a standardization of terminology to improve the access channel for communication on MDWE. A set of concepts for MDWE methodologies is currently being described and related.

3.3 The Approach Characteristic Template Component Based on the Quality Model

Templates with subcharacteristics and metrics for each characteristic are based on the Quality Model and are used to describe an input methodology. These templates are used as input to QuEF. They are analyzed in the evaluation process and compared with the model quality of the Quality Model component. Templates for MDE, Web Modelling, Tool Support and Maturity have already been developed.

3.4 The Quality Evaluation Process Component

The Quality Evaluation Process component contrasts the information from each input approach template with information from the Quality Model. The main purpose of this evaluation is to identify tradeoffs and sensitivity points of the methodology under study. The idea is to determine which aspect needs to be improved on MDWE methodology. A simple evaluation is made with the MS Excel considering weights for metrics, subcharacteristics and quality attributes. However, fuzzy logic is currently being considered in order to improve this evaluation process.

4 Main Contributions of the Research to Web Engineering

A quality environment for the assessment of MDWE methodologies is proposed. With regards to the contributions obtained from this research, a framework is required for the improvement of current proposals or even for the development of a new methodology. We consider that the use of QuEF will enhance the quality of products, processes and techniques of approaches. Furthermore, it could be used for optimizing a continuous quality improvement since we can select and improve the minimal number of subcharacteristics which have a majority number of influences in quality attributes using the matrix of influences. Furthermore, in various papers we have evaluated subcharacteristics related with MDE, Maturity and Tool Support of the NDT Methodology, which are required for the measurement of the value of MDWE methodologies in order to be able to assess and improve their Functionality, Usability, Portability and Reliability. Other characteristics such as Web Modelling and Quality Assurance have yet to be described and other quality factors such as Maintainability remain to be described. Therefore the use of QuEF can improve the efficiency and effectiveness of MDWE methodologies since this approach evaluation helps one understand the strengths and weaknesses of a methodology. On the other hand, QuEF could be extended to other areas or domains. Besides, different approaches as NDT, UWE,

WebML and OOHDMM methodology are currently being evaluated. Further characteristics and quality factors have yet to be developed. To this end, Microsoft Excel is currently being employed as a first prototype although a new Web 2.0 tool remains to be developed.

Acknowledgements

This research has been supported by the project QSimTest (TIN2007-67843-C06_03) and by the RePRIS project of the Ministerio de Educación y Ciencia (TIN2007-30391-E), Spain.

References

1. Domínguez-Mayo, F.J., Escalona, M.J., Mejías, M., Torres, A.H.: Towards A Quality Evaluation Framework for Model-Driven Web Engineering Methodologies. In: Proceedings of the 6th International Conference on Web Information Systems and Technologies (WEBIST 2010), vol. 2(1), pp. 191–194 (2010)
2. Domínguez-Mayo, F.J., Escalona, M.J., Mejías, M., Ramos, I., Fernández, L.: A Quality Evaluation Framework for MDWE Methodologies. In: Proceedings of the eighteen International Conference on Software Quality Management (SQM 2010), vol. 1(1), pp. 171–184 (2010)
3. Escalona, M.J., Torres, J., Mejías, M., Gutiérrez, J.J., Villadiego, D.: The treatment of navigation in Web Engineering. In: Advances in Engineering Software, vol. 38, pp. 267–282. Elsevier Ltd., England (2007)
4. Escalona, M.J., Koch, N.: Requirements Engineering for Web Applications – A comparative study. *Journal of Web Engineering* 2(3), 193–212 (2004)
5. IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology (1990)
6. ISO- International Organization for Standardization, ISO/IEC 9126-1, <http://www.iso.org>
7. Kroiß, C., Koch, N.: UWE Metamodel and Profile, User Guide and Reference. Technical Report 0802. Programming and Software Engineering Unit (PST), Institute for Informatics. Ludwig-Maximilians-Universität München, Germany (2008)
8. Mohagheghi, P., Aagedal, J.: Evaluating Quality in Model-Driven Engineering. In: International Workshop on Modeling in Software Engineering (MISE 2007). IEEE Computer Society, Los Alamitos (2007)
9. Schwinger, W., Retschitzegger, W., Schauerhuber, A., Kappel, G., Wimmer, M., Pröll, B., Cachero Castro, C., Casteleyn, S., De Troyer, O., Fraternali, P., Garrigos, I., Garzotto, F., Ginige, A., Houben, G.-J., Koch, N., Moreno, N., Pastor, O., Paolini, P., Pelechano Ferragud, V., Rossi, G., Schwabe, D., Tisi, M., Vallecillo, A., van der Sluijs, Zhang, G.: A survey on web modeling approaches for ubiquitous web applications. *International Journal of web Information Systems* 4(3), 234–305 (2008)

Consistent Cache Maintenance for Database Driven Websites

Paweł Leszczyński and Krzysztof Stencel

Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Chopina 12/18, 87-100 Toruń
{pawel.leszczynski, stencel}@mat.umk.pl
<http://www.mat.umk.pl>

Abstract. Since databases became bottlenecks of modern web applications, several techniques of caching data have been proposed. Caching data helps to resolve a problem of a database scalability, however it introduces an additional problem of a consistency maintenance. We are going to develop an existing caching model for an automatic consistency maintenance of the cached data and data stored in a database. We prepare for proposing a dependency graph which provides a mapper between update statements in a relational database and cached objects. When an update on a database is performed the graph allows detecting cached objects which have to be invalidated in order to preserve the consistency of the cache and the data source. We believe that the constructed model allows keeping the number of invalidations as low as possible. We also provide some benchmarks which prove that our method is efficient when compared with other approaches.

Keywords: web applications, scalability, database caching, cache consistency.

1 Introduction


WEB 2.0 applications are data-intensive. As the number of users grows, the backend database rapidly becomes a bottleneck. Thus, various data caching techniques have been developed. Most e-commerce applications have high *browse-to-buy* ratios [6], which means that the read operations are dominant. We are going to develop a novel method of running the cache of a data-intensive WEB 2.0 application. The goal of our research is to minimize the number of objects residing in a cache which must be invalidated after an update to the stored data.

The problem of maintaining cached data up to date is difficult and known for a long time from a materialized view maintenance. However we are going to solve it for WEB 2.0 applications which allows us to define additional model assumptions. As stated earlier read operations are dominant. We remark that web applications use only basic SQL statements and we can limit SQL syntax complexity in our model. We can also identify most of performed statements

before an application deployment and find the frequent ones to cache them and improve performance. These assumptions clearly simplify the problem and give a chance of solving it although general solution may be still unknown.

2 Motivating Example—A Community Forum Application

Let us now consider a community forum application as an example. From the database's point of view the website consists of three views which are database extensive: listing forums, listing topics and listing posts. Figure 1 contains a view of a real forum application to show which data is needed when displaying list of topics.





TOPICS	REPLIES	LAST POST
 [Tutorial] How to install by batt - Sat Jan 26, 2008 3:36 am	20	by g_tsch9  Wed Mar 04, 2009 6:15 am

Fig. 1. The figure shows a single line from a list of visible topics. Each line contains: a topic name which is the first post name, an owner of the topic and the date it was created, a post count, and an information about the last post.

Performing a query to display it each time the website is loaded is too extensive when avoiding a redundancy in the database. This can be done by providing extra counters, an information about the last post, etc. However this introduces problem of maintaining counters by an application logic and does not scale well.

The solution is to use a cache objects in memory. As on the figure 1, each object can contain topic data, post counter and information about the last post added. However, the data is updated by the users who add new posts. Whenever this happens, many post counters have to be recalculated. The desired property of a cache is to recompute only those counters which have become invalid and possibly nothing else. The idea of caching objects and storing them in a *key-value* storage is widely applied by mechanisms like: *Memcached* [12], *Velocity* [13] or *Redis* [14]. However all of them lack automatic consistency maintenance and require the programmer to take care of cached data to be up to date.

In this paper we show a novel solution to the problem of consistent cache for WEB 2.0 applications. As the result, of this research, web engineers will get a scalable web cache which fully prevents storing stale data.

3 Related Work

Before the WEB 2.0 era several techniques of caching whole websites have been proposed and surveyed in [11][15]. They construct mappers between URLs and database statements. However this approach loses efficiency in WEB 2.0 applications since the same HTML can be used on many pages and it does not make sense to invalidate the whole pages.

When caching single queries it is hard to discover similarities and differences between the queries and their result, for example when they return the same result in a different order. The other option is storing table fragments, which can be understood as storing data sets in caches and allowing them to be queried [5,6,16]. This however lacks count operations which are nowadays common.

Authors of [2,3,4] present model based on statements' templates which is similar to ours. However they cannot handle join or aggregation in select statements. Our approach is based on a graph which edges determine the impact of the update statements on the cached data. The idea of the graph representation has been first presented in [7,8,9]. The vertexes of the graph represent instances of update statements and cached data objects. However nowadays most webpages are personalized and number of data objects has increased and multiplied by a number of application users. According to this the graph size grows rapidly and the method becomes impractical. In our approach vertexes of the dependency graph represent classes of statements instead of instances, which reduces the number of vertexes.

4 Research

According to the Brewer's theorem [1] it is impossible to provide consistency, availability and partition tolerance at once in a single system. A distributed system can satisfy any two of these guarantees at the same time, but not all three. Our research can be defined as finding the best trade-off between those three parameters in the terms of WEB 2.0 applications.

The planned research is divided into two parts. First an algorithm for maintaining data consistency needs to be developed and then tested in multiple benchmarks to evaluate its efficiency. The first part is mainly accomplished. We construct the dependency graph between update statements and select statements used for creating cached objects. Our construction does not use concrete statements but groups them into classes. This is efficient since in WEB 2.0 era the same statements are executed several times having only different parameters. This allows us to reduce a number of graph vertexes.

An example of the dependency graph for a simple forum application can be seen on a figure 2. The graph consists of vertexes of update statements U_1, U_2, \dots , vertexes of database columns in tables, vertexes of select statements S_1, S_2, \dots and vertex of cached object *topic* created by them. When an update is performed we check which object vertexes may need invalidation and then resolve object instances which are not up to date. The graph edges determine dependencies between vertexes and let us trace the impact of an update statement.

The second part of the research involves providing plausible results of the constructed model carried out in different benchmarks: RUBiS [17], RUBBoS [18] and TPC-W [19]. These all provide us with web application and client emulator which visits websites and performs operation according to a predefined transition probabilities.

We are going to examine applications in different modes: without caching, caching with a defined *time-to-live* of the cached objects and caching with the invalidation management based on the dependency graph.

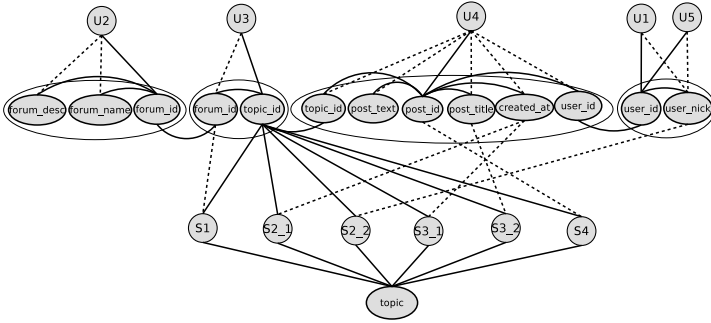


Fig. 2. A graph created for a topic objects in the community forum example

When inventing the dependency graph and invalidation algorithm we wanted to estimate the efficiency of the constructed algorithm. Therefore we have implemented the algorithm in a RUBiS benchmark. A figure 3 display achieved results. Our consistency preserving model reduces up to 54% of performed queries. This proves the presented model to be worth further development and performing more benchmarks. To do so we are heading towards creating transparent database proxy which implements our algorithm and detects statements which can be computed from the cache. The construction of the proxy gives us the possibility to carry out experiments on different benchmarks since it does not involve any changes in an application.

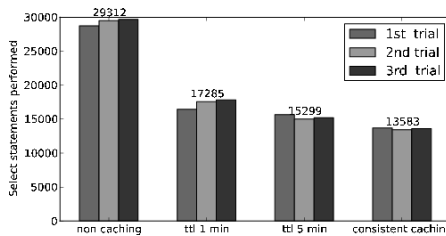


Fig. 3. The comparison of different caching techniques. The numbers indicate average count of select statements for each technique.

Additionally, when compared to materialized views, our model does not assume anything of the data stored in cached objects. As a result it allows to contain HTML code fragments. The research can be extended to providing intelligent mapper between SQL statements and HTML code fragments generated

according to them. Such a mapper is strongly desired in modern WEB 2.0 applications where website consists of many separate components displayed under different url addresses.

References

1. Brewer, E.A.: Towards robust distributed systems. In: Annual ACM Symposium on Principles of Distributed Computing (2000)
2. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable query result caching for web applications. Proceedings of the VLDB Endowment archive 1(1), 550–561 (2008)
3. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable Consistency Management for Web Database Caches. Computer Science Technical Reports, School of Computer Science, Carnegie Mellon University (2006)
4. Manjhi, A., Gibbons, P.B., Ailamaki, A., Garrod, C., Maggs, B., Mowry, T.C., Olston, C., Tomasic, A., Yu, H.: Invalidation Clues for Database Scalability Services. In: Proceedings of the 23rd International Conference on Data Engineering (2006)
5. Altnel, M., Bornhvd, C., Krishnamurthy, S., Mohan, C., Pirahesh, H., Reinwald, B.: Cache tables: Paving the way for an adaptive database cache. In: Proc. VLDB 2003, pp. 718–729 (2003)
6. Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Woo, H., Lindsay, B., Naughton, J.: Middle-tier database caching for e-business. In: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 600–611 (2002)
7. Iyengar, A., Challenger, J., Dias, D., Dantzig, P.: High-Performance Web Site Design Techniques. IEEE Internet Computing (4), 17–26 (2000)
8. Challenger, J., Dantzig, P., Iyengar, A., Squillante, M.S., Zhang, L.: Efficiently Serving Dynamic Data at Highly Accessed Web Sites. IEEE/ACM Transactions on Networking (12), 233–246 (2004)
9. Challenger, J., Iyengar, A., Dantzig, P.: A Scalable System for Consistently Caching Dynamic Web Data (1999)
10. Zhao, W., Schulzrinne, H.: DotSlash: Providing Dynamic Scalability to Web Applications with On-demand Distributed Query Result Caching, Computer Science Technical Reports, Columbia University (2005)
11. Katsaros, D., Manolopoulos, Y.: Cache management for Web-powered databases. Web-Powered Databases, 201–242 (2002)
12. Memcached, Danga Interactive, <http://www.danga.com/memcached/>
13. Velocity, <http://code.msdn.microsoft.com/velocity>
14. Redis, <http://code.google.com/p/redis/>
15. Li, W., et al.: CachePortal II: Acceleration of very large scale data center-hosted database-driven web applications. In: Proc. VLDB (2003)
16. Amiri, K., Tewari, R.: DBProxy: A Dynamic Data Cache for Web Applications. In: Proc. ICDE, pp. 821–831 (2003)
17. RUBiS (Rice University Bidding System), <http://rubis.ow2.org/>
18. RUBBoS (Bulletin Board Benchmark), <http://jmob.ow2.org/rubbos.html>
19. TPC-W (Transactional Web e-Commerce benchmark), <http://www.tpc.org>

Improvements of Webometrics by Using Sentiment Analysis for Better Accessibility of the Web

Radek Malinský and Ivan Jelínek

Department of Computer Science and Engineering, Faculty of Electrical Engineering
Czech Technical University in Prague,

Karlovo náměstí 13, 121 35 Prague, Czech republic

{malinrad, jelinek}@fel.cvut.cz

<http://webing.felk.cvut.cz>

Abstract. The paper discusses the webometric model for effective acquisition of relevant information from the web that would better separate the useful data from the useless. Our research emphasis has been placed on techniques that would better reflect the semantic content of single pages. Webometrics is purely a quantitative approach to the web, which can be enhanced by qualitative methods and thereby allows us to expand the possibilities of a study problem. Sentiment analysis may be used as a qualitative complement to quantitative approach. This analysis provides a technique of sophisticated analysis of sentences using mathematical and statistical methods and linguistic analysis of text. Extension of the webometric techniques of sentiment analysis methods leads up to a better machine understanding of a web page and its overall semantic meaning. It can be assumed that the designed model will reduce the irrelevant web search results and thereby facilitate user access to the information on the web. The introductory part of the paper explains the concept of the sentiment analysis and the basic functional background of the webometric techniques.

Keywords: Webometrics, Sentiment analysis, Semantics, Web 2.0.

1 Introduction

Until recently, webometric research has been focused on the evaluation of pages without any previous knowledge. Many of these quantitative studies have focused on hyperlinks. For example, research focused on the issue, whether interlinking between local government bodies in Finland follows a strong geographic, or rather a geopolitical pattern [3], or a correlation between universities' research and their presence on the web exists [1]. The quality of pages is determined by bibliometric and informetric approaches [4, 9]. These approaches are very simple and do not reflect the semantic content of single pages.

The volume of information on the web has been growing at a very high rate and becoming a network of heterogeneous data, this makes things difficult to

find and is therefore not almost useful. It is necessary to design suitable metric for such volume of information, which would reflect semantic content of pages in the better way. However, the primary task is to define criteria for the quality assessment of found information, and “distance” between relevant information. On this basis, we can design rules for evaluation of semantic content in relation to user’s queries.

One of the options for more accurate comprehension of semantic information is to use a sophisticated analysis of sentences using mathematical and statistical methods and linguistic analysis of text (called Sentiment Analysis [6]).

The knowledge gained will be useful for algorithm design to facilitate user access to the information on the web, and to obtain the public opinion on specific issues above all. The algorithm will be based on current webometric techniques [1], [2], [4], [10], [11] extended by sentiment analysis [6], [7], [8].

2 Webometrics

Webometrics is a scientific discipline that studies the quantitative aspects of information resources and their use. Webometrics is based on informetric and bibliometric methods [9]. The information sources that are studied by webometrics are web documents. According to narrow definition, webometrics encompass two basic categories: web link structure analysis and the evaluation of search engines using informetric methods.

Current search engines are based on the idea of citation analysis and impact factor [4], [9]. This strategy especially excels in queries to a specific case, but when you enter a complex query, it returns a large number of irrelevant links and does not reflect the semantic content of single pages.

Webometrics is used in scientific communication, especially for citation counts (scientometrics) and for analysis of electronic journals and universities websites [5], [9]. Non-academic parts of the web are rather heterogeneous, therefore; it is difficult, but not impossible to find common webometric characteristics.

In recent years, there have been efforts to apply webometric techniques to non-academic parts of the web. Google PageRank [4] or Ranking of World Universities [1] are good examples of successful projects. Webometrics in the area of Web 2.0¹ has been mainly used to find information on blogs [2] and for trend detection [10]. In the beginning, it was a simple context analysis based on counting how often searched word was mentioned online. Current research is focused on sophisticated analysis of sentences [7], [8], which aims to determine the polarity of the text and the attitude of a writer with respect to some topic.

Several types of techniques have been designed during the gradual development of webometrics: web link structure analysis, web page content analysis, web usage analysis, web technology analysis, etc. The first two mentioned techniques are the most important for our research at this point.

¹ The term “Web 2.0” is a designation for the developmental stage of the web, where static content is replaced by space for creating and sharing content.

2.1 Link Analysis

Link Analysis [1], [3], [5] is used to determine the impact of information, documents, pages or organizations on the web, or to determine the online relationships between them. Input data is gathered either from commercial search engines or from research crawlers. The evaluated output can be reported as inlink counts or as a network diagram with nodes representing web sites and arrows between the nodes representing the links between them.

PageRank [4] used by the Google internet search engine is the most reliable and effective link analysis algorithm. The algorithm evaluates the relevance of search result pages by the number of inlinks and by the quality of source of these inlinks. The quality of inlinks has a much higher significance than the number of inlinks.

2.2 Web Mention Analysis

Web Mention Analysis [11] is the evaluation of the “web impact” of documents or ideas by counting how often they are mentioned online. Assessment is a combination of several types of methods:

Web Mentions. This method determines the popularity estimation of ideas or documents using reported hit count estimates from commercial search engines. The hit count estimates are the numbers reported by search engines in their result pages as the estimated maximum number of matching pages.

Content Analysis. Content Analysis is a systematic separation into categories, such as the document type, national origins, industrial sector, etc. It is used to reduce the irrelevant search results that have nothing to do with the specific category.

Hyperlink Analysis. Hyperlink Analysis is based on the extraction of information from URLs. This is very useful in Content Analysis because URL extraction can provide information such as the geographic spread or the type of organization that is interested in the document.

3 Sentiment Analysis

Sentiment Analysis or Opinion Mining enables us to automatically detect opinions from structured but also unstructured data. This involves several research areas such as natural language processing, computational linguistic and text mining. The research in this field originated from the demand of commercial companies, who wanted to know public opinion on price, quality and other features of their products [6].

Before the massive spread of the Internet, companies had been gaining customer feedback via phone, email surveys or interviews. It was very slow, expensive and annoying for some customers. With the Internet, companies may be

able to gain feedback from comments in e-shops, blogs, customer reviews, social networks, etc. These methods of obtaining information are very fast and thanks to that the company that has just launched a major advertising campaign may gain quick public feedback on its impact.

The main goal of sentiment analysis is to identify positive/negative polarity of the text and recognize a subjective/objective impression of the text [8]. As a sub task, analysis is capable to determine the attitude of a writer with respect to some topic. The attitude may express affective state of the author when writing or intended emotional effect which author wishes to present to the reader.

4 Webometrics + Sentiment Analysis

Current research on Sentiment Analysis has made great strides and is currently used in many industrial applications, such as the analysis of the comments of goods in e-shops [6] or the videos on YouTube [7]. We built our research on the achievements in this area [6], [7], [8], [12], which we want to use for improvements of webometrics for better accessibility of the web. Webometrics is purely a quantitative approach to the web, which can be enhanced by qualitative methods and thereby allows us to expand the possibilities of a study problem [11].

Original webometric techniques [1], [2], [4], [10], [11] improved searching and provided trend detection. However, they are not able to distinguish a polarity of the text, its semantic meaning and the emotional state of the author. Extension of the webometric techniques of sentiment analysis methods leads up to gaining insights into a public opinion with respect to some topic and to a better machine understanding of the text. Better understanding of the content on the web site could have a significant impact on the quality of site evaluation.

The aim of our work is to propose a theoretical model for a more effective way of obtaining relevant information from the web. The model builds on webometrics and starts from the idea that almost any text can be machine-recognized. This idea is supported by current research in sentiment analysis [6], [7], [8], which aims at sophisticated analysis of sentences using mathematical and statistical methods and linguistic analysis of text.

The model provides complete content analysis of crawled pages. The analysis is performed in three phases. The first phase; Webometric Web Mention Analysis, determines some basic information about the document such as the document type, geographic spread, type of organization that is interested in the document etc. and chooses keywords or entire phrases which represent it. The second phase; Sentiment Analysis, determines the polarity and impression of the text and evaluates the selected key words and phrases. The third phase; Webometric Link Analysis, determines the impact of an analyzed document. The next part of the model is a proper citation index that stores the output of the content analysis and allows quick and easy searches with regard to users' queries.

Web 2.0 can serve as a data source for our research because it contains vast amount of information from many different users. Thanks to blogs and social networks; its popularity has been constantly rising and the volume of information has been growing at a very high rate.

5 Work Plan

Our research should be improving the content representation of a web page and its semantic meaning. There are several important steps leading to the completion of our work. 1. Design an appropriate assessing method of analyzed information that would better separate the useful data from the useless. 2. Propose an appropriate citation index structure and algorithm for the information obtained from it. 3. Create and implement an experimental system for gathering and processing data from Web 2.0, which will be able to verify the theoretical assumptions.

Furthermore, the final model will be used to develop an algorithm which improves the quality of search engines on the principle of webometrics. This will lead to a better machine understanding of user queries and thereby the reduction of irrelevant web search results.

Acknowledgments. This research has been supported by MSMT under research program No. 6840770014. This research has been supported by the Grant Agency of the CTU in Prague, grant No. SGS10/202/OHK3/2T/13.

References

1. Aguillo, I.F., Ortega, J.L., Fernández, M.: Webometric Ranking of World Universities: Introduction, Methodology, and Future Developments. *Higher Education in Europe* 33(3), 234–244 (2008)
2. Han, S.K., Shin, D., Jung, J.Y., Park, J.: Exploring the Relationship between Keywords and Feed Elements in Blog Post Search. *World Wide Web - Internet and Web Information Systems* 12(4), 381–398 (2009)
3. Holmberg, K., Thelwall, M.: Local government web sites in Finland: A geographic and webometric analysis. *Scientometrics* 79(1), 157–169 (2009)
4. Langville, A.N., Meyer, C.D.: *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton (2006), ISBN 978-0691122021
5. Ortega, J., Aquillo, I.F.: Visualization of the Nordic academic web: Link analysis using social network tools. *Information Processing & Management* 44(4), 1624–1633 (2008)
6. Pang, L., Lee, L.: Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1), 1–135 (2008)
7. Potthast, M., Becker, S.: Opinion Summarization of Web Comments. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) *ECIR 2010*. LNCS, vol. 5993, pp. 668–669. Springer, Heidelberg (2010)
8. Prabowo, R., Thelwall, M.: Sentiment analysis: A combined approach. *Journal of Informetrics* 3(1), 143–157 (2009)
9. Thelwall, M.: Bibliometrics to Webometrics. *Journal of Information Science* 34(4), 605–621 (2008)

10. Thelwall, M.: Blog searching: The first general-purpose source of retrospective public opinion in social science? *Online Information Review* 31(3), 277–289 (2007)
11. Thelwall, M.: *Introduction to Webometrics: Quantitative Web Research for the Social Sciences*. Morgan & Claypool, San Rafael (2009), ISBN 9781598299946
12. Thelwall, M., Wilkinson, D.: Public dialogs in social network sites: What is their purpose? *Journal of the American Society for Information Science and Technology* 61(2), 392–404 (2010)

Social Interaction with Cultural Heritage on the Web^{*}

Max Arends and Doron Goldfarb

Institut für Softwaretechnik und Interaktive Systeme
Technische Universität Wien
Favoritenstraße 9-11, A-1040 Wien

Abstract. This research proposal aims at bridging some of the current trends regarding Cultural Heritage institution's usage of Web based technologies. These trends include the increased use of Social Web principles with respect to traditional Web presences and the employment of Web3D technologies for single- or multi-user 3D environments for entertainment and education. Seen individually, each of these topics has already been covered by a number of different research projects, but relatively little work has been done on finding ways to elaborate these approaches by combining them within a common framework. The latter can also be observed in the practical usage patterns of these technologies, being usually limited to rather self-contained projects. Therefore, the aim is to provide a framework for the presentation of cultural heritage artefacts in a 3D virtual collaborative environment where visitors are able to participate using both synchronous and asynchronous means of communication.

1 Introduction

Cultural Heritage Institutions nowadays need far more advanced Web presentations in order to get in touch with their potential visitors. It's not sufficient anymore to have a Web page that informs visitors only about basic information like opening hours and the current exhibition. Since the Web 2.0 era, visitors are much more involved in the contribution of information, and appreciate the information contributed by others. This can also be helpful for different groups of visitors, as people with a novice knowledge use a different jargon than persons with an expert knowledge of Cultural Heritage information. With the use of user-contributed tags for artefacts, this gap may be closed. As Chan observes "these user keywords most often are generally descriptive, allowing users to discover objects that are difficult to discover through the Museum's formal classification system" [3]. Cultural Heritage Institutions nowadays also aim to contribute their whole collections on their Web site and to provide an idea of their cultural goal. This is where a 3D environment has advantages to traditional viewing. In a study presented in [1] two groups of students were presented with either a paper reproduction of an artefact or a 3D representation, and then asked to come up with an appropriate title of the artefact. Participants of the 3D group tended to "think not 'what' they faced, but to 'why' or 'how' something was in front of them". If this is combined with multi-user capabilities, so called Multi-User Virtual Environments

^{*} Supervised by Dieter Merkl, PhD, Associate Professor.

(MUVE), where multiple users are able to interact with each other simultaneously, the benefits of imparting cultural heritage information shall increase even further. Therefore the development of a multi-user 3D environment featuring social-web functionality might prove beneficial for both, the institutions and their visitors.

In our expectation, users of our 3D MUVE will especially gain a better overview of the cultural context the presented objects are embedded in, as for example the knowledge about relationships between artists, their oeuvre and general historical information is important for the understanding of individual artworks.

2 Research Objectives

2.1 Find Useful Visualisation Approaches

The spectrum of visualisation methods spans from real-life simulation [5] to completely abstract Information Visualisation metaphors [8]. Key elements of Cultural Heritage related presentations are the visualisation of the artefacts themselves, their respective cross-references to other objects, concepts or people according to specific attributes like creator, style or location. The environment in which all the information is presented plays an important role as well. Given the huge variety of Cultural Heritage content, it cannot be expected to find a single visualisation approach that satisfies all the different requirements. Therefore, a number of experiments should be carried out in order to find suitable methods for well-defined usage scenarios, e.g. to find the best way of displaying relationships between different painters and how these relations might have influenced their work.

2.2 Derive New Cross-References and Classifications through User Contribution

Cross-references between artefacts and their respective classification schemes are usually derived from expert knowledge. As Web 2.0-style user contributions like tagging, rating or commenting may lay the foundation of user-generated indexes - as for example discussed in [3] - that might be easier to comprehend for larger audiences, their integration into the environment is another dedicated objective. This raises issues regarding the interfaces that provide the ability to contribute, as well as how participation can be fostered.

2.3 Analyze Usage Patterns

Since user's behaviour in virtual environments can be tracked much more easily than in real life, valuable information regarding the visitor's interests and interaction patterns can be collected. The respective findings can be used by Cultural Heritage institutions to better understanding their audiences, potentially leading to increased benefit for both sides. An interesting example where visitor behaviour is studied can be found in [4].

2.4 Construct Narratives

Simply visualising content organised according to a number of common attributes might be perceived as too artificial and lacking consistent storytelling. "Hard coding"

of narratives would, however, take away much of the flexibility of virtual presentation systems. Therefore, mechanisms should be included that enable domain experts and interested visitors to create “stories” about the presented content, e.g. to organise a collection of artefacts according to specific attributes and to put them into a chronological sequence that might be more based on subjective criteria than on hard facts only. This can only be achieved by providing some kind of “high-level” language that is dedicated to such storytelling information. Examples for such languages are proposed in [9] and in [6].

2.5 Harvesting Web Based Data Repositories

Cultural Heritage data is usually stored in proprietary formats, dependent on the respective institution’s policy. Throughout the recent years, however, more and more data repositories are opened up to the public and organized according to common standards and technologies. Providing interfaces to these open data collections would therefore enable to overcome institutional boundaries and facilitate the linkage of existing information sources by combining them to “Mash-Ups” within the 3D environment.

3 Research Methodology

Get an overview of Cultural Heritage Web institutions use of Web based technologies. The websites of cultural heritage institutions are analysed according to their web features.

Overview of Web 3D in Cultural Heritage. Different Web3D solutions are examined in order to get an idea of how cultural heritage is represented in virtual environments.

Find a suitable platform. Different multi-user virtual 3D environments like Second Life, OpenSim, Project Wonderland and Unity3D are evaluated according to openness, scalability, data integration and performance.

Consider different approaches for the internal representation of Cultural Heritage information. A data model is developed that is scalable of good performance to represent large amounts of cultural heritage information by studying what representation models are available for cultural heritage and how thesauri and web data-sources may be used in order to connect different sources.

Consider different visualisation metaphors for different audiences. Different skills of visitors like school children, adults or experts need different visualisation metaphors in order put across the information. Therefore an open system is developed, where different skilled visitors are able to collaboratively use the system.

The elaborated methods are implemented in a prototype.

Evaluation. The prototype is evaluated by testers with different levels of computer skills and knowledge of Cultural Heritage information. Evaluation should focus on both improving the learning outcome as well as improving the overall usability. We intend this process to iterate between the steps of finding visualisation metaphors, implementation and evaluating their effect.

4 Current Status

Until this moment we are working on the implementation of our first prototype. We started by getting an overview of the way that Cultural Heritage Institutions use Web sites and community-oriented Web 2.0 features in order to communicate with their potential visitors. This was done by analysing the websites of 69 art museums from all over the world, including the 20 most visited, according to their collaborative and interactive aspects. Our results were presented at the IADIS 2009 and published in [2].

By checking the use of Web 3D in cultural heritage institutions, it was interesting to see that there are two different kinds of approaches. The first is to create a realistic replica of a museum in a 3D virtual environment, as done by the Old Masters Picture Gallery of the Dresden State Art Collections [5], where the whole facade including all rooms and artefacts are modelled true-to-scale in Second Life. The other approach is more abstract, where information is thematically arranged and it is possible to arrange and compare artefacts according to their artists, sujets, era, geographical information, etc.

We were then looking for a suitable platform that combines openness, free availability for academic use and the possibility to run the application as a browser-plugin. After looking at Second Life, OpenSim and Project Wonderland, we decided to use the GameEngine Unity3D, which comes with a browser-plugin that allows users to access the application directly with their browser and not having to install and run a stand-alone application on their computer.

In order to develop a data-model that offers scalability and is suited for the use of cultural heritage information, we first chose to use the CIDOC CRM, "a formal ontology intended to facilitate the integration, mediation and interchange of heterogeneous cultural heritage information" [7], but later on decided to follow the VRA Core based approach introduced in [10]. In order to improve the access to the information for users, we integrated the Getty Thesaurus of Geographic Names (TGN) and the Union List of Artist Names (ULAN) into our data-model.

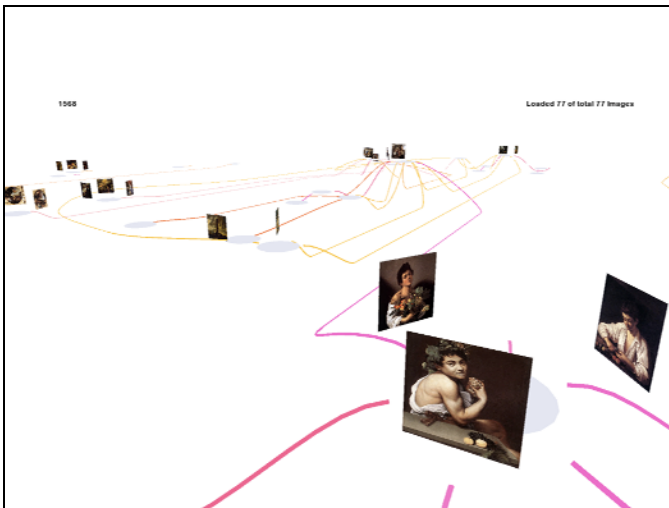


Fig. 1. Information Landscape displaying artists and their work, interconnected by lines representing their different types of relationships



Fig. 2. Close up of an artwork. Existing metadata is displayed around the virtual canvas.

We are currently working on the development of our first prototype, getting familiar with Unity3D and implement different narratives for different groups of users.

The following figures should give an impression of our current environment. Figure 1 shows an overview on the Information Landscape that is created from relationships between artists from the Getty ULAN database. Each artist is represented as a node and is surrounded by his or her artworks. The differently coloured edges between the nodes show known relations between artists, each color represents a certain type of relationship like teacher/student, patronage etc. In Figure 2, a close up of an artwork is shown. Metadata describing the title, date of creation, type and sujet etc. is displayed when the virtual visitor gets close to the artifact.

5 Main Contribution

The main contribution of this work to the field of web engineering consists of providing a Web3D based environment for browsing and querying Cultural Heritage content that is available on the Web. Finding methods to organise and present “Mash-Ups” of existing data sources in a dynamic Multi-User 3D environment might provide new insight about existing relationships and more intuitive ways of interaction with Web content.

Acknowledgment

This work is funded by the FWF (Fonds zur Förderung der wissenschaftlichen Forschung / Austrian Science Fund), Project No. L602, "The Virtual 3D Social Experience Museum".

References

- [1] Antonietti, A., Cantoia, M.: To see a painting versus to walk in a painting: an experiment on sense-making through virtual reality. *Computers & Education* 34(3-4), 213–223 (2000)
- [2] Arends, M., Goldfarb, D., Merkl, W., Weingartner, M.: Interaction With Art Museums on the Web. In: Isaias, P., White, B., Nunes, M.B. (eds.) *Proceedings of the IADIS Int'l. Conference WWW/Internet 2009*. IADIS Press (2009)
- [3] Chan, S.: Tagging and searching - Serendipity and museum collection databases. In: *Proceedings of the Museums and the Web Conference*, San Francisco, CA (2007)
- [4] Cosley, D., Baxter, J., Lee, S., Alson, B., Nomura, S., Adams, P., Sarabu, C., Gay, G.: A tag in the hand: supporting semantic, social, and spatial navigation in museums. In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, Boston, MA, USA (2009)
- [5] Dresden Gallery, <http://www.dresdengallery.com>
- [6] Not, E., Callaway, C., Rocchi, C., Stock, O., Zancanaro, M.: Cinematographic techniques for automatic documentary-like presentations. In: Stock, O., Zancanaro, M. (eds.) *PEACH: Intelligent Interfaces for Museum Visits*. Cognitive Technologies Series, pp. 23–44. Springer, Berlin (2007)
- [7] Past and Future of ISO21127:2006 or CIDOC CRM, [http://cidoc.mediahost.org/standard_crm\(en\)\(E1\).xml](http://cidoc.mediahost.org/standard_crm(en)(E1).xml) (2006)
- [8] Ruffaldi, E., Evangelista, C., Neri, V., Carrozzino, M., Bergamasco, M.: Design of information landscapes for cultural heritage content. In: *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, Athens, Greece, pp. 113–119 (2008)
- [9] TourML specification, <http://wiki.museummobile.info/museums-to-go/productsservices/tourml>
- [10] Van Assem, M., Van Ossenbruggen, J., Schreiber, G.: The VRA Core Application Profile for searching and presenting cultural heritage: the MultimediaN case. In: *Proceedings of the International Conference on Dublin Core and Metadata Applications*, Pittsburgh, PA, USA (2010)

Author Index

- Ambroszkiewicz, Stanisław 256
Arbelaitz, Olatz 486
Arellano, Cristóbal 417
Arends, Max 587
- Baez, Marcos 384
Bartyna, Waldemar 256
Bauer, Cordula 421
Bechhofer, Sean 232
Becker, Pablo 102
Belhajjame, Khalid 232
Biermann, Elmarie 550
Bozzon, Alessandro 61
Braatz, Benjamin 187
Brambilla, Marco 61
Brandt, Christoph 187
Bravo, Jose 459
Burella, Juan 13
Byun, Giseob 73
- Caballero, Ismael 91
Cabri, Giacomo 289
Calero, Coral 91
Campi, Alessandro 61
Cappiello, Cinzia 360
Carboni, Davide 431
Casati, Fabio 277
Castillo, Roger 126
Ceri, Stefano 61
Chaisatien, Prach 372
Chun, Soon Ae 175
Cisternino, Antonio 425
Comai, Sara 25
Comerio, Marco 347
Corcho, Óscar 266
Corcoglioniti, Francesco 61
Coulleri, Cristina 509
- Daniel, Florian 277
Dannecker, Lars 1
Daramola, Olawande J. 521
De Angeli, Antonella 396
De Coi, Juri Luca 150
De Giorgio, Teodoro 338
Díaz, Oscar 417
- Di Bitonto, Pierpaolo 498
Di Martino, Beniamino 211
Di Noia, Tommaso 138
Di Sciascio, Eugenio 138, 199
Di Tria, Francesco 498
Domínguez-Mayo, F.J. 571
Douglas Jacyntho, Mark 37
Drory, Tal 408
Dustdar, Schahram 347
- Elgammal, Amal 325
Embury, Suzanne M. 232
Escalona, M.J. 571
- Faderewski, Marek 256
Feldmann, Marius 1
Firmenich, Sergio 566
Fisichella, Marco 150
Fontecha, Jesus 459
Fraternali, Piero 61
Funahashi, Takuya 114
- Gambi, Alessio 25
Ganzha, Maria 289
Garcia, Ander 486
Gawinecki, Maciej 289
Geiger, Nina 49
Geller, James 175
George, Tobias 49
Gil, Miriam 463
Giner, Pau 463
Giordano, Alessandro 431
Goldfarb, Doron 587
Gómez-Goiri, Aitor 447
Gordillo, Silvia 13, 566
Grigera, Julián 13
Guinard, Dominique 442
Guo, Junxia 372, 545
- Hahn, Marcel 49
Halima, Riadh Ben 313
Han, Hao 372
Herrera, Mayte 91
Hervás, Ramon 459
Hübsch, Gerald 1

- Ieva, Saverio 199
 Iturrioz, Jon 417
- Jelínek, Ivan 581
 Jmaiel, Mohamed 301, 313
 Jubeih, Ruben 49
 Jugel, Uwe 1
- Karnin, Ehud D. 408
 Kern, Robert 421
 Kim, Taisiya 73
 Kiyavitskaya, Nadzeya 509
 Kokash, Natallia 325
 Kroeger, Reinhold 163
 Kuuskeri, Janne 244
- Laterza, Maria 498
 La Vecchia, Gioacchino 425
 Lee, Bong Gyou 73
 Leser, Ulf 126
 Leszczyński, Paweł 576
 Leymann, Frank 325
 Lim, Erbin 560
 Linaza, Maria Teresa 486
 López-de-Ipiña, Diego 447
- Maâlej, Afef Jmal 301
 Malinský, Radek 581
 Marzouk, Soumaya 301, 313
 Matera, Maristella 150, 360
 Maurino, Andrea 347
 Mdhaffar, Afef 313
 Mejías, M. 571
 Mican, Daniel 85
 Mich, Luisa 509
 Mikkonen, Tommi 244
 Mikułowski, Dariusz 256
 Minno, Michele 223
 Mirizzi, Roberto 138
 Moraga, M^a Ángeles 91
 Mostarda, Michele 223
 Mtsweni, Jabu 550
 Muñoz, Pablo 463
 Muthmann, Klemens 1
- Namoun, Abdallah 396
 Nestler, Tobias 1, 396
 Noro, Tomoya 372
- Oliveira, Fábio 412
 Olsina, Luis 102
 Ortiz-Chamorro, Sebastian S. 533
- Palmisano, Davide 223
 Paprzycki, Marcin 289
 Park, Keon Chul 73
 Park, Soo Kyung 73
 Picozzi, Matteo 360
 Pilski, Marek 256
 Pintus, Antonio 431
 Piras, Andrea 431
 Pretorius, Laurette 550
- Ragone, Azzurra 138
 Ramos, Isabel 412
 Ripa, Gianluca 338
 Rivero, José Matías 13
 Robles Luna, Esteban 13
 Rodolfi, Marta 360
 Rodríguez, Carlos 277
 Roselli, Teresa 498
 Rossano, Veronica 498
 Rossi, Gustavo 13, 533, 566
 Ruta, Michele 199
- Santos, Leonel 412
 Saquicela, Victor 266
 Satzger, Gerhard 421
 Schumm, David 325
 Schuster, Nelly 538
 Schwabe, Daniel 37, 533
 Scioscia, Floriano 199
 Seo, Hyunsik 73
 Silveira, Patrícia 277
 Singh, Ravinder 232
 Soi, Stefano 384
 Souffriau, Wouter 474, 486
 Stencil, Krzysztof 576
 Stepniak, Marcin 256
 Stynes, Jeanne 163
- Tai, Stefan 538
 Tangorra, Filippo 498
 Terlikowski, Grzegorz 256
 Textor, Andreas 163
 Thies, Hans 421
 Thiran, Philippe 560
 Tian, Tian 175
 Toffetti, Giovanni 25
 Tokuda, Takehiro 372, 545

- Tomai, Nicolae 85
Truong, Hong-Linh 347
Turetken, Oktay 325
- Vadacca, Salvatore 61
van den Heuvel, Willem-Jan 325
Vansteenwegen, Pieter 474, 486
Vilches-Blázquez, Luis.M. 266
- Walach, Eugene 408
Winckler, Marco 566
- Yamana, Hayato 114
- Zeni, Nicola 509
Zirpins, Christian 538
Zuccalà, Maurilio 338
Zündorf, Albert 49