

Luciano Baresi
Piero Fraternali
Geert-Jan Houben (Eds.)

LNCS 4607

Web Engineering

7th International Conference, ICWE 2007
Como, Italy, July 2007
Proceedings

 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

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

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Luciano Baresi Piero Fraternali
Geert-Jan Houben (Eds.)

Web Engineering

7th International Conference, ICWE 2007
Como, Italy, July 16-20, 2007
Proceedings

Volume Editors

Luciano Baresi
Piero Fraternali
Politecnico di Milano
Dipartimento di Elettronica e Informazione
piazza Leonardo da Vinci 32, 20133 Milano, Italy
E-mail: {luciano.baresi, piero.fraternali}@polimi.it

Geert-Jan Houben
Vrije Universiteit Brussel
Department of Computer Science
Pleinlaan 2, 1050 Brussels, Belgium
Geert-Jan.Houben@vub.ac.be

Library of Congress Control Number: 2007930462

CR Subject Classification (1998): D.2, C.2, I.2.11, H.4, H.2, H.3, H.5, K.4, K.6

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

ISSN 0302-9743
ISBN-10 3-540-73596-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73596-0 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 is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12089673 06/3180 5 4 3 2 1 0

Preface

This volume contains the proceedings of the seventh International Conference on Web Engineering (ICWE2007), which was held in Como (Italy) in July 2007. The conference is the flagship event of the community, promoting research and scientific progress in the Web engineering field. The importance of the Web — and of its many related technologies— is widening the scope of the conference year after year, and is also leading to the cross-fertilization of several related disciplines (e.g., requirements engineering, testing and analysis, communication science, HCI, multimedia, and more). The conference brings together practitioners, scientists, and researchers committed to investigating and innovating the technologies, methodologies, tools, processes, and techniques used to construct, verify, and maintain Web-based applications and services.

This year, the Call for Papers attracted a high number of submissions with a very good coverage of all the different facets of the Web engineering discipline. A total of 172 submissions (as for research papers) allowed us to build an attractive program of high technical and scientific quality. The 39 selected submissions comprise 26 full papers and 13 short papers (with an acceptance rate close to 23%); they cover the different aspects highlighted in the Call for Papers and represent well the many Web engineering research groups active worldwide. The program spans from service-based systems to testing and analysis, from quality and metrics to models, and from semantic issues and Web 2.0 to application development techniques.

This year, we strove to attain a very homogeneous program in which the different initiatives (technical program, workshops, tutorials, demos, and doctoral symposium) are intertwined in a coherent sequence of events. The technical program remained the main forum for presenting innovative, and already consolidated, research results, while the collateral events were valuable opportunities to present interesting ideas and practical realizations and to discuss on-going research, and thus advance the state of the art in the field.

The conference would not have been possible without the help of Politecnico di Milano, WebRatio, Acer, and Cefriel, which kindly accepted to support the conference, the endorsement of the International World Wide Web Conference Committee and the International Society for Web Engineering, and the enthusiastic work of the various chairs (Emilia, Nora, Maristella, Sara, Fabio, Sven, Alexander, Michalis, Antonio, and Marco), and of the local organizers. Finally, special thanks to all the contributors and participants who, at the end of the day, are what a conference like ICWE is all about.

May 2007

Luciano Baresi
Piero Fraternali
Geert-Jan Houben

Organization

General Chair

Piero Fraternali, Politecnico di Milano, Italy

Program Chairs

Luciano Baresi, Politecnico di Milano, Italy

Geert-Jan Houben, Vrije Universiteit Brussel, Belgium

Program Committee

Grigoris Antoniou, University of Crete, Greece

Uwe Assmann, TU Dresden, Germany

Maria Bielikova, Slovak University of Technology in Bratislava, Slovakia

Judith Bishop, University of Pretoria, South Africa

Marco Brambilla, Politecnico di Milano, Italy

Chris Brooks, University of San Francisco, USA

Fabio Casati, Università di Trento, Italy

Sven Casteleyn, Vrije Universiteit Brussel, Belgium

Dan Chiorean, University Babeş-Bolyai, Cluj, Romania

Sara Comai, Politecnico di Milano, Italy

Paul Dantzig, IBM T. J. Watson Research Center, USA

Yogesh Deshpande, University of Western Sydney, Australia

Olga De Troyer, Vrije Universiteit Brussel, Belgium

Peter Dolog, Aalborg University, Denmark

Schahram Dustdar, Vienna University of Technology, Austria

Martin Gaedke, University of Karlsruhe, Germany

Dragan Gasevic, Simon Fraser University, Canada

Michael Gertz, UC-Davis, USA

Athula Ginige, University of Western Sydney, Australia

Angela Goh, NTU, Singapore

Jaime Gomez, Universidad de Alicante, Spain

Volker Gruhn, Universität Leipzig, Germany

Gerti Kappel, Vienna University of Technology, Austria

Alexander Knapp, Ludwig Maximilians Universität München, Germany

Nora Koch, Ludwig Maximilians Universität München, Germany

Frank Leymann, Universität Stuttgart, Germany

David Lowe, University of Technology Sydney, Australia

Maristella Matera, Politecnico di Milano, Italy

Heinrich Mayr, University of Klagenfurt, Austria

Emilia Mendes, University of Auckland, New Zealand
San Murugesan, University of Southern Cross, Australia
Moira Norrie, ETH, Switzerland
Oscar Pastor, Universidad Politecnica de Valencia, Spain
Vicente Pelechano, Universidad Politecnica de Valencia, Spain
Michalis Petropoulos, University at Buffalo, SUNY, USA
Claudia Pons, University of La Plata, Argentina
I.V. Ramakrishnan, Stony Brook University, USA
Gustavo Rossi, Universidad Nacional de la Plata, Argentina
Daniel Schwabe, PUC Rio de Janeiro, Brazil
Katsumi Tanaka, Kyoto University, Japan
Ernest Teniente, UPC, Spain
Bernhard Thalheim, Universität Kiel, Germany
Philippe Thiran, University of Namur, Belgium
Riccardo Torlone, Università di Roma, Italy
Paolo Traverso, ITC, Italy
Antonio Vallecillo, Universidad Politecnica de Valencia, Spain
Jean Vanderdonckt, Université Catholique de Louvain, Belgium
Fabio Vitali, University of Bologna, Italy
Petri Vuorimaa, Helsinki University of Technology, Finland
Jeffrey Yu, Chinese University of Hong Kong, China

Industrial Track Committee

Fabio Casati, Università di Trento, Italy (Chair)
Boualem Benatallah, University of New South Wales Sydney, Australia
Aldo Bongio, WebRatio, Italy
Arthur Ryman, IBM Toronto, Canada
Regis Saint-Paul, University of New South Wales Sydney, Australia
Erwin Schaumlechner, Tiscover, Austria
Ming-Chien Shan, SAP Research, USA
Bebo White, SLAC, USA
Jin Yu, Martsoft, USA

Doctoral Consortium Committee

Nora Koch, Ludwig Maximilians Universität München, Germany (Chair)
Piero Fraternali, Politecnico di Milano, Italy
Gerti Kappel, Vienna University of Technology, Austria
Alexander Knapp, Ludwig Maximilians Universität München, Germany
David Lowe, University of Technology Sydney, Australia
Emilia Mendes, University of Auckland, New Zealand
Luis Olsina, Universidad Nacional de la Pampa, Argentina
Oscar Pastor, Universidad Politecnica de Valencia, Spain
Antonio Vallecillo, Universidad Politecnica de Valencia, Spain

Demo Chairs

Sven Casteleyn, Vrije Universiteit Brussel, Belgium

Alexander Knapp, Ludwig Maximilians Universität München, Germany

Tutorial Chair

Michalis Petropoulos, University at Buffalo, SUNY, USA

Publicity Chairs

Maristella Matera, Politecnico di Milano, Italy

Antonio Vallecillo, Universidad Politecnica de Valencia, Spain

Treasurer and Sponsorship Chair

Sara Comai, Politecnico di Milano, Italy

Local Organization Chair

Marco Brambilla, Politecnico di Milano, Italy

Additional Referees

Omar Alonso

Andreas Bartho

Bettina Biel

Sören Blom

Matthias Book

Eugene Borodin

Alessandro Bozzon

Daniele Braga

Michael Byrd

Alina Campan

Alessandro Campi

Sebastian Cech

Alejandra Cechich

Florian Daniel

Sergio España

Federico Michele Facca

Paul Fodor

Joan Fons

Conny Franke

Irene Garrigos

Robert Gombotz

Sam Guinea

Michael Göschka

Falk Hartmann

Christian Wende

Florian Heidenreich

Christian Huemer

Malte Hülder

Angelo Di Iorio

Horst Kargl

Nima Kaviani

Nicolas Kicillof

Engin Kirda

Gerhard Kramler

Katja Lehmann

Tammo van Lessen

Jalal Mahmud

Daniel Martin

Johannes Meinecke

Santiago Melia

Elke Michlmayr

Ralph Mietzner

Milan Milanovic

Hisashi Miyamori

Shinsuke Nakajima

Satoshi Nakamura

Jose I. Panach Navarrete

Johann Oberleitner

Hiroaki Ohshima

Satoshi Oyama

Filippo Pacifici

Conrad Parker

Ins Pederiva

Pierluigi Plebani

Marco Plebani

Birgit Pröll

Alessandro Raffio

Sebastian Richly
Thomas Richter
Florian Rosenberg
Andreas Schroeder
Wieland Schwinger
Clemens Schäfer
Martina Seidl
Anu Singh
Keishi Tajima
Puay Siew Tan
Taro Tezuka

Dave Thau
Massimo Tisi
Victoria Torres
Martin Treiber
Christina Tziviskou
Tobias Unger
Pedro Valderas
Francisco Valverde
Martin Vasko
Roberto De Virgilio
Valentino Vranic

Hui Wan
Junhu Wang
Jianshu Weng
Branimir Wetzstein
Matthias Wieland
Daniel Wutke
Qing Zhu
Daniel Zinn
Chang Zhao

Table of Contents

Services

On Embedding Task Memory in Services Composition Frameworks	1
<i>Rosanna Bova, Hye-Young Paik, Salima Hassas, Salima Benbernou, and Boualem Benatallah</i>	
A QoS Test-Bed Generator for Web Services	17
<i>Antonia Bertolino, Guglielmo De Angelis, and Andrea Polini</i>	
Engineering Compensations in Web Service Environment	32
<i>Michael Schäfer, Peter Dolog, and Wolfgang Nejdl</i>	
Context-Aware Workflow Management	47
<i>Liliana Ardissono, Roberto Furnari, Anna Goy, Giovanna Petrone, and Marino Segnan</i>	
Practical Methods for Adapting Services Using Enterprise Service Bus	53
<i>Hyun Jung La, Jeong Seop Bae, Soo Ho Chang, and Soo Dong Kim</i>	

Metrics and Quality

On the Quality of Navigation Models with Content-Modification Operations	59
<i>Jordi Cabot, Jordi Ceballos, and Cristina Gómez</i>	
Metamodeling the Quality of the Web Development Process' Intermediate Artifacts	74
<i>Cristina Cachero, Coral Calero, and Geert Poels</i>	
The Use of a Bayesian Network for Web Effort Estimation	90
<i>Emilia Mendes</i>	

Caching

Sequential Pattern-Based Cache Replacement in Servlet Container	105
<i>Yang Li, Lin Zuo, Jun Wei, Hua Zhong, and Tao Huang</i>	
A Hybrid Cache and Prefetch Mechanism for Scientific Literature Search Engines	121
<i>Huajing Li, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles</i>	

Interfaces

Finalizing Dialog Models at Runtime	137
<i>Stefan Betermieux and Birgit Bomsdorf</i>	
Transparent Interface Composition in Web Applications	152
<i>Jeronimo Ginzburg, Gustavo Rossi, Matias Urbieto, and Damiano Distante</i>	
Fine-Grained Specification and Control of Data Flows in Web-Based User Interfaces	167
<i>Matthias Book, Volker Gruhn, and Jan Richter</i>	
Authoring Multi-device Web Applications with Database Access	182
<i>Giulio Mori, Fabio Paternò, and Carmen Santoro</i>	
Enriching Hypermedia Application Interfaces	188
<i>André T.S. Fialho and Daniel Schwabe</i>	

Models

Functional Web Applications	194
<i>Torsten Gipp and Jürgen Ebert</i>	
Integrating Databases, Search Engines and Web Applications: A Model-Driven Approach	210
<i>Alessandro Bozzon, Tereza Iofciu, Wolfgang Nejdl, and Sascha Tönnies</i>	
A Method for Model Based Design of Rich Internet Application Interactive User Interfaces	226
<i>M. Linaje, Juan C. Preciado, and F. Sánchez-Figueroa</i>	
Improving Communication in Requirements Engineering Activities for Web Applications	242
<i>Pedro Valderas and Vicente Pelechano</i>	
Meta-model to Support End-User Development of Web Based Business Information Systems	248
<i>Buddhima De Silva and Athula Ginige</i>	

Verification and Testing

Easing Web Guidelines Specification	254
<i>Barbara Leporini, Fabio Paternò, and Antonio Scorcìa</i>	

A Transformation-Driven Approach to the Verification of Security Policies in Web Designs	269
<i>Esther Guerra, Daniel Sanz, Paloma Díaz, and Ignacio Aedo</i>	
Efficiently Detecting Webpage Updates Using Samples	285
<i>Qingzhao Tan, Ziming Zhuang, Prasenjit Mitra, and C. Lee Giles</i>	
Auto-Generating Test Sequences for Web Applications	301
<i>Hongwei Zeng and Huaikou Miao</i>	
A Survey of Analysis Models and Methods in Website Verification and Testing	306
<i>Manar H. Alalfi, James R. Cordy, and Thomas R. Dean</i>	
Semantics and Web 2.0	
Building Semantic Web Portals with WebML	312
<i>Marco Brambilla and Federico M. Facca</i>	
Engineering Semantic-Based Interactive Multi-device Web Applications	328
<i>Pieter Bellekens, Kees van der Sluijs, Lora Aroyo, and Geert-Jan Houben</i>	
Towards Improving Web Search by Utilizing Social Bookmarks	343
<i>Yusuke Yanbe, Adam Jatowt, Satoshi Nakamura, and Katsumi Tanaka</i>	
Designing Interaction Spaces for Rich Internet Applications with UML	358
<i>Peter Dolog and Jan Stage</i>	
A Behavioral Model for Rich Internet Applications	364
<i>Sara Comai and Giovanni Toffetti Carughi</i>	
Search	
Considering Web Accessibility in Information Retrieval Systems	370
<i>Myriam Arrue and Markel Vigo</i>	
Fixing Weakly Annotated Web Data Using Relational Models	385
<i>Fatih Gelgi, Srinivas Vadrevu, and Hasan Davulcu</i>	
Creating Personal Histories from the Web Using Namesake Disambiguation and Event Extraction	400
<i>Rui Kimura, Satoshi Oyama, Hiroyuki Toda, and Katsumi Tanaka</i>	

Comparing Clustering Algorithms for the Identification of Similar Pages in Web Applications 415
Andrea De Lucia, Michele Risi, Giuseppe Scanniello, and Genoveffa Tortora

Structural Patterns for Descriptive Documents 421
Antonina Dattolo, Angelo Di Iorio, Silvia Duca, Antonio Angelo Feliziani, and Fabio Vitali

Application Development

Component-Based Content Linking Beyond the Application 427
Johannes Meinecke, Frederic Majer, and Martin Gaedke

A Double-Model Approach to Achieve Effective Model-View Separation in Template Based Web Applications 442
Francisco J. García, Raúl Izquierdo Castanedo, and Aquilino A. Juan Fuente

Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces 457
Damiano Distante, Paola Pedone, Gustavo Rossi, and Gerardo Canfora

On Refining XML Artifacts 473
Felipe I. Anfurrutia, Oscar Díaz, and Salvador Trujillo

Demonstrations

Mixup: A Development and Runtime Environment for Integration at the Presentation Layer 479
Jin Yu, Boualem Benatallah, Fabio Casati, Florian Daniel, Maristella Matera, and Regis Saint-Paul

Squiggle: An Experience in Model-Driven Development of Real-World Semantic Search Engines 485
Irene Celino, Emanuele Della Valle, Dario Cerizza, and Andrea Turati

WebTE: MDA Transformation Engine for Web Applications 491
Santiago Meliá, Jaime Gómez, and José Luís Serrano

Noodles: A Clustering Engine for the Web 496
Giansalvatore Mecca, Salvatore Raunich, Alessandro Pappalardo, and Donatello Santoro

WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications.....	501
<i>Roberto Acerbis, Aldo Bongio, Marco Brambilla, and Stefano Butti</i>	
Extending Ruby on Rails for Semantic Web Applications	506
<i>Cédric Mesnage and Eyal Oren</i>	
Personalized Faceted Navigation in the Semantic Web	511
<i>Michal Tvarožek and Mária Bielíková</i>	
WebVAT: Web Page Visualization and Analysis Tool	516
<i>Yevgen Borodin, Jalal Mahmud, Asad Ahmed, and I.V. Ramakrishnan</i>	
Smart Tools to Support Meta-design Paradigm for Developing Web Based Business Applications.....	521
<i>Athula Ginige, Xufeng Liang, Makis Marmaridis, Anupama Ginige, and Buddhima De Silva</i>	
Industrial Track	
Next-Generation Tactical-Situation-Assessment Technology (TSAT): Chat	526
<i>Emily W. Medina, Sunny Fugate, LorRaine Duffy, Dennis Magsombol, Omar Amezcua, Gary Rogers, and Marion G. Ceruti</i>	
Tool Support for Model Checking of Web Application Designs	533
<i>Marco Brambilla, Jordi Cabot, and Nathalie Moreno</i>	
Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA.....	539
<i>Roberto Acerbis, Aldo Bongio, Marco Brambilla, Massimo Tisi, Stefano Ceri, and Emanuele Tosetti</i>	
The Challenges of Application Service Hosting.....	545
<i>Ike Nassi, Joydip Das, and Ming-Chien Shan</i>	
Doctoral Consortium	
Securing Code in Services Oriented Architecture	550
<i>Emilio Rodriguez Priego and Francisco J. García</i>	
Service Level Agreements: Web Services and Security	556
<i>Ganna Frankova</i>	

Risk Management for Service-Oriented Systems	563
<i>Natallia Kokash</i>	
A Framework for Situational Web Methods Engineering.....	569
<i>Sebastian Lahajnar</i>	
Author Index	575

On Embedding Task Memory in Services Composition Frameworks

Rosanna Bova¹, Hye-Young Paik², Salima Hassas³, Salima Benbernou¹,
and Boualem Benatallah²

¹LIRIS, University Lyon 1, France

{rosanna.bova, salima.benbernou}@liris.cnrs.fr

²CSE, University of New South Wales, Australia

{hpaik, boualem}@cse.unsw.edu.a

³LIESP, University Lyon 1, France

hassas@bat710.univ-lyon1.fr

Abstract. With the increasing availability of Web services and adoption of services oriented paradigm, there is a growing need to dynamically compose services for realizing complex user tasks. While service composition is itself an important problem, a key issue is also how to support users in selecting the most appropriate compositions of services to fulfill a task. In existing dynamic services selection approaches, combinations of services are repeatedly discovered (e.g., using ontology-based matching techniques) and selected by users whenever needed. To improve their effectiveness, we propose a new technique that provides an efficient access to what is named a “task memory”. A task memory is used to provide users with a context-aware service selection by recommending combinations of services that are most appropriate in a given context. A task memory is formed using the service composition history and their metadata. We present an incremental approach for building the task memory in which we monitor how users use and rank the services. The continuous updates of the task memory over time will result in more fine-tuned recommendations for composite services.

Keywords: composite web services reuse, context-aware composite web services selection, service oriented architecture.

1 Introduction

Advances in service oriented computing and semantic web technologies provide foundations to enable automated services selection and aggregation [1]. Coupled with other advances in communication technologies, these foundations constitute the pillars of a new computing paradigm in which users and services establish on-demand interactions, possibly in real-time, to realize useful experiences. This paradigm offers effective automation opportunities in a variety of application domains including personal information management, office tasks, travel, healthcare, and e-government. For example, a driver might use location, travel route computation, traffic information, and road conditions services to get timely information regarding a trip in progress.

Business travelers can cope with schedule changes by seamlessly combining services to find and book hotels, search and book nearby rental cars, change flight reservations, modify meeting schedules and notify attendees [2].

A key issue to facilitate seamless and efficient composition of services is providing appropriate support for services selection. This is especially important in environments where there may be large number of services offering similar functionality [3]. Services discovery and composition are very active area of research and standardization. Efforts in these areas focused mainly on designing languages for process-based services composition (e.g., BPEL), designing rich and machine understandable representations of service properties, capabilities, and behavior, as well as reasoning mechanisms to select and aggregate services (e.g., OWL-S and services matching techniques) [4]. Main stream services discovery and selection approaches typically rely on descriptions matching techniques (e.g., whether descriptions of services and requests are compatible). Descriptions refer to meta-data such as service capabilities and non-functional properties (e.g., quality of service properties) [3]. It should be noted that, in a description-based matching approach, identifying that a service has a capability to answer a user request, does not mean that the service will be selected by the user. For example, not all travel services offer airfares from Lyon to Sydney. Other approaches improve the effectiveness of description-based approaches by considering also content-based matching (e.g., using content summary [5] or using service probing [6]).

Although existing techniques have produced promising results that are certainly useful, more advanced techniques that cater for context (e.g., user location, computer environment), specially in large and dynamic environments, are necessary. This will relieve users from repeating the same selection refinement process to deal with a potentially large number of relevant services returned by a matching system every time they need to perform an activity. We observe that, while performing routine tasks, there is valuable knowledge being exposed to the service matching and selection component of a service infrastructure, that is, the information about the contexts in which a certain combination services were considered most appropriate by users. This information can be helpful in terms of reuse because users would select similar services in similar contexts (e.g., repetitive, regular tasks). Unfortunately, this information is not effectively captured and utilized in existing service matching and selection approaches. In this paper, we present an approach that leverages and seamlessly extends existing service matching and selecting techniques to cater for context-aware services selection by utilizing the knowledge on past experience. More precisely, we make the following contributions:

1. We introduce a notion of *task memories* that effectively represent the knowledge about service selection and contexts. We use task memories during services selection to suggest most relevant candidate services.
2. We use *incremental acquisition techniques to build and update task memory*. By applying continuous feedback and monitoring of ongoing usage of services, the system is able to maintain and evolve the task memory. Keeping the task memory up-to-date should result in more fine-tuned services selection.
3. We propose a multi-agent architecture; called, *WS-Advisor*, that seamlessly extends existing service matching and service selection techniques. The interactions among

the agents are well coordinated to cater for a comprehensive service provisioning environment which supports effective capturing and utilization of user knowledge during service matching and selection.

2 WS-Advisor: Design Overview

The proposed architecture builds on existing services matching, selection, and composition frameworks. This is to take advantage of the already known techniques [7], [8], but, more importantly, to strengthen the notion of reuse in the frameworks. We propose that using incrementally acquired knowledge about service capabilities and their usage history during service matching and selection will help promote effective adoption of reuse. The added value of this extension is making the system (named WS-Advisor) capable of providing context-aware and adaptive service provisioning in dynamic environments.

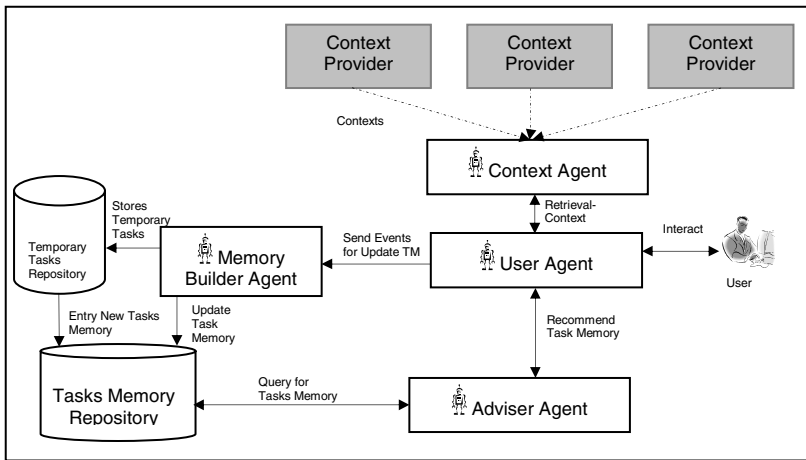


Fig. 1. Overview of WS-Advisor architecture

The main proposition of WS-Advisor is to offer effective recommendations on “best-fit” services during the process of service selection. The architecture as a whole relies on the notion of building, maintaining and querying a service usage history. By continuously monitoring which services are used in which context and how users rate the services after execution, WS-Advisor can build up extensible knowledge about a history of service usage. The knowledge is stored in the form of *task memories*, which are then queried during the selection of services to draw recommendations which are based on past experience (i.e., the best candidate services that performed well for the given task and context). In summary, WS-Advisor has two core functions, namely: (i) constructing task memories, and (ii) using task memories to recommend “best-fit” services based on the past experience.

It is noted that WS-Advisor is based on a multi-agent architecture. Figure 1 shows the agents involved in the system, namely: *user agent*, *adviser agent*, *task memory builder agent* and *context agent*. These agents use their internal knowledge and

policies to perform their functions. They interact pro actively to collect information for building the task memories and adapt continuously to provide effective service selection in dynamic environments. In the following, we introduce each agent.

User Agent (UA). There are two types of users in WS-Advisor: administrators and end users. An administrator interacts with a user agent to manage tasks (i.e., create, update or delete). For an end user, a user agent acts like a proxy, performing various actions on behalf of the user. A user agent maintains a folder of tasks (e.g., travel booking, organizing a board meeting). The user can browse, select, and execute the tasks. When a task is chosen, the user agent performs the following automatic actions: (i) it contacts a context agent (see below) to retrieve context information, (ii) after obtaining the necessary contexts, it asks the adviser agent to recommend the services suitable for the chosen task. These recommendations are passed back to the user agent who makes the final decision on which services to run, (iii) when the services are finally chosen, the user agent interacts with a service orchestration engine (e.g., BPEL execution engine) to execute the task by invoking the involved services and orchestrating their interactions.

Adviser Agent (AA). A recommendation request from the user agent includes task attributes (e.g., departure date and destination city for a travel booking task) and context (e.g., current time and location) attributes. The knowledge that the adviser agent uses for recommendation is encoded in task memories. Briefly stated, a task memory consists of tuples, each tuple containing a combination of services, the contexts in which the services were selected and executed, a score indicating how “successful” the execution was. More detailed description of task memories and the scores will be given in the later sections.

Builder Agent (BA). This agent is responsible for incremental knowledge acquisition in the task memories. It interacts with the user agent to gather service usage history (e.g., which services were recommended in which contexts, which of the recommended services were eventually chosen to be executed in the end, etc). It also continuously monitors and collects information about how the users rank the service performance after a task is completed and carry out updates in the task memories accordingly. We will discuss this agent in details in the later sections.

Context Agent (CA). The context agent collects an assortment of contexts from context providers and disseminates the information to the user agent. A context may refer to a user context (e.g., preferences, location, timezone), an environment context (e.g., hardware and software characteristics of the user’s devices). We assume that a context providing service, such as the one implemented in [9], [10], exists and it will generate the context attributes and value pairs.

3 User Agent

In this section, we describe the concepts that are important, namely, *service and context ontologies*; *tasks*, to explain the activities performed by a user agent during task provisioning.

3.1 Concepts and Definitions

Service Ontology. Briefly stated, the service ontology provides a description (e.g., domain, properties and capabilities) of potential services that could be used to execute specific activities. A service ontology can be described using an ontology description language such as OWL-S. In our approach, the service ontology is described by a name that represents the domain of services and a set of *service categories*. A service category is specified by a set of attributes and a set of operations. An attribute describes a service property and is described by its name and type.

An operation describes a service behavior and is described by its name and signature (i.e, input and output parameters of the service). Categories within a service ontology can be related by specialization and generalization relationships. In this paper, we assume that service ontologies are available and accessible for instance from registries (e.g. UDDI registries). For example, in Figure 2, the domain Travel has a category Transportation, which is described using attributes origin, destination and price, etc., and this category has three sub-categories.

A service provider advertises a service by specifying which ontology the service is complaint to and the service categories that are supported by the service. Let us assume that the service Alitalia offers a range of flight information. The service may register itself with the Transportation ontology and advertise that it supports all operations in the category Flight as well as all attributes inherited from the categories AirTransport and Transportation.

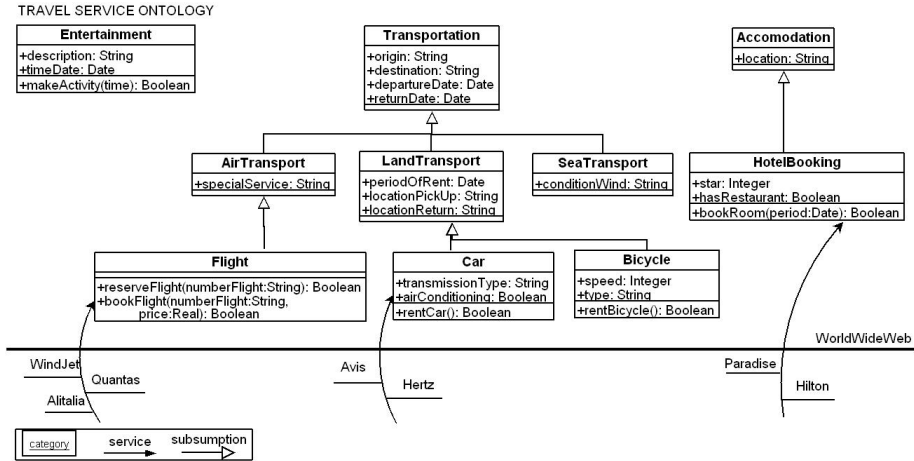


Fig. 2. An example of Service Ontology: “Travel Service Ontology”

Context Ontology. Various definitions exist in the literature for the notion context [11], [12]. For the purpose of our work, a context represents environmental or circumstantial factors that are relevant to effectively selecting services to perform a given task. We use a simple context ontology that consists of a set of context classes. Each class represents a specific aspect of task context (e.g., Spatio-Temporal context,

Computing Environment context, ConditionalEnvironment context, User context etc). These are generic classes in the sense that they are used to describe context of any task. Each class is described by a set of attributes representing specific state of the task environment. For instance, the class `Hardware` that is a sub-class of `ComputingEnvironment` contains the attributes: `memoryFree`, `cpuUsage`, `storage` and `network` and etc. It should be noted that, although the adopted context ontology has a limited number of context classes (for the sake of illustration), it is extensible: new classes can be added without fundamentally altering the service selection techniques built on top of this ontology.

Task Definition. A task in WS-Advisor represents a set of coordinated *activities* that realize recurrent needs (i.e. a process that orchestrates the execution a number of individual activities). For example, a user may define a business travel task or a driving planning task. The business travel task may include activities such as hotel booking, car rental, flight reservation, meeting scheduling and attendee's notification. A driving planning task may include activities such as gathering traffic and road conditions and producing an optimum driving route. An activity can be one of three types: (i) an *elementary* task that refers to an operation of an actual service, (ii) an elementary task that refers to an operation defined in a service ontology, or (iii) a sub-task (i.e., a task consists of other tasks).

The user agent provides support for defining new tasks and a repository for storing them. Tasks are, for example, defined by an administrator based on common patterns in recurring processes. A task is described in terms of services ontologies and is represented using state charts [7]. The choice of such notation is motivated by the fact that state charts offer main constructs that are needed to define typical user tasks such as sequence of activities, branching, and parallel activities. In addition, to their expressive power, well-defined semantics, state charts can also be translated to executable processes such BPEL. It should be noted however, that any other task modeling notation such as petri nets could be used to define tasks in our approach.

In a nutshell, a state chart representing a user task consists of states and transitions. A state can be basic, or composite. Each basic state is labeled with an execution of activity that refers to:

- *An invocation of a concrete service operation* in case the service is deemed relevant the corresponding activity whenever the task is performed. This means that, the binding of an activity to a service operation is done at task definition time.
- *An invocation of operation defined in a category of a service ontology.* The binding of this operation to an operation of a concrete service of the corresponding ontology is done at run-time. In this case, an activity represents a request for a service instead of an invocation of a service. Since, activities describing a user task are labeled with requests for services, concrete Web services belonging to the required service ontologies are selected during the execution of the composite task. Hence, it is possible to execute tasks in different ways by allocating different Web services to execute component activities in the task.

A composite state allows the nesting sub-tasks (represented as state-charts) inside a parent task. Transitions represent dependencies among the activities of a task (e.g., a

transition may represent that an activity a_1 should be executed after an activity a_2 or a_1 and a_2 should be executed in parallel). A simplified state chart diagram specifying a “Travel Planner” task is depicted in figure 3. In this task, a search is performed to find a flight reservation service. After that, if the flight reservation is successful, an AND state follows, in which a search hotel booking service is performed in parallel with an invocation of a car rental service, and finally a search for an entertainment is performed. Note that states `BookFlight` and `BookHotel` are labeled with requests for services whereas the state `RentCar` is labeled with an invocation to an actual service (called “Avis”). The latter invocation style is useful when the service to use for executing specific task is the preferred (e.g., `Avis` is the preferred car rental service in the country of destination by the user of the task).

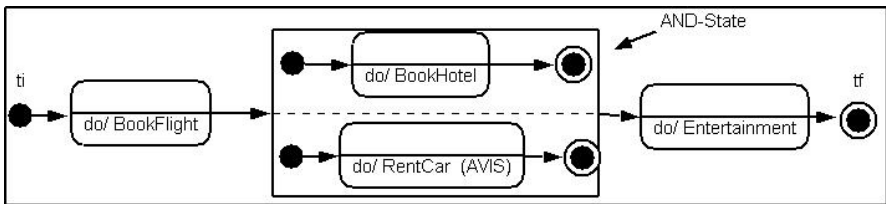


Fig. 3. State chart of the “Travel Planner”

Annotating Tasks with Context Information. To cater for context-aware service selection, in addition to the activities and their dependencies, a task definition includes context attributes from the context ontology. The administrator associates each task with its relevant contexts (e.g., for a travel booking task, the user’s timezone, local currency, type of Web browser, may be relevant). Therefore, when a user chooses a task to perform, the user agent is able to determine the contexts associated with the task and contacts the context agent to retrieve the values of each context attribute. For example, in the “Travel Planner” task, the administrator may choose the following relevant context attributes:

- for the state `BookFlight`, the attributes `preferences` of the context class `User` and `time` and `location` of the context class `Spatio-Temporal`. This may be needed because the user may have some preferences in the choice of airline company and this choice depends from time and location of this user;
- for the state `HotelBooking`, the attribute `noise` of the context class `ConditionalEnvironment`. This may be needed because the user may want, for example, a room with low noise level.

3.2 Provisioning Task

At a usage phase, a user chooses a task to perform from task repository (a repository maintained by the user agent). The user configures the required information to execute the task. In other words, user needs to specify a query that will be used by the system to select services to execute activities of the task. A user query is expressed in terms of attributes of service ontologies associated with the task. To simplify the

process of expressing queries, each task is associated with task service schema (service schema for short). Given a task definition, a service schema describes the attributes that can be seen as a global schema for selecting the services to execute such tasks. The attributes of such service schema are derived from the attributes, inputs and outputs of operations referenced in that task definition. In addition, since our approach caters for task context, a query is expanded by the user agent to specify the current context. The user agent interacts with the context agent to get the values of context attributes that are relevant to a given task.

Example. Assume, that the schema of the task shown in Figure 3 contains: (i) the context attributes preferences, time, location, temperature, (ii) the service attributes origin, destination, departureDate, returnDate, specialService, numberOfFlight, price, location, star, hasRestaurant, period. The user agent retrieves the values of the context agent from the context agent. For instance, the result of querying of the context agent can be: (preferences = "Austrian Airlines", time = "8:30 AM CET", location = "Lyon", noise = "no", temperature = "25°C"). After that, the user fills the value of service attributes if desired and the input/output of operations. For instance the user query in this case can be: (origin = "Lyon", destination = "Sydney", departureDate = "02/01/2007", returnDate = "04/03/2007", specialService = "seat far to window", numberOfFlight = "OS 402", price "1000,00€", location = "Randwick", star = "2", hasRestaurant = "no", period = "03/01/2007 – 03/03/2007").

4 Adviser Agent

The core idea of our services selection approach is to *recommend combinations* of services that are most appropriate to meet the user's needs in given contexts. The recommendations are based on the past execution history of a task (i.e., task memories). In this section, we define the notion of task memories and discuss how the adviser agent makes service selection recommendations. The issue of building a task memory will be discussed in the next section when we discuss the memory builder agent.

4.1 Task Memory

A task memory is associated with a specific task and it captures the information about the contexts and combinations of services that have been successfully used in the past to execute the task. It is a kind of a dynamic folder that associates contexts to combinations of services. Dynamic, here, means that the contexts and the combinations of services may evolve over time. In this way, service selection is not only based on the description or content of services but also on how likely they will be relevant in a given context. We represent a task memory as a table that has two attributes, namely, *context summary*, and *recommendations*.

Context Summary. Briefly stated, a context summary is a query representing a context that is considered by the system (or a system administrator) as relevant for selecting a combination of services to execute a task. It is specified using a conjunctive query of

atomic comparisons involving context attributes, service attributes, service operation inputs/outputs, and constants. The second column of the table 1 shows examples of context summary queries. While context summaries could be defined using sophisticated query languages such as XQuery or Xpath, without loss of generality, we choose to use a simplified representation model in terms of attribute/value comparisons for clarity of presentation. The concept of context summary allows capturing a set of possibly relevant contexts to effectively select services instead of encoding all possible service selection queries which may incur high performance cost. In other word, the notion of context summary allows the adviser agent to maintain a partial, concise and effective index of service selection queries.

Table 1. An example of task memory table

ID	CSQ	Combination GA
CSQ1	origin = 'Lyon' ^ destination = 'Sydney' ^ 100 < p < 250	{{(Quantas, Hilton), 0.6}, (Quantas, Paradise), 0.4}}
CSQ2	origin = 'Lyon' ^ destination = 'Hong Kong' ^ 100 < p < 250	{{(VolareWeb, Paradise), 0.7}, (Alitalia, Paradise), 0.75}}
CSQ3	origin = 'Milan' ^ destination = 'Sydney' ^ 300 < p < 500	{{(Alitalia, Hilton), 0.8}, (Alitalia, Paradise), 0.65}}

We assume that an administrator can identify a set of context summary queries that are relevant to a give task. This can be done by identifying a subset of attributes of the task schema that can be used to specify context summary queries. For each of these attributes, ranges of values are formed by dividing the domain of the attribute into a set of non-overlapping ranges known as Attribute Value Groups (AVGs). For nominal attributes, an AVG contains one or more distinct nominal values; for continuous attributes, an AVG specifies values range [5]. The union of AVGs of attribute is equivalent to the domain of the attribute. The Context Summary Queries (CSQs) are generated based on a cartesian product of these values. Table 2 lists examples of AVGs, assuming origin, destination, price, star, memoryFree and temperature are selected as summary attributes for the task "Travel Planner". The AVGs can be either manually defined by an administrator or discovered from query logs using query or context discovery techniques such as those presented in [13]. Once summary attributes are selected and AVGs are defined, the context summary queries are fixed.

Table 2. Example AVGs of summary attributes

Attribute	AVGs
Origin	'Lyon', 'Milan
Destination	'Sydney', 'Hong Kong'
Price	100 < p < 250, 300 < p < 500
Star	1 < s < 3, 4 < s < 5
Temperature (°C)	20 < t < 25, 4 < t < 7

Recommendations. For each context summary query, the task memory maintains the K ($K \geq 0$) most preferred combinations of services to execute a given task. Each services combination is associated with a positive weight value, called *Global Affinity*

(GA), exceeding a predefined threshold (parameter sets by a system administrator, for example). A task memory is represented by a table that has three columns: ID (identifier of context summary query), CQS (Context Summary Query), Combination_GA (combinations of services with their associated GAs). For instance, the column Combination_GA of Table 1 shows examples of service combinations and their associated GAs. The global affinity of a services combination measures the relevance of this combination to perform a task in a given context. More precisely, this value represents a weighted average of the values that measure the level of satisfaction of users, about a services combination, which respect to all the possible combinations in the that have been selected in that context. A more detailed description the notion of global affinity and its computation is given in [14].

4.2 Making Services Selection Recommendation

During services selection, in a response to a query from the UA, the AA identifies the potential combinations of services having answers to the query. The AA provides an operation called `recommendCombinations()`, that takes as input a selection query and returns a set of service combinations that can be potentially used to executed the corresponding task. The AA matches the user query against context summary queries of the corresponding task memory. The matching process relies on *subsumption* (containment) or equivalence between a user query and context summaries queries. If no combination is found to be appropriate based on the task memory, the AA forwards the query to a matching service engine to discover new possible combinations of services. For instance, the query Q: (category = “TravelToSydney”, attributes: origin, destination, price and values: origin = “not defined”, destination = Sydney, price = 150€) may not pass through the context filter of CSQ3 as the price is not included in its range. Hence any service associated to CQ2 is selected to answer to query Q, but CSQ1 can be used recommended to user.

5 Task Memory Builder Agent

The task of building a task memory is to associate context summary queries to combinations of services. This process is facilitated by a task memory builder agent (or simply builder agent). The Builder Agent (BA) is responsible for the incremental acquisition and to update of the elements the task memory table. Instead of asking an administrator to populate and update the task memory table, this agent incrementally captures the combinations of services that should be associated to context summary queries, by continuously monitoring how users use and rank services through interactions with the user agent.

This agent has access to operational knowledge such as service usage patterns as well as means for analyzing such patterns and updating task memories. There can be two approaches to build a task memory.

- A *lazy* approach consists to consider that the builder agent incrementally update a task memory starting from an initial table (e.g., an empty or a manually crafted table), during the service selection process. In this approach the builder agent

maintains a usage table that consists of context summary queries and their associated service combinations. The usage table contains only combinations that have been used at least once with satisfaction. Every time that a combination is used with satisfaction (respectively dissatisfaction) the associated global affinity will be upgraded (respectively, degraded).

- An *eager* approach that consists to periodically search for services usage patterns by calculating the global affinities of previously selected service combinations. This can be achieved for instance by a logging facility associated with the builder agent. The agent logs events related to services selection. The logged information could be analyzed in real-time (during services selection phase) or periodically to identify patterns that help updating the task memory. Then, each pattern is associated to a task memory update operation (e.g., adding a new combination of services).

More specifically, in the current architecture, the builder agent relies on the following building blocks to incrementally construct selection policies:

- *Logging service selection events.* Table 3 summarizes basic events that are logged by the builder agent. Over time, these events are used as a basis to identify service usage patterns (e.g., identify that a combination of services needs to associate to a given context because the number of times this combination was selected with satisfaction is greater than a given threshold).
- *Task memory table update operations.* Table 4 summarized main task memory update operations. The evolution of a task memory table is realized through update operations.
- *Task memory table update rules.* Table 5 summarizes the main update operations supported in our framework. Operations to perform for updating a task memory table as a result of the occurrence certain service usage pattern are captured using *Pattern Action* where *Pattern* is a condition over service selection events, and *Action* is a table update operation. More precisely, a condition of a rule is a sequence over service selection events. A rule is defined for each update operation.

In the lazy update strategy, whenever a combination is selected, the builder agent checks if an update rule can be triggered (i.e, checks if the associated event pattern is true, and eventually performs the rule action if true). In the eager strategy, the agent relies on a pre-defined rule triggering policy (e.g, specified by an administrator). For instance a triggering policy may say “analyze the logged events periodically (e.g, each 2 days) to detect the occurrence of event patterns” or “whenever the task

Table 3. Selected events supported in WS-Advisor

Events	Descriptions
services_selected (cs, cqs)	The combination of services cs is selected by the AA as a relevant candidate in a context identified by context query summary cqs
services_used (cs, cqs)	The combination of services cs was selected by the AA as a relevant candidate and used with satisfaction by the user in context identified by context query summary cqs
services_discarded used (cs, cqs)	The combination of services cs is selected by the AA as a relevant candidate but discarded by the user in context identified by context query summary cqs

Table 4. Selected operations supported in WS-Advisor

Operations	Description
Upgrade_Score (cs, cqs)	The GA of combination cs is upgraded with regard to context query summary cqs
Downgrade_Score (cs, cqs)	The GA of the combination cs with regard to context query summary cqs
Add_combination (cs, cqs)	A new combination cs is added and its score is initialized with regard to context query summary cqs
Remove_Combination (cs, cqs)	A combination cs is removed with regard to context query summary cqs

Table 5. Selected rules supported in WS-Advisor

Rules	Pattern	Action
Upgrade_Score_Rule (cs, cqs)	<services_selected (cs, cqs), services_used (cs, cqs)>	Upgrade_Score (cs, cqs)
Downgrade_Score_Rule (cs, cqs)	<services_selected (cs, cqs), services_discarded (cs, cqs)>	Downgrade_Score (cs, cqs)
Add_Combination (cs, cqs)	<services_selected (cs, cqs), services_used (cs, cqs)>	Add_combination (cs, cqs)
Remove_Combination (cs, cqs)	<services_selected (cs, cqs), services_discarded (cs, cqs)>	Remove_Combination (cs, cqs)

memory becomes a bottleneck.” Note that a task memory might become a bottleneck when the AA forwards the user query to a service matching engine frequently as the user is never happy with the recommendations of the AA or the agent does not find any relevant services combination.

6 WS-Advisor: Implementation Architecture

We adopt a layered architecture for the implementation of the whole WS-Advisor system. Figure 4 shows the elements of this architecture which are grouped into two layers: the agents layer and the infrastructure services layer. The agent layer consists of services implementing the user, adviser, builder, and context agents. The implementation of the agents is based on Java (using JADE platform), XML, and some generic services provider by the infrastructure layer of the architecture. In other words, all the agents are implemented as Java classes. The infrastructure services layer consists of generic services that we reuse from existing Web services environments to implement specific functionalities of the agents proposed in our approach.

The user agent provides a GUI to assist administrators in the creation and maintenance of tasks. It provides an editor for describing a statechart of a task. The editing process consists of annotating states of a task with services descriptions based on services ontologies. In addition, a task is also associated to a number of context attributes from the Context ontology. After the editing process, the user agent generates an XML file that represents a BPEL skeleton (a parametric process where invocations refer to service definitions instead of concrete services). The implementation of the task editor and the generation of BPEL process skeleton rely on the state-charts editor and BPEL process generation components of the Self-Serv Prototype [7]. The user agent also provides a GUI to assist users in browsing tasks, selecting services, and

execute tasks. It invokes the adviser agent to select services for executing a task. Once services are selected the user agent generates a BPEL executable process from the BPEL skeleton of the task and invokes a BPEL engine (ActiveBPEL) [15] to perform the execution of a task. The user agent provides means to inform the builder agent about the selected services.

The adviser agent provides methods for querying a Task Memory which represented as XML file. It also provides methods to query the service discovery engine. The service discovery engine facilitates the location Web services from external service registries. The implementation of this component relies on the services matching component of the WS-CatalogNet prototype [8]. The builder agent provides methods receiving notifications from the user agent, registering event patterns to the event monitoring service, and triggering actions for updating the task memory file. The event monitoring service is used for tracking and monitoring service usage and relies on the event management component of the WS-CatalogNet prototype. The context agent provides a method for querying context information. The implementation of this agent is a work in progress and will rely on the context service implemented in the PCAP prototype [7] which is an extension of Self-Serv to cater for context awareness in service oriented architectures.

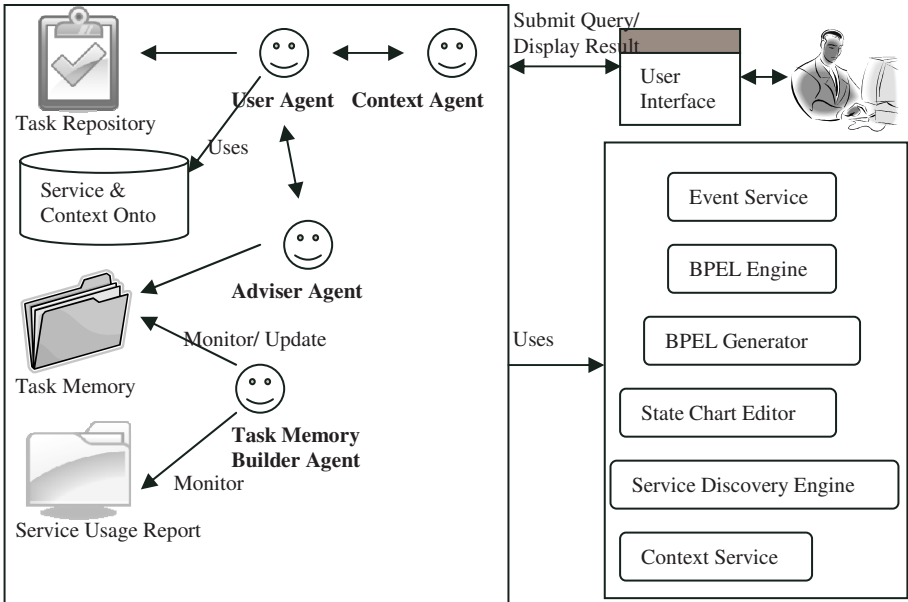


Fig. 4. WS-Advisor: Implementation Architecture

7 Discussion and Conclusions

A large body of research exists in the general area of web services discovery, selection, and composition. For example, early approaches based on the UDDI standard

provide limited services search facilities, supporting only keyword-based search of businesses, services, category names, and service identifiers [16]. To cope with this limitation, other approaches based on semantic web technology, and in particular web ontology languages such OWL-S, to support service description and discovery emerged. Main stream approaches in this area focus on description-based matchmaking techniques based on subsumption and equivalence relationships [17]. As pointed out before, other approaches leverage content summarization techniques to improve the accuracy of description-based services selection and matching approaches. It should be also noted that the problem of description based matching has also been addressed by several other research communities, e.g., federated databases, information retrieval, software reuse systems and multi-agent communities. More details about these approaches and their applicability in the context of the semantic Web services area can be found in [18] and in [19].

Our work is also related to the general area of recommender systems, especially those based on multi-agents. (e.g., Amalthea [20], SAGE [21]). These efforts focused on analyzing documents (e.g., web pages, email folders) to recommend relevant documents as in search engines or products as e-commerce systems. Efforts in this area build upon personalization techniques in Web applications including content-based, collaborative and rule-based filtering [22]. Other agent-based approaches catered for context awareness in the orchestration of interactions among components of a composite service [23]. Our work is complementary to efforts in user context modelling [10, 24, 25]. We focus on capturing task memories to allow effective services selection.

Our approach features embedding intelligence, consisting of task memories, into services composition frameworks allowing context-aware services selection. It builds upon ontology support as in web services, services discovery, selection and composition to develop a context-aware services recommender facility during execution of routine tasks. This approach is based on the observation that, in performing routine tasks, the service matching and selection component of service infrastructure may produce valuable information on the contexts in which combinations of services were considered most appropriate by users. This information can be helpful to users in selecting services to perform a task because sometimes users would select similar services in similar contexts. Unfortunately, this information, which we call task memory in our framework, is not effectively captured in existing service matching and selection approaches.

We use task memory during services selection to suggest most relevant candidate services. We proposed to use incremental acquisition techniques to build and update task memory. A task is associated to an agent that monitors how users use and rank services. We believe that the proposed approach is an essential ingredient that will work in a tandem with services discovery and selection cooperative service techniques to provide more personalized and context-aware selection of services. Ongoing work consists of extending the agent-based architecture presented in this paper to cater for collaboration among different users via social networks to share task memories. Our future work will focus on experimenting with the proposed approach using some case studies to test its validity in real settings. More specifically, we will investigate the validity of the assumptions and approaches related to context summary queries acquisition and global affinity computation. We also plan to investigate the use of incremental knowledge acquisition techniques as means to learn context summary queries.

References

1. Huhns, M.N., Singh, M.P.: *Service-Oriented Computing: Key Concepts and Principles*. IEEE Internet Computing 9, 75–81 (2005)
2. Teevan, J., Jones, W., Bederson, B., B.: Special issue on Personal information management, vol. 49 (2006)
3. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-Aware Middleware for Web Services Composition. IEEE Trans. Software Eng., 30, 311–327 (2004)
4. Medjahed, B., Bouguettaya, A.: A Dynamic Foundational Architecture for Semantic Web Services. Distributed and Parallel Databases 17, 179–206 (2005)
5. Sun, A., Benatallah, B., Hassan, M., Hacid, M.S.: Querying E-Catalogs Using Content Summaries. In: Cooperative Information System (2006)
6. Caverlee, J., Liu, L., Rocco, D.: Discovering and ranking web services with BASIL: a personalized approach with biased focus. ICSSOC, pp. 153–162 (2004)
7. Sheng, Z., Benatallah, B., Dumas, M., E., O.Y.: SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. In: Proc. of the 28th International Conference on Very Large Databases. Hong Kong, China. September (2002)
8. Baina, K., Benatallah, B., Paik, H., Rey, C., Toumani, F.: WS-CatalogNet: An Infrastructure for Creating, Peering, and Querying e-Catalog Communities. In: Proc. of the 30th International Conference on Very Large Databases. Toronto, Canada (2004)
9. Sheng, Q.: CompositeWeb Services Provisioning in Dynamic Environments. PhD thesis, School of Computer Science, University of New South Wales, Sydney, Australia (2005)
10. Dey, A. K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, College of Computing, Georgia Institute of Technology (2000)
11. Lei, H.: Context Awareness: a Practitioner's Perspective. In: IEEE International Workshop on Ubiquitous Data Management (UDM 2005), in conjunction with ICDE 2005, Tokyo, Japan, April (invited paper, 2005)
12. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. Technical Report GIT-GVU-99-22, GVU Center, Georgia Institute of Technology, June (1999)
13. Chakrabarti, K., Chaudhuri, S., Hwang, W., S.: Automatic categorization of query results. In: Proc. of ACM SIGMOD'04, Paris, France, June, pp. 755–766 (2004)
14. Bova, R., Hassas, S., Benbernou, S.: An Immune System-Inspired Approach for Composite Web Service Reuse. In: Int. Workshop of ECAI06 Artificial Intelligence for Service Composition. Riva del Garda, Trento, Italy (2006)
15. ActiveBPEL Engine <http://www.activebpel.org/>
16. Dustdar, S., Treiber, M.: A View Based Analysis on Web service Registries. In: Distributed and Parallel Databases, Springer, Heidelberg (2006)
17. Benatallah, B., Hacid, M., Leger, A., Rey, C., Toumani, F.: On automating Web services discovery. VLDB J. 14, 84–96 (2005)
18. Paolucci, M., Kawamura, T., Payne, T., R., Sycara, K., P.: Semantic Matching of Web Services Capabilities. In: International Semantic Web Conference, pp. 333–347 (2002)
19. Bernstein, A., Klein, M.: Towards High-Precision Service Retrieval. In: International Semantic Web Conference, pp. 84–101 (2002)
20. Moukas, A., Maes, P.: Amalthea: An Evolving Multi-Agent Information Filtering and Discovery System for the WWW. Journal of Autonomous Agents and Multi-Agent Systems 1(1), 59–88 (1998)

21. Blake, M.B., Kahan, D., R., Nowlan, M., F.: Context-aware agents for user-oriented web services discovery and execution. In: *Distributed and Parallel Databases*, Springer, Heidelberg (2006)
22. Paik, H.: *Community-Based Integration Adaptation of Electronic Catalogs*. PhD thesis, School of Computer Science, University of New South Wales, Sydney, Australia (2004)
23. Maamar, Z., Mostefaoui, S., M., Yahyaoui, H.: Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. In: *IEEE Transactions on Knowledge and Data Engineering* (2005)
24. Jovanovic, J., Knight, C., Gasevic, D., Richards G.: Learning Object Context on the Semantic Web. In: *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*. Netherlands. July 5-7 (2006)
25. McCalla, G.: The Ecological Approach to the Design of E-Learning Environments: Purpose-based Capture and Use of Information About Learners. *Journal of Interactive Media in Education* (2004)

A QoS Test-Bed Generator for Web Services

Antonia Bertolino, Guglielmo De Angelis, and Andrea Polini

Istituto di Scienza e Tecnologie dell'Informazione - CNR
Via Moruzzi 1, 56124 Pisa - Italy
{antonia.bertolino, guglielmo.deangelis,
andrea.polini}@isti.cnr.it

Abstract. In the last years both industry and academia have shown a great interest in ensuring consistent cooperation for business-critical services, with contractually agreed levels of Quality of Service. Service Level Agreement specifications as well as techniques for their evaluation are nowadays irremissible assets. This paper presents Puppet (Pick UP Performance Evaluation Test-bed), an approach and a tool for the automatic generation of test-beds to empirically evaluate the QoS features of a Web Service under development. Specifically, the generation exploits the information about the coordinating scenario (be it choreography or orchestration), the service description (WSDL) and the specification of the agreements (WS-Agreement).

1 Introduction

The attractive promise of the Service Oriented Architecture (SOA) paradigm is to enable the dynamic integration between applications belonging to different enterprises and globally distributed across heterogeneous networks. Within the SOA paradigm, the most concrete technology is today realized by the Web Services (WSs).

Although WSs constitute a quite new drift in software application development, research in this technology has already evolved through a few stages. Initially the focus was in mechanisms which could allow for the loose interconnection among services independently developed and implemented on different machines. Such vision can only be achieved through the disciplined usage of standard notations and protocols, and in fact the WS domain is characterized by a strong boost toward standardization. Thus key achievements at this stage have been the establishment of common service descriptions, the definition of open service directories for storing and retrieving such descriptions, and the enactment of dynamic discovering and binding mechanisms. The technology for basic WS interconnection is now well established, with WSDL, SOAP and UDDI being just the most representative elements.

Nevertheless, the need soon arose of allowing for more complex scenarios, beyond simple point-to-point interactions [1]. The standardization of adequate mechanisms for services composition and interaction constituted thus the next stage, which is still very active. Two directions currently lead the scene within two different, but related, contexts [2]: the first aims at defining the composition of services (referred to as *orchestration*), the second at describing how related services should cooperate to perform a given

task (*choreography*). Both interpretations of service integration provide a means to describe interacting scenarios. On one side, orchestration approaches foresee the availability of an execution engine that, by executing the code defining the orchestration, reproduces the specified interactions; as a fact, this need limits the applicability of orchestration to those cases in which a governing organization is in charge for defining the business process. In contrast, the choreography approach foresees the availability of a specification of the interactions to which the various services must conform, but it does not introduce *per se* any mechanism for forcing such interactions. Currently, the most significant proposals concerning the specification of WS orchestration and choreography are represented by the Business Process Execution Language (WSBPEL) [18] and the Web Services Choreography Description Language (WS-CDL) [22], respectively.

Eventually, the openness of the environment characterizing the SOA paradigm naturally led to the pursuit of mechanisms for specifying the provided levels of Quality of Service (QoS) and establishing an agreement on them, in line with the widely accepted idea that service delivery cannot just focus on functional aspects, and ignore QoS-related properties.

Indeed, not only for Service Oriented systems, but for many other kinds of enterprise applications [6] [20], communication networks and embedded systems [4], solutions that do not put adequate consideration of non functional aspects [17] are no longer acceptable. Correspondingly, in recent years much research has been devoted to methodologies for QoS evaluation, including *predictive* and *empirical* techniques [13]. Predictive approaches are crucial during the design and the development of a software system, to shape the quality of the final product [20]: they perform analytical QoS evaluation, based on suitable models, such as Petri Nets or Queueing Networks. But increasingly modern applications are deployed over complex platforms (i.e., the middleware), which introduce many factors influencing the QoS and not always easy to model in advance. In such cases, empirical approaches, i.e., evaluating the QoS via run-time measurement, could help smoothing platform-dependent noise. However, such approaches require the development of expensive and time consuming prototypes [15], on which representative benchmarks of the system in operation can be run.

In this last direction, however, when computer-processable specifications exist, and *code-factories* can be used to automatically generate a running prototype from a given specification, there is large room for the adoption of empirical approaches. In particular, and this is the position we take in this work, given the high availability of standardized computer processable information, WSs and related technologies [2][10][8][22][14] yield very promising opportunities for the application of empirical approaches to QoS evaluation.

According to this intuition, in this paper we introduce an approach, called Puppet (Pick UP Performance Evaluation Test-bed), which realizes the automatic derivation of test-beds for evaluating the desired QoS characteristics for a service under development, before it is deployed. In particular, we are interested in assessing that a specific service implementation can afford the required level of QoS (e.g., latency and reliability) when playing one of the roles in a specified choreography or when used in composition with other services (orchestration). To this purpose, Puppet relies on the availability of the QoS specification of both the service under evaluation and the interacting services. Such

assumption is in line with the increasing adoption of formal contracts to establish the mutual obligations among the involved parties and the guaranteed QoS parameters, which is referred to as the Service Level Agreement (SLA) for WSs.

In the next section we provide a basic background on the emerging languages for the definition of SLAs. Then, in Sec. 3 we illustrate the general scenario in which the Puppet tool should be employed. Successively, in Sec. 4 we describe the approach and its logical architecture. An exploratory example is presented in Sec. 5 while related work is summed up in Sec. 6. In Sec. 7 we draw conclusions and hint at future work.

2 Specification of Service Level Agreements

An important ingredient of the SOA paradigm is the QoS level agreement specifications among interacting services.

Traditionally, agreements were expressed informally, not in machine-readable form. In software engineering quite basic notions of agreements were established by means of Interface Description Languages [17]. Concerning the WS technology, Service Level Agreements (SLAs) represent instead one of the most interesting and actively pursued issues. SLAs aim at ensuring a consistent cooperation for business-critical services. Relevant experiences in this direction are certainly represented by work around Web Service Level Agreement (WSLA) [14] or SLAng [19].

The approach introduced in this paper has been conceived to be as independent as possible of a specific SLA language. Indeed, concerning the goal of the work, any SLA languages predicating on the concepts we are considering are equivalent. However, when it comes to developing a specific implementation of our conceptual environment, we obviously need to consider a specific technology. Hence, in the remainder of the paper, we will focus on a proof-of-concept development carried on using the WS-Agreement language [10]. To make the paper self-contained, we report below the background notions behind its current proposal.

WS-Agreement is a language defined by the Global Grid Forum (GGF) aiming at providing a standard layer to build agreement-driven SOAs. The main assets of the language concern the specification of domain-independent elements of a simple contracting process. Such generic definitions can be augmented with domain-specific concepts.

As shown in Fig. 1 the top-level structure of a WS-Agreements offer is expressed by means of a XML document which comprises the agreement descriptive information, the context it refers to and the definition of the agreement items.

The Context element is used to describe the involved parties and other aspects of an agreement not representing obligations of parties, such as its expiration date. An agreement can be defined for one or more contexts.

The defined consensus or obligations of a party core in a WS-Agreement specification are expressed by means of Terms. Special elements (e.g., AND/OR/XOR operators) can be used to combine terms, via the specification of alternative branches or the nesting within the terms of agreement.

The obligation terms are organized in two logical parts. The first specifies the involved services by means of the Service Description Terms. Such part primarily describes the functional aspects of a service that will be delivered under an agreement. A

```

<wsag:Agreement
  AgreementId=xsd:string>
  <wsag:Name>
    xs:NCName
  </wsag:Name>
  <wsag:AgreementContext>
    wsag:AgreementContextType
  </wsag:AgreementContext>
  <wsag:Terms>
    wsag:TermCompositorType
  </wsag:Terms>
</wsag:Agreement>

```

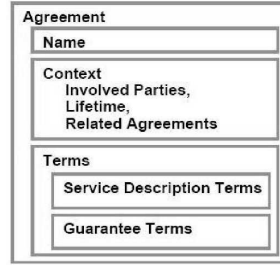


Fig. 1. WS-Agreement Structure

term for the service description is defined by means of its name, and the name of the service which it refers to. In some case, a domain-specific description of the service may be conditional to specific runtime constraints. A special kind of Service Description Terms is the *Service Reference*, which defines a pointer to a description of a service, rather than describing it explicitly into the agreement. The second part of the terms definition specifies measurable guarantees associated with the other terms in the agreement and that can be fulfilled or violated. A Guarantee Term definition consists of the obliged party (i.e., *Service Consumer*, *Service Provider*), the list of services this guarantee applies to (*Service Scope*), a boolean expression that defines under which condition the guarantee applies (*Qualifying Condition*), the actual assertion that have to be guaranteed over the service (*Service Level Objective*) and a set of business-related values (*Business Value List*) of the described agreement (i.e., importance, penalties, preferences). In general, the information contained into the fields of a Guarantee Term are expressed by means of domain-specific languages.

3 A WS Development and Evaluation Scenario

As explained in the Introduction, initially WSs were intended for loose and basic interactions, but soon the need for mechanisms to describe more complex integration of services emerged. The basic assumption of our approach is that such a description indeed exists. This is not an unrealistic assumption, as the global definition of applications resulting from the dynamic integration of unrelated services is seen as one of the most relevant factors at the basis of the take-off of the Service Oriented paradigm.

Our view¹ is that the integration of services offers major guarantees, and will be fostered, by the existence of predefined choreographies and the definition of orchestration. Given a definition of the integration of different services, based on choreography or orchestration, our objective is to provide a tool to support the QoS evaluation of services to be integrated, but still under development. The scenario we envisage is depicted in Fig. 2.

¹ This view is shared within the EU FP6 Strep n.26955 - PLASTIC, see at <http://www.ist-plastic.org>

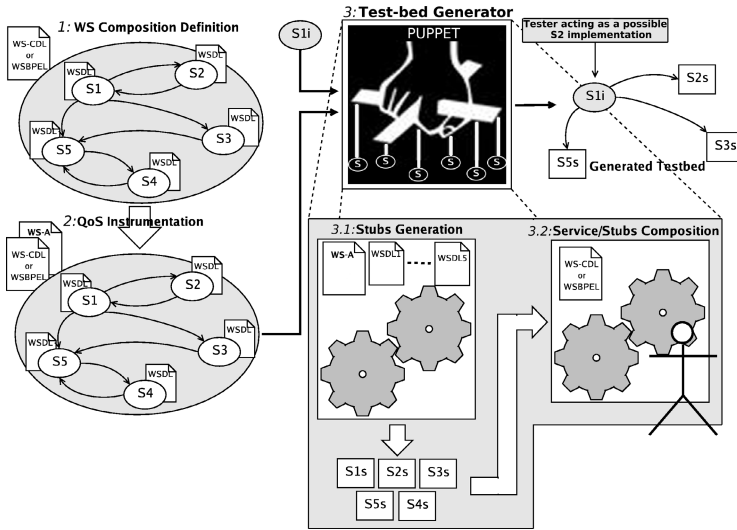


Fig. 2. The Puppet approach and supporting tool

The first step in this process, referred to as “1: WS Composition Definition” in Fig. 2 is indeed the specification of service integration, in terms of WS-CDL or WSBPEL, and of a set of WSDL descriptions defining the interfaces of the services involved in the interaction.

The process continues with the annotation of the composition with QoS attributes for each service involved in the integration. In Fig. 2 this step is referred to as “2: QoS instrumentation”. For this step we distinguish between the case of a choreography and that of an orchestration. In the former case, the organization that released the specification of the composition is in charge of augmenting the specification with QoS attributes. Developers of services will take such a specification as a reference for their implementation, expecting that their required services do the same. As a consequence each developer of a service is interested in evaluating that it actually can provide the required service according to the specified QoS. If this is not true we can expect that no other service in the given choreography will agree on binding to such service.

In the case of an orchestration, the derivation of QoS values for the services to be integrated is quite different. In this case the service that will be the subject of the validation is actually the orchestrated service. The developer company is interested in deriving a competitive service in terms of provided QoS. In this case the parameters for the QoS could not have been defined by someone else, i.e. a “standard” body as in the case of a choreography, but should be derived by the QoS defined by similar services possibly registered within a directory. Moreover in order to have a reliable picture of the final run-time environment the developer should also consider the QoS defined for the services that will be composed in the orchestration. Therefore retrieving from the directory

service the QoS specification for similar services, and the QoS provided by the composed services it is possible to get a quite trustable picture of the run-time conditions for the case of orchestrating services.

Summarizing, Puppet assumes the availability of a WSDL specification for each service, a definition of a composition in terms of WS-CDL or WSBPEL, and a WS-A description for the services in the composition/coordination. At this point, Puppet can automatically generate a test-bed to validate the implementation of a service before its deployment in the target environment (referred as “3: Test-bed Generator” in Fig. 2). The test-bed will consist of fake versions of the used services and of the possible clients. Such services are successively composed by the tester in order to reproduce different run-time conditions for the service under evaluation, as discussed in the following.

As illustrated in Fig. 2, step 3 consists of two different phases. The first one is the generation of the stubs simulating the non functional behavior of the services in the composition, and referred as “3.1: Stubs Generation” in Fig. 2. The second one, referred as “3.2: Service/Stubs Composition” in Fig. 2 foresees the composition of the implementation of a service, called “S1i” in Fig. 2 with the services with which it will interact. The next paragraphs provide a short introduction to both phases, that will be successively described in detail in Sec. 4.

The generation of the stubs proceeds through two successive steps (not detailed in the figure). In the first one a skeleton of the stubs is generated starting from the WSDL description. At this time the generated skeletons contain no behavior. Hence, in the second step the implementation is “filled” with some behavior that by construction fulfills the required non functional properties, for the corresponding service. The necessary information is retrieved from the WS-A specification and used to apply automatic code transformation according to rules that we have defined and we describe in Sec. 4. At the end of the “Stubs Generation” phase, a set of stubs providing the services specified in the composition according to the desired properties is available.

In turn Puppet permits also to derive stubs simulating the behaviour of possible clients for the service under evaluation as “S2” in Fig. 2. To generate such stubs Puppet considers the part of the WS-A document defining the constraints among the client and the service. Possible constraints can be for instance the number of invocations within a time frame, or a minimum time between successive invocations, and similar ones.

The “Service/Stubs Composition” phase implements the final setting of the test-bed. Goal of this step is to derive a complete environment in which to test the service. To this purpose, Puppet composes the service under test, “S1i” in Fig. 2 with the required services and according to the composition specified in the choreography or in the orchestration. At the same time client stubs are composed, by the tester, to derive a meaningful workload for the service under evaluation. Currently this phase requires the assistance of a human agent, as illustrated by the presence of a stick man in Fig. 2 that has to hand-code the composition in the service stubs. Nevertheless we are working on further automatizing the process on the base of the forthcoming final WS-CDL specification and WS-BPEL specification.

Concluding, the final product provided by the Puppet tool is an environment for the QoS validation of a composite service. The evaluation will require the development of a tester that integrates the client stubs generated by Puppet, in order to reproduce

meaningful workloads. Such a tool will have to verify that the properties specified in the QoS document for the service under evaluation are fulfilled. In Fig. 2 also such a tool is shown, nevertheless how this component can be derived is not within the scope of our work and we refer to the literature on the argument [8] for possible approaches.

4 Description of the Approach

The inspiring idea behind Puppet is that the technologies introduced within the WS infrastructure make it possible to automatically generate a test-bed environment for a service. The generated environment can then be used to test if the specified QoS properties (e.g. performance) will be respected by the service under development after its deployment in the final environment.

Specifically, the generation exploits the information about the coordinating scenario (be it choreography or orchestration), the service description (WSDL) and the specification of the agreements that the involved *roles* will abide (see Sec. 3). Tools and techniques for the automatic generation of service skeletons, taking as input the WSDL descriptions, are already available and well known in the Web Services communities [2]. Nevertheless such tools only generate an empty implementation of a service and do not add any logic to the service operations. Puppet exploits and improve those solutions by processing the empty implementation of a service operation and augmenting it with fake code resulting from the appropriate transformation of the SLA specification.

4.1 Skeleton Generation Process

In Puppet we automatically generate service stubs whose behaviors are derived from the terms defined in a WS-Agreement document. The UML Component Diagram in Fig. 3 outlines the architecture we propose. In the picture we directly refer to Apache-Tomcat/Axis [2] as the technology used for the derivation of the various intermediate artifacts needed for the derivation and deployment of the generated services stub. Nevertheless, the approach is not bound to a particular technology and other solutions are possible: the only requirement is to identify the corresponding tools in the chosen platform with respect to Apache-Tomcat/Axis. Any Web Services platform provides in fact some WSDL compiler permitting to automatically derive the different harness needed both for the deployment of the service and for enabling service clients to invoke the published service operations [1].

More specifically, the generation of a QoS stub service simulator for the service S1 in Fig. 3 undergoes three main phases: service skeleton structure definition, QoS behavior generation, service stub deployment.

The first step in the process is directly performed exploiting the Apache-Axis *WSDL-2Java* utility [2]. Such tool, taking as input a WSDL description of a service, generates a collection of Java classes and interfaces according to the abstract part of the specification. Thus, for each binding a service skeleton structure will be automatically defined and released. At the same time the tool generates both a deployment and an undeployment descriptors. Such descriptors can be identified by the extension WSDD (Web Service Deployment Descriptor) [2]. The deployment specification represents the contact

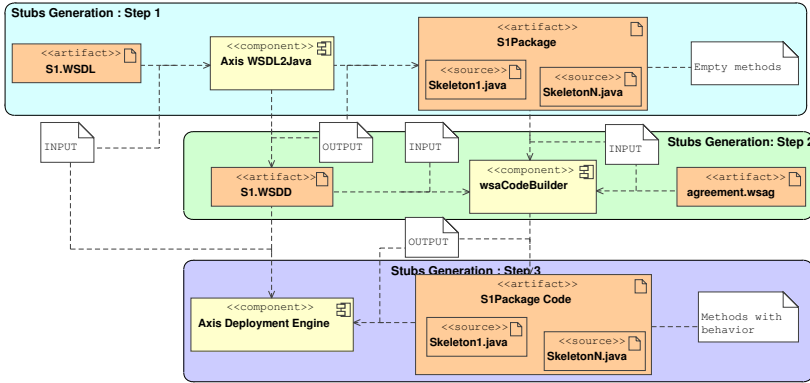


Fig. 3. Puppet Test-bed Generator Logical Architecture

point between the abstract definition of the service, expressed into the WSDL, and the corresponding concrete implementation of the endpoints coded into the Java skeletons.

Thus far, no behaviors are coded into the skeletons, but only the operations they export are derived. According to a set of specified transformation rules from WS-Agreement to Java and the relations in the deployment WSDD file, the `wsCodeBuilder` then generates the simulation code and inserts it into the proper operations. At the time of writing a running implementation of the `wsCodeBuilder` unit has been developed.

The last step of the process concerns the deployment of the services simulating the selected QoS. The deployment descriptor coming from the first phase and the new version of the skeletons are then used as input for the `Axis Deployment Engine`.

For the sake of completeness, it is important to remark that the generation of meaningful stubs would require to also handle their return values. Such values could in fact be part of a parametric QoS specification or influence the behavior of the tested service. The current implementation of Puppet does not provide a general solution, but returns values arbitrarily chosen among a set previously built for each possible data type. If no behavioral contract has been defined for the service this should be considered correct by the service under test. Nevertheless, within the WS community there is a great boost toward the definition of functional contracts for services [12] [5] [3]. To manage such situations, future releases of Puppet will integrate mechanisms that enable the instrumentation of stub services with code that returns conforming value with respect to the associated functional contract [21] (a related problem is that the determination of the return values must be low effort-intensive not to invalidate the evaluated QoS).

As above mentioned, the current version of the tool needs human support for the setting of the choreography. This means that the binding among services under development and stubs is manually derived from the WS-CDL or WSBPEL.

4.2 Matching WS-Agreement Statements to Java Code

Puppet can handle those QoS constraints that can be simulated by means of a parameterizable portion of code. The approach implemented in Puppet requires then that for

Table 1. Service Level Objective Mapping for Latency

<pre> ... <wsag:ServiceLevelObjective> <puppet:PuppetRoot> <puppet:Latency> <puppet:TagDelay> 1000 </puppet:TagDelay> <puppet:Distribution> normal </puppet:Distribution> </puppet:Latency> </puppet:PuppetRoot> </wsag:ServiceLevelObjective> ... </pre>	<pre> ... try{ Random rnd = new Random(); float val = rnd.nextFloat(); int sleepingPeriod = Math.round(val*1000); Thread.sleep(sleepingPeriod); } catch (InterruptedException e) {} ... </pre>
--	--

each concept in a SLA language a precise mapping must be provided. This is clearly a quite complex task; nevertheless given a specific language and a possible interpretation of the corresponding statements, it has to be done only once and for all. Currently we have defined a simple language for SLA that can predicate over several QoS characteristics (latency, reliability, workload) and have set a precise mapping for this concepts, defined in XML format, to composable Java code segments.

The mapping between the XML statements of the WS-A and the Java code has been specified in a parametric format that is instantiated each time one occurrence of the pattern appears. The examples reported in the remainder of this section show the transformations we have defined and encapsulated in Puppet.

In particular, conditions on latency can be simulated introducing *delay* instructions into the operation bodies of the services skeletons. For each Guarantee Term in a WS-Agreement document, information concerning the maximum service latency is defined as a Service Level Objective according to a prescribed syntax. The example in Tab. 1 reports the XML code for a maximum latency declaration of *1000mSec* normally distributed and the correspondent Java code that Puppet will automatically generate.

Table 2. Service Level Objective Mapping for Reliability

<pre> ... <wsag:ServiceLevelObjective> <puppet:PuppetRoot> <puppet:Reliability> <puppet:TagRate> 99.50 </puppet:TagRate> <puppet:Window> 2000 </puppet:Window> </puppet:Reliability> </puppet:PuppetRoot> </wsag:ServiceLevelObjective> ... </pre>	<pre> ... if (this.possibleFailureInWindow()){ Random rnd = new Random(); float val = rnd.nextFloat()*100; if (val>99.50f) { String fCode = "Server.NoService"; String fString="No target service to invoke!" org.apache.axis.AxisFault fault = new AxisFault(fCode, fString, "", null); this.incNumberOfFailure(); throw fault; } } ... </pre>
--	---

Constraints on services reliability can be declared by means of a percentage index into the Service Level Objective of a Guarantee Term. Such kind of QoS can be reproduced introducing code that simulates a service container failure. Thus, given the size of a sliding window defining the time interval within which reliability is measured, the generated stub for a service will raise remote exceptions according to the specified rate in the window. The XML code in Tab. 2 provides an example of the transformation for

reliability constraint description, assuming that the Apache-Tomcat/Axis [2] platform is used.

The case of workload can be simulated by equipping the generated skeletons with client-side code for the automatic invocation of the service under evaluation. Currently Puppet permits to describe the maximum number of requests that the clients can deliver to the service in a given period (i.e. WinSize in Tab. 3). The generation process augments the stubs with a private method for the remote invocation (i.e. invokeService in Tab. 3) and an exported public method that triggers the emulation request stream. The transformation in Tab. 3 reports the code for the trigger method. The information required for the instantiation of parameters such as the target endpoint is obtained from the part of the guarantee term concerning the scoping aspects discussed below.

According to what described in Sec. 2 a guarantee in a WS-Agreement document can be enforced under an optional condition. Such additional constraints are usually defined in terms of accomplishments that a service consumer must meet: for example the latency of a service can depend on the time or on the day in which the request is delivered. In these cases, the transformation function wraps the simulating behavior code-lines obtained from the Service Level Objective part with a conditional statement (see Tab. 4a).

As mentioned, the scope for a guarantee term describes the list of services to which it applies. In particular, Tab. 4b points out how to apply the term to a sub-set of the service operations. In this case, for each listed service, the transformation function adds the behavior previously obtained from the Service Level Objective and Qualifying Condition parts only to those operations declared in the scope.

5 Working Example

This section illustrates the application of Puppet to derive a test-bed to verify the QoS characteristics of one service when interacting with other services. The case study considered refers to an on-line booking of flights. Several clients access a Travel Operator

Table 3. Service Level Objective Mapping for Workload Generator

<pre> ... <wsag:ServiceLevelObjective> <puppet:PuppetRoot> <puppet:Workload> <puppet:NRequest> 20 </puppet:NRequest> <puppet:WinSize> 60000 </puppet:WinSize> </puppet:Workload> </puppet:PuppetRoot> </wsag:ServiceLevelObjective> ... </pre>	<pre> ... public void generateTraffic () throws MalformedURLException, RemoteException{ Random rnd = new Random(); int sleepPeriod; String endpoint="http://myhost/axis/services/"; String service="client"; String method="planJourney"; int winSize=60000; for (int i=0; i<20; i++){ this.invokeService(endpoint,service,method); sleepPeriod = rnd.nextInt(winSize); try { this.sleep(sleepPeriod); } catch (InterruptedException e) {} winSize = winSize - sleepPeriod; } ... </pre>
--	---

Table 4. More Mappings

<pre> ... <wsag:QualifyingCondition> <puppet:PuppetRoot> <puppet:Condition operator="Not"> <puppet:Var> interFly </puppet:Var> </puppet:Condition> </puppet:PuppetRoot> </wsag:QualifyingCondition> ... </pre>	<pre> ... if (!interFly){ try{ ... } } ... </pre>	a) Qualifying Condition
<pre> ... <wsag:ServiceScope wsag:ServiceName="ABS1"> <puppet:PuppetRoot> <puppet:Operation>checkFlight</puppet:Operation> </puppet:PuppetRoot> </wsag:ServiceScope> ... </pre>		b) Service Scope

Service (TOS) that in turn accesses different airline booking services (ABSs) to check the availability of a route for the required journeys. In the general case, different ABSs can provide the same functionality according to different QoS specifications.

In the scenario depicted in Fig. 4, clients invoking the TOS service have some restrictions due the maximum number of invocations that can be generated within a time frame. In particular it is supposed that clients cannot generate a workload higher than 20 invocations each 60 seconds. Furthermore, the TOS can interact with two different airline booking services providing different reliability and latency QoS agreements, in particular when invoked with request for intercontinental route.

In the example one the airline booking service (ABS₁) assures a latency of 15 seconds for checking seat availability on a specified flight. The company guarantees service replies reliable up to 99.5% of the requests per day. On the other hand the second airline service (ABS₂) declares to provide the same service in 10 seconds and a reliability of 98% every day. However, it is supposed that the company offering ABS₂ does not directly operate on intercontinental flights. If an international flight operation is required, ABS₂ could need to contact other airline partners providing segments of the selected journey. In these cases, the declared latency agreement reduces to 20 seconds.

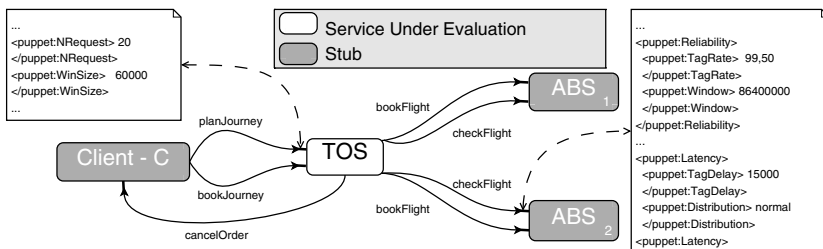


Fig. 4. Example Scenario

Table 5. ABS₂:*checkFlight* Generated Code

```

...
/** RELIABILITY EMULATOR CODE
 */
if (this.possibleFailureInWindow()){
    Random rnd = new Random();
    float val = rnd.nextFloat()*100;
    if ( val>98.00f){
        String fCode="Server.NoService";
        String fString =
            "No target service to invoke!"
        org.apache.axis.AxisFault fault=new
            AxisFault(fCode,fString,"",null);
        this.incNumberOfFailure();
        throw fault;
    }
}
/**QUALIFYING COND. GENERATED CODE*/
if (interFly){
    try{
        Random rnd = new Random();
        float val = rnd.nextFloat();
        int sleepingPeriod = Math.round(val*20000);
        Thread.sleep(sleepingPeriod);
    }
    catch (InterruptedException e) {}
}
/** QUALIFYING CONDITION GENERATED CODE */
if (!interFly){
    /** LANTENCY EMULATOR CODE */
    try{
        Random rnd = new Random();
        float val = rnd.nextFloat();
        int sleepingPeriod = Math.round(val*10000);
        Thread.sleep(sleepingPeriod);
    }
    catch (InterruptedException e) {}
}
...

```

Starting from this description Puppet is able to generate a set of stubs for the services interacting with the service under test. In particular Table 5 shows the derived source code for the method `checkFlight` provided by ABS₂ starting from the corresponding QoS values specified in Fig. 4 formatted according to the WS-Agreement specification as shown in Section 4.

Table 3 instead shows the code generated for the client stub. The generated client defines a method, to be used by the tester, that will raise the specified number of invocations within the specified time frame.

In the idea of Puppet the tester should combine all the generated services in different configurations, reproducing real run-time scenarios. Then launching the scenario it is possible to check if the specification of QoS defined for the Service under evaluation are respected and under which conditions. For instance the developer can try to understand which is the maximum number of clients that can be introduced in the scenario while still being able to abide by the specified QoS.

6 Related Work

Some years ago the authors of [9] recognized that the combination of testing and QoS evaluation, performance in particular, was a seldom explored path in software engineering research. Today the situation has changed and some interesting work starts to appear on this topic. The application of testing for QoS evaluation requires to solve two main problems. The first one is the derivation and simulation of an environment faithfully reproducing the final execution conditions. The second problem is the derivation of a testing suite representative of the real usage scenario, with particular reference to the specific property to be assessed.

This work is mainly related to the first point. Nevertheless the application of empirical approaches to QoS evaluation asks to solve both problems cited above. For works on how to automatically derive test cases, to be used for the successive evaluation of a system in a simulated environment, we refer to the literature, e.g., [9][6][7].

With reference to the generation of test-beds for the validation of WS QoS, we did not find many works. With reference to the area of Component-based software the work presented in [7] shows some similarities with what we propose here. Nevertheless the two approaches are quite different in their motivations and hypotheses. In [7] starting from the hypothesis that the middleware strongly influences the performance behavior of a deployed CBS, the authors generate an environment for the evaluation of the architecture of the system under development. The generated environment then is not intended to be used for the evaluation of a single real component implementation. Instead our target here is to develop a test-bed for the evaluation of a real implementation of a service. Moreover in [7] the stubs are directly derived starting from the architectural definition, no descriptions of QoS are considered (that work was mainly aimed at early performance evaluation). In our work instead, thanks to the availability of the QoS specification (such as the WS-A document), we can generate stubs behaving in accordance to what is defined in the corresponding WS-A document.

A work that has much in common with what we propose here is [11], in which a performance test-bed generator, in the domain of SOA, is presented. The approach proposed is structured in several steps. First, the service under development is described as a composition of services. Then, from this compositional model a collection of service stubs is generated. At this point for each service a description of the load that will be generated by possible clients is defined. From such descriptions, clients simulating the defined load are automatically developed. Finally, clients are executed in order to stress the service composition. The main differences with our proposal are twofold: on the one hand, the system in [11] makes no use of any kind of contract or agreement specification, differently from the approach we propose which is heavily based on the SLA. On the other hand, in [11] the service under development reacts to external stimuli generated by some clients according to the given load model. In our approach the service under development is considered plugged in a well specified choreography. Since none of the choreography members implementation is supposed to be available, our approach automatically builds an environment according to the specification of the coordination scenario.

Finally a main stream of SOA literature is devoted today to runtime evaluation of QoS by means of monitoring approaches. We do not tackle such related work here for size limitations.

7 Conclusions and Future Work

The paper proposed the Puppet approach for the automatic generation of test-beds to empirically pre-validate the QoS of a composite WS before it is deployed. To be applicable the approach requires the availability of the specification of the composition in which the service under evaluation will be inserted. We also assume that the composition is augmented with QoS properties, for instance expressed as a WS-A specification. QoS annotations of WS are not state-of-practice nowadays, nevertheless the relevance of this topic is raising fast in this domain. This is due to the fact that the SOA paradigm aims at removing the barrier among different organisations that can then directly cooperate. In a such scenario the reduced control over the required services certainly asks

for the introduction of agreements concerning the quality of non functional properties in addition to the functional behaviour.

Puppet requires first to define a precise mapping of the terms used for specifying QoS properties to simple parameterized Java code. This step gives a sort of semantic to each type of terms and is done once and for all. In some sense, it implicitly defines the simplest instance of a service providing the specified QoS. More complex definitions of QoS properties are obtained by composing terms of the QoS specification language. Thus, the translation function composes simple transformations according to the rules for the composition. In such a manner Puppet is able to generate service stubs according to composite QoS properties. These stubs can be used to validate possible real implementations of an under development service participating to the same coordination scenario.

The approach we are working on seems promising, nevertheless some issues remain open. Particularly interesting seems the generation of stubs that permit to return meaningful values without introducing complex code that could undermine the realization of stubs behaving in accordance to a specified QoS property.

The approach has been shown to be applicable and a tool is currently under finalization as an Eclipse plug-in. As described the tool will provide stubs mimicking real services, according to the corresponding QoS definition. To carry on reliable experiments the developer will have to distribute the stubs among various machines trying to reproduce as much as possible the final deployment environment. Another important factor strongly influencing the evaluation will be the definition of workload actually representing the final deployment condition. We are working on adding support for these steps that currently rely on human intervention to solve some technical issues. Nevertheless it is important to stress that the reproduction of a representative environment will never be achieved in a completely automated way.

Acknowledgements

The authors wish to thank Giovanni Possemato for his important contribution to the implementation of the tools enabling the proposed approach.

G. De Angelis PhD grant is sponsored by Ericsson Lab Italy in the framework of the PISATEL initiative (<http://www1.isti.cnr.it/ERI/>)

This work is partially supported by the PLASTIC Project (EU FP6 Strep No. 26955).

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web Services—Concepts, Architectures and Applications*. Springer-Verlag, Heidelberg (2004)
2. Apache Software Foundation. *Axis User's Guide*.
<http://ws.apache.org/axis/java/user-guide.html>.
3. Baresi, L., Ghezzi, C., Guinea, S.: Smart Monitors for Composed Services. In: *ICSOC 2004. Proc. 2nd Int. Conf. on Service Oriented Computing*, pp. 193–202. ACM Press, New York (2004)

4. Bertolino, A., Bonivento, A., De Angelis, G., Sangiovanni Vincentelli, A.: Modeling and Early Performance Estimation for Network Processor Applications. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) *MODELS 2006*. LNCS, vol. 4199, Springer, Heidelberg (2006)
5. Bertolino, A., Frantzen, L., Polini, A., Tretmans, J.: Audition of Web Services for Testing Conformance to Open Specified Protocols. In: Reussner, R., Stafford, J.A., Szyperski, C.A. (eds.) *Architecting Systems with Trustworthy Components*. LNCS, vol. 3938, Springer, Heidelberg (2006)
6. Bertolino, A., Mirandola, R.: Software Performance Engineering of Component-Based Systems. In: *WOSP 2004*, pp. 238–242. ACM Press, New York (2004)
7. Denaro, G., Polini, A., Emmerich, W.: Early Performance Testing of Distributed Software Applications. In: *WOSP 2004*, pp. 94–103. ACM Press, New York (2004)
8. Draheim, D., Grundy, J., Hosking, J., Lutteroth, C., Weber, G.: Realistic Load Testing of Web Applications. In: *Proc. Conf. on Software Maintenance and Reengineering*, pp. 57–70. IEEE Computer Society Press, Los Alamitos (2006)
9. Weyuker, E., Vokolos, F.: Experience with performance testing of software systems: Issues, and approach, and case study. *IEEE Transaction on Software Engineering* 26(12), 1147–1156 (2000)
10. Global Grid Forum: Web Services Agreement Specification (WS–Agreement), version 2005/09 (edn.) (September 2005)
11. Grundy, J., Hosking, J., Li, L., Liu, N.: Performance Engineering of Service Compositions. In: *SOSE 2006. Proc. Int. Workshop on Service-Oriented Software Engineering*, pp. 26–32. ACM Press, New York (2006)
12. Heckel, R., Lohmann, M.: Towards Contract-based Testing of Web Services. *Electronic Notes in Theoretical Computer Science* 116, 145–156 (2005)
13. Hrischuk, C.E., Rolia, J.A., Woodside, C.M.: Automatic Generation of a Software Performance Model Using an Object-Oriented Prototype. In: *MASCOTS 1995. Proc. 3rd Int. Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pp. 399–409. IEEE Computer Society Press, Los Alamitos (1995)
14. IBM. WSLA: Web Service Level Agreements, version: 1.0 revision: wsla-, 2003/01/28 edn. (2003)
15. Liu, Y., Gorton, I.: Accuracy of Performance Prediction for EJB Applications: A Statistical Analysis. In: Gschwind, T., Mascolo, C. (eds.) *SEM 2004*. LNCS, vol. 3437, pp. 185–198. Springer, Heidelberg (2005)
16. Liu, Y., Gorton, I., Liu, A., Jiang, N., Chen, S.: Designing a test suite for empirically-based middleware performance prediction. In: *CRPIT '02: Proc. 4th Int. Conf. on Tools Pacific*, pp. 123–130. ACS (2002)
17. Ludwig, H.: WS-Agreement Concepts and Use – Agreement-Based Service-Oriented Architectures. Technical report, IBM (May 2006)
18. OASIS. Web Services Business Process Execution Language (WSBPPEL) 2.0 (December 2005) http://www.oasis-open.org/committees/tc.home.php?wg_abbrev=wsbpel.
19. Skene, J., Lamanna, D.D., Emmerich, W.: Precise Service Level Agreements. In: *Proc. 26th Int. Conf. on Software Engineering (ICSE 2004)*, pp. 179–188 (2004)
20. Smith, C.U., Williams, L.: *Performance Solutions: A practical Guide To Creating Responsive, Scalable Software*. Addison–Wesley, London, UK (2001)
21. Tkachuk, O., Rajan, S.P.: Application of Automated Environment Generation to Commercial Software. In: *ISSTA 2006. Proc. ACM Int. Symp. on Sw Testing and Analysis*, pp. 203–214. ACM Press, New York (2006)
22. W3C. Web Services Choreography Description Language (WS–CDL) 1.0 (November 2005) <http://www.w3.org/TR/ws-cdl-10/>

Engineering Compensations in Web Service Environment

Michael Schäfer¹, Peter Dolog², and Wolfgang Nejdl¹

¹ L3S Research Center, University of Hannover,
Appelstr. 9a, D-30167 Hannover, Germany

Michael.Schaefer@stud.uni-hannover.de, nejdl@l3s.de

² Aalborg University, Department of Computer Science,
Fredrik Bajers Vej 7E, DK-9220 Aalborg East, Denmark
dolog@cs.aau.dk

Abstract. Business to business integration has recently been performed by employing Web service environments. Moreover, such environments are being provided by major players on the technology markets. Those environments are based on open specifications for transaction coordination. When a failure in such an environment occurs, a compensation can be initiated to recover from the failure. However, current environments have only limited capabilities for compensations, and are usually based on backward recovery. In this paper, we introduce an engineering approach and an environment to deal with advanced compensations based on forward recovery principles. We extend the existing Web service transaction coordination architecture and infrastructure in order to support flexible compensation operations. A contract-based approach is being used, which allows the specification of permitted compensations at runtime. We introduce the *abstract service* and *adapter* components which allow us to separate the compensation logic from the coordination logic. In this way, we can easily plug in or plug out different compensation strategies based on a specification language defined on top of basic compensation activities and complex compensation types. Experiments with our approach and environment show that such an approach to compensation is feasible and beneficial.

1 Introduction

The Web service environment has become the standard for Web applications supporting business to business transactions and user services. Processes such as payroll management or supply chain management are realized through Web services. In order to ensure that the results of the business transactions are consistent and valid, Web service coordination and transaction specifications [12,13,11] have been proposed. They provide the architecture and protocols that are required for transaction coordination of Web services.

The transaction compensation [7] is a replacement for an operation that was invoked but failed for some reason. The operation which replaces the original one either undoes the results of the original operation, or provides similar capabilities

as the original one. The notion of compensation was introduced for environments where the isolation property of transactions is relaxed but the atomicity needs to be maintained. Several protocols have been proposed to control transactional processes with compensations [20].

Current open specifications for transaction management in Web service environment provide only limited compensation capabilities [8]. In most cases, the handling of a service failure is restricted to *backward recovery* in order to maintain consistency, i.e. all running services are aborted, and all already performed operations are reversed [1]. This approach is very inflexible and can result in the abortion of many services and transactions. Especially if dependencies between multiple transactions exist, the failure of one service can lead to cascading compensations. Furthermore, current approaches do not allow any changes in a running transaction. If for example erroneous data was used in a part of a transaction, then the only possible course of action is to cancel the transaction and to restart it with correct data.

In this paper, we investigate an engineering approach for advanced compensation operations adopting *forward recovery* within Web service transactions. Forward recovery proactively changes the state and structure of a transaction after a service failure occurred, and thus enables the transaction to finish successfully. The main idea is the introduction of a new component called an *abstract service*, which functions as a mediator for compensations, and thus hides the logic behind the introduced compensations. Moreover, it specifies and manages potential replacements for primary Web services to be used within a transaction. The compensations are performed according to predefined rules, and are subject to contracts [14]. We introduce a framework based on the abstract services, which enables the compensations described in the compensation specifications.

Such a solution has the following advantages:

- Compensation strategies can be defined on both, the service provider and the client side. They utilize local knowledge (e.g. the provider of a service knows best if and how his service can be replaced in case of failure) and preferences, which increases the flexibility and efficiency.
- The environment can handle both, internally and externally triggered compensations.
- The client of a service is informed about complex compensation operations, which makes it possible to trigger additional compensations. Compensations can thus consist of multiple operations on different levels, and consistency is achieved through well defined communication protocols.
- By extending the already adopted Web service specification, it is not necessary to discontinue current practices if compensations are not required.
- The separation of the compensation logic from the coordination logic allows for a generic definition of compensation strategies, independent from the coordination specification currently in use. They are therefore more flexible and can easily be reused in a different context.

The rest of the paper is structured as follows. Section 2 introduces the motivating scenario, which will be used in the paper in order to exemplify the

concepts. Section 3 introduces the proposed design for an infrastructure that is able to handle internally and externally triggered compensations without transaction aborts, and describes the basic components and compensation specifications based on compensation activities and compensation types. A prototype implementation of the design is described in section 4, along with two experiments that use the new compensation capabilities. Section 5 reviews related work in the area of forward recovery. Section 6 concludes this paper and provides a direction for future work on this topic.

2 Motivating Scenario

The motivating scenario for this paper is a company’s monthly payroll processing. In order to introduce real-life dependencies, both, the company’s and the employee’s responsibilities are considered.

Company: In the first step of the payroll processing procedure, the company has to calculate the salary for each employee, which can depend on a multitude of factors like overtime hours or bonuses. In the next step, the payment of the salary is performed, which comprises several operations. First of all, the salary is transferred from the company’s account to the employee’s account. Then the company transfers the employee’s income tax to the account of the fiscal authorities. Finally, the company prints the payslip and sends it to the employee.

Employee: The employee has only one task which he has to perform each month in this scenario: He transfers the monthly instalment for his new car to the car dealer’s account.

The company’s and the employee’s operations are each controlled by a business process, and are implemented using Web services from multiple providers. The two business processes use transactions in order to guarantee a consistent execution of all required operations. This is depicted in Figure 1. Only the services of transaction T1 are shown.

It is obvious that there are multiple dependencies in this simple scenario, between and within these transactions. Therefore, it is vitally important that no

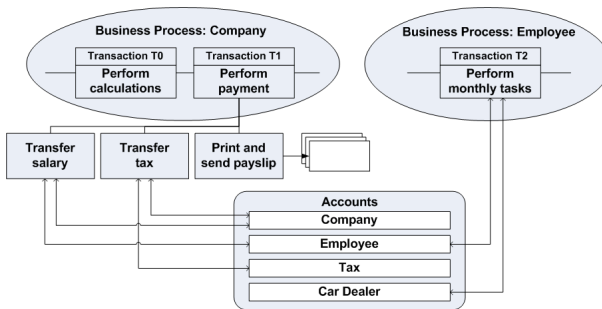


Fig. 1. The motivating scenario

transactions have to be aborted and compensated in order to avoid cascading compensations. However, such a situation can become necessary quite easily:

1. It can always happen that a service which participates in a transaction fails. Here, it could be that the service that handles the transfer of the salary fails due to an internal error. The transaction inevitably has to be aborted, even though the error might be easily compensatable by using a different service that can perform the same operation. Such a *replacement* is encouraged by the fact that usually multiple services exist that have the same capabilities.
2. A mistake has been made regarding the input data of an operation. In this scenario, it could be that the calculation of the salary is inaccurate, and too much has been transferred to the employee's account. The flaw is spotted by an administrator, but the only option is again to abort the complete transaction, although it would be very easy to correct the mistake by transferring the sum that has been paid too much back to the company's account.

Although it should be possible to handle these situations without the need to cancel and compensate the transaction(s), current technology does not allow to do so in a sensible way.

3 Web Service Environment with Transaction Coordination

We base our work on Web service coordination and transaction specifications [12,13,11]. These transaction specifications provide a conceptual model and architecture for environments where business activities performed by Web services are embedded into transactional contexts.

Figure 2 depicts an excerpt of such an environment with the main components. The client runs business activities A1 to A5, which are embedded in a transactional context. The transactional context and conversation is maintained by a transaction coordinator. Client and server stubs are responsible for getting

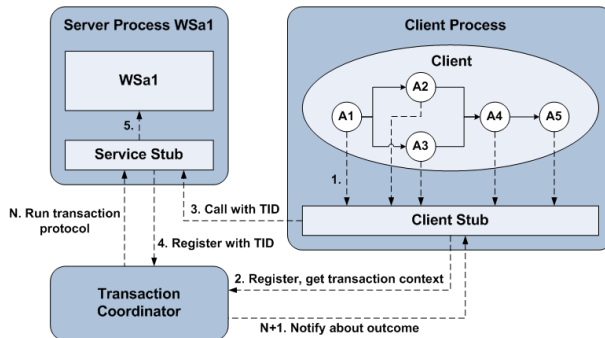


Fig. 2. Transactional environment for Web services adopted from [11]

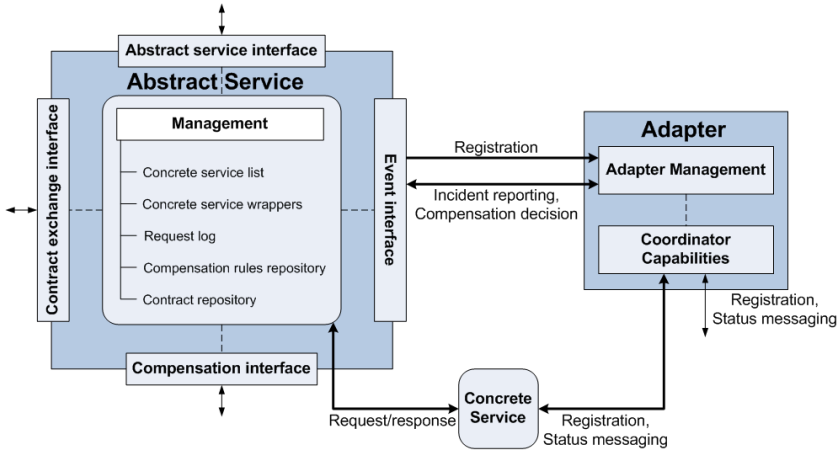


Fig. 3. The abstract service and adapter

and registering the activities and calls for Web services in the right context. The sequence of conversation messages is numbered. For clarity, we only show a conversation with a Web service provider that performs business activity A1. The transaction coordinator is then responsible for running appropriate protocols, such as two phase commit or some of the distributed protocols for Web service environments such as [2].

As pointed out above, the compensation capabilities are left to the client business activities according to the specifications in [12,13,11]. We extend the architecture and the infrastructure based on those specifications, so that it can handle internally and externally triggered compensations. Figure 3 depicts the extension to the transaction Web service environment, namely the *abstract service* and the *adapter* components. This extension does not change the way how client, transaction coordinators and Web service providers operate. Clients, instead of invoking concrete Web services, invoke abstract services which wrap several services and compensations for them. The adapter functions as a mediator between transaction coordinator, abstract service and concrete service to ensure proper transactional context.

3.1 Abstract Service

The central element of the extension is the notion of an *abstract service*. The client stub communicates with the Web service provider stub through the abstract service. An abstract service does not directly implement any operations, but rather functions as a management unit, which allows to:

- define a list of Web services which implement the required capabilities,
- invoke a service from the list in order to process requests which are sent to the abstract service,

- replace a failed service with another one from the list without a failure of the transaction, and
- process externally triggered compensations on the running transaction.

Distributed applications consisting of collaborating Web services have the advantage that normally single operations can be performed by multiple services from different providers. Which service will be chosen depends usually on the quality of service (QoS) requirements of the distributed application. The abstract service takes advantages of the existing diversity. To the outside, it provides an abstract interface and can be used like any other Web service, and uses the same mechanisms like SOAP [15] and WSDL [4]. On the inside, it manages a list of Web services (called *concrete services*) which provide the required capabilities. When the abstract service receives a request, it chooses one of these services and invokes it. Interface and data incompatibilities between the abstract interface and the interfaces of the concrete services are solved by predefined wrappers.

This approach has multiple benefits:

- Usually, a client does not care which specific service handles his requests, as long as the job will be done successfully and in accordance with the contract. The abstract service design supports this notion by providing the capabilities to separate the required abilities from the actual implementation.
- The available list of concrete services enables the abstract service to provide enhanced compensation possibilities.
- The definition of an abstract service can be done independently from the business process in which it will be used. It can therefore be reused in multiple applications without the need for changes. If a specific service implementation is no longer usable, then the business process does not have to be changed, as this is being managed in the abstract service.

Figure 3 depicts the basic structure of an abstract service. Four interfaces are supplied to the outside: The service operations for which the abstract service has been defined can be accessed via the *abstract service interface*. A contract can be exchanged or negotiated by using the *contract exchange interface*. Execution events of a service (e.g. a failure) can be signaled via the *event interface*. Compensations can be triggered from the outside using the *compensation interface*.

On the inside, the main component is the *management unit*, which receives and processes requests, selects and invokes concrete services, and handles compensations. In order to do so, it has several elements at its disposal:

- *Concrete service list*: Contains the details of all available concrete services.
- *Concrete service wrappers*: Define the mapping of the generic abstract service interface to the specific interface of each concrete service.
- *Request log*: Holds all requests of the current session.
- *Compensation rules repository*: Manages the rules that control the compensation handling process.
- *Contract repository*: Contains the existing contracts with the different clients.

3.2 Adapter

Abstract services could be used in conjunction with a wide variety of technologies. Therefore, it would be preferable if the definition of the abstract service itself could be generic. However, the participation in a transaction requires capabilities that are different for each transaction management specification.

That is why the transaction specific requirements are encapsulated in a so-called *adapter* (see Figure 3). An abstract service registers at this adapter, which in turn registers with the transaction coordinator. To the coordinator it looks as if the abstract service itself has registered and sends the status messages. When the abstract service invokes a concrete service, it forwards the information about the adapter, which functions as a coordinator for the service. The service registers accordingly at the adapter as a participant in the transaction.

As it can be seen, the adapter works as a mediator between the abstract service, the concrete service, and the transaction coordinator. The adapter receives all status messages from the concrete service and is thus able to process them before they reach the actual coordinator. Normal status messages can be forwarded directly to the coordinator, while the failure messages can initiate the internal compensation handling through the abstract service.

If the adapter receives such an error message, it informs the abstract service, which can then assess the possibility of compensation. The adapter will then be informed about the decision, and can act accordingly. If for example the replacement of a failed concrete service is possible, then the adapter will deregister this service and wait for the replacement to register. In this case, the failure message will not be forwarded to the transaction coordinator. The compensation assessment could of course also show that a compensation is not possible (or desirable). In such a case, the adapter will simply forward the failure message to the coordinator, which will subsequently initiate the abort of the transaction.

3.3 Compensation Specifications

Compensation specifications enable the abstract service to handle both kinds of compensations: Internally triggered compensations (arising from internal errors) and externally triggered compensations. An example for an externally triggered compensation could be the handling of the mistake spotted by an administrator as described in the motivation scenario section. We distinguish between *compensation activities* and *compensation types* in our compensation specifications, whose interaction are shown in Figure 4.

Basic Compensation Activities are the basic operations which can be used in a compensation. *ServiceReplacement* replaces the currently used Web service with a different one, which can offer the same capabilities and can thus act as a replacement. *LastRequestRepetition* resends the last request to the Web service. *PartialRequestRepetition* resends the last n requests from the request sequence of the current session (i.e. within the current transaction) to the Web service, while *AllRequestRepetition* resends all requests. *CompensationForwarding* forwards the

Nr.	Compensation Type	Compensation Activities									
		ServiceReplacement	LastRequestRepetition	PartialRequestRepetition	AllRequestRepetition	CompensationForwarding	AdditionalServiceInvocation	AdditionalRequestGeneration	ServiceAbortInitiation	RequestSequenceChange	ResultResending
01	NoCompensation										
02	Repetition		X								
03				X							X
04	Replacement	X	X								
05		X		X							X
06		X			X						X
07	Forwarding	(X)	(X)	(X)	(X)	X	(X)	(X)	(X)	(X)	(X)
08	AdditionalService						X				
09	AdditionalRequest							X			
10	SessionRestart				X				X	X	X

X Included compensation activity (X) Possibly included compensation activity

Fig. 4. The compensation types and their included activities

external compensation request to a different component, which will handle it. *AdditionalServiceInvocation* invokes an additional (external or internal) service, which performs some operation that is important for the compensation (e.g. the invocation of a logging service, which collects data about a specific kind of compensation). *AdditionalRequestGeneration* creates and sends an additional request to the Web service. Such a request is not influenced by the client, and the result will not be forwarded to the client. *ServiceAbortInitiation* cancels the operations on the Web service, i.e. the service aborts and reverses all operations which have been performed so far. *RequestSequenceChange* performs changes in the sequence of requests that have already been sent to the Web service. *ResultResending* sends new results for old requests, which have already returned results.

Compensation Types aggregate multiple compensation activities, and thus form complex compensation operations, as shown in Figure 4. These types are the compensation actions which can be used for internal and external compensations, and which form the basis of the compensation specification language. There are currently 7 different compensation types.

The most simple type is *NoCompensation*, which does not perform any operation. If a Web service fails, then this will be signaled to the transaction coordinator, which will initiate the transaction abort.

The *Repetition* type is important for the internal error handling, as it repeats the last request or the last n requests. The last request can for example be resend to a Web service after a response was not received within a timeout period. A

partial resend of n requests can for instance be necessary if the request which failed was part of a sequence, which has to be completely repeated after the failure of the final request. A partial repetition of requests will result in the resending of results for old requests to the client, which has to be able to process them.

The compensation type *Replacement* can be used if a Web service fails completely. It replaces the current service with a different one, and resends either all requests, a part of the requests, or only the last one. Resending only the last request is possible if a different instance of the service that has failed can be used as replacement, which works on the same local data and can therefore simply continue with the operations.

Forwarding is special in comparison with the other types, as it only indirectly uses the available activities. It forwards the handling of the compensation to a different component, which can potentially use each one of the compensation activities (which are therefore marked as "possibly included") in the process.

In an externally triggered compensation, it is sometimes necessary to invoke additional services and send additional requests to the concrete service. For this purpose, the compensation types *AdditionalService* and *AdditionalRequest* exist.

The final compensation type is *SessionRestart*. This operation is required if the external compensation request can not be handled without a restart of the complete session, i.e. the service has to be aborted and subsequently the complete request sequence has to be resend. The requested change will be realised by a change in the request sequence prior to the resending.

Compensation Protocol controls the compensation process and its interaction with the different participants. An externally triggered compensation always has the purpose of changing one particular request that has already been processed at the service. More specifically, the compensation request contains the original request with its data that has to be changed ($\text{request1}(\text{data1})$), and the new request-data (data2) to which the original request has to be changed to ($\text{request1}(\text{data2})$). The participants in the protocol are the *abstract service*, the *client* which uses the abstract service in its business process, the *initiator* which triggers the external compensation (either the client itself, or any other authorized source like an administrator), and the *transaction coordinator*. An externally triggered compensation can only be performed if the transaction in which the abstract service participates has not yet finished, as it usually has consequences for the client due to result resending.

The protocol consists of two stages. The first stage is the *compensation assessment*: As soon as the abstract service receives a request for a compensation, it checks whether it is feasible and what the costs would be. To that end, predefined compensation rules are being used, which consist of a *compensation condition* (defines when a compensation rule can be applied) and a *compensation plan* (defines the compensation actions that have to be performed). The second stage of the protocol is the *compensation execution*, which performs the actual compensation according to the plan. Whether this stage is actually reached depends on the initiator: After the assessment has been completed and has come to a

positive conclusion, the initiator, based on this data, has to decide whether the compensation should be performed or not.

As the client and the initiator of an external compensation can differ, the protocol contains the means to inform the client about the compensation process. It also ensures that the transaction coordinator is informed about the status of the external compensation, because the assessment and the execution stages have consequences for the abstract service's status in the transaction. While assessing the possibilities for a compensation, and while performing it, the abstract service can not process additional requests (and either has to store the requests in a queue, or has to reject them with an according error message). Moreover, its status can change as a result of a successful compensation.

3.4 Application on the Client and Provider Side

The abstract service design can be applied on both, the client and the provider side. A client which wants to create a new distributed application using services provided by multiple providers can utilize abstract services in two different ways:

1. The client can include the abstract service from a provider in its new business process, and can use the added capabilities.
2. The client can define a new abstract service, which manages multiple concrete services that can perform the same task.

The main goal of a Web service provider is a successful and stable execution of the client's requests in accordance with the contracts. If the service of a provider fails too often, he might face contractual penalties, or the client might change the provider. He can use abstract services in order to enhance the reliability and capability of his services by creating an abstract service which encapsulates multiple instances or versions of the same service. These can be used in case of errors to compensate the failure without the need for a transaction abort.

4 Discussion and Experiments

The described design approach has been used in a prototype implementation based on the scenario in section 2 and we performed two experiments with the implemented environment.

The four services participating in the payment transaction have been realized as abstract services. The abstract services manage the standard Web services performing the required operations as concrete services. The implementation has been done using Apache Tomcat as Web container, and Apache Axis as SOAP engine. The WS-Transaction specification has been chosen for the transaction coordination, more specifically the BusinessAgreementWithCoordinator-Completion protocol with the extension for transaction concurrency control that has been introduced in [2]. It is necessary for externally triggered compensations that the transaction coordinator is able to adapt to the changes that have to be performed in the process.

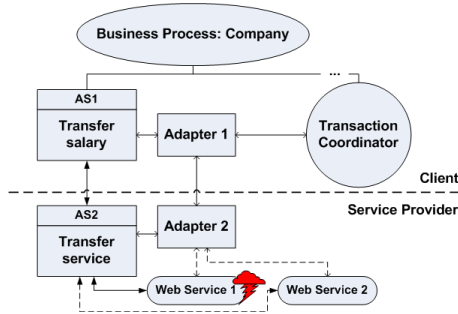


Fig. 5. Compensation on the provider side

The first experiment was devoted to the evaluation of the compensation of an internal service error. In this case, a failure of the concrete service on the provider side is simulated. Figure 5 shows the setup for the *transfer salary* operation: The abstract service *AS1* on the client side currently uses a concrete service that is itself an abstract service (*AS2*), which is operated by a service provider. The abstract service *AS2* uses *Web Service 1*, which performs the required operations. Figure 5 also depicts the interconnection of the services: *AS1* is registered as a participant at the *Transaction Coordinator* via *Adapter 1*, *AS2* is registered at *Adapter 1* via *Adapter 2*, and *Web Service 1* is registered at *Adapter 2*.

Now *Web Service 1* fails due to an internal error, and is thus not able to perform all operations required for the salary transfer. Instead of informing the transaction coordinator, abstract service *AS2* is informed, which assesses its compensation rules, the contract, and the available substitution services, and decides that a compensation is possible. *Web Service 1* is discarded and the request that failed is sent to *Web Service 2*, which registers at the adapter. *Web Service 2* is another instance of the same service, and can therefore simply continue with the request as it operates on the same local resources. This scenario shows that the signal of the service failure can be intercepted and the service replaced, without the need to cancel the complete transaction.

The second experiment evaluates an externally triggered compensation. Figure 1 summarizes the operations on the different accounts in the scenario described in section 2. In this experiment, an administrator has found an error in the calculation of the salary: The company transferred 50 units too much to the account of the employee. The administrator directly sends a compensation request to the abstract service that handles the salary transfer (*AS1*). The abstract service assesses the request by consulting its compensation rules. In this scenario, the rules specify that this compensation is only allowed if the employee's account would still be in credit after the additional debit operation, in order to avoid the employee's account being in debit after the transaction.

The result of the assessment is positive, which is reported to the administrator, who can decide based on this data whether the compensation should be performed. He decides that the compensation is necessary. The abstract service

Table 1. The transfer operations on the accounts in the scenario

Nr.	TRANSACTION	COMPANY (C)	EMPLOYEE (E)	TAX (T)	CAR DEALER (D)
		10.000	0	Y	Z
01	T1.debit(C,1.000)	9.000			
02	T1.credit(E,1.000)		1.000		
03	T1.debit(C,500)	8.500			
04	T1.credit(T,500)			Y+500	
05	T2.debit(E,150)		850		
06	T2.credit(D,150)				Z+150
		8.500	850	Y+500	Z+150

Table 2. The additional operations on the accounts

Nr.	TRANSACTION	COMPANY (C)	EMPLOYEE (E)	TAX (T)	CAR DEALER (D)
...
07	T1.debit(E,50)		800		
08	T1.credit(C,50)	8.550			
		8.550	800	Y+500	Z+150

compensates operations 01 and 02 from Table 1 by creating an additional debit and credit operation, as can be seen in Table 2. The operations transfer 50 from the employee's account back to the company's account, which thus compensates the initial problem. As an additional service, the abstract service initiates a precautionary phone call, which informs the employee about the change.

Subsequently, the compensation will be reported to the client, who has to assess whether any other services are affected according to its business process. It decides that the tax transfer does not have to be changed, while the payslip has to be updated, as the details of the salary have changed. The business process therefore initiates a compensation on the respective service, which handles this request by printing and mailing a new payslip. This shows that even the more complex initial problem could be solved without the need to abort the transaction.

These two experiments have shown that the proposed design is successful in employing flexible compensation strategies in Web service transactions. It is thus possible to develop more robust distributed applications, where the abstract services are able to adapt their compensation rules to the contract they have with the client. Especially in long-running transactions, this approach helps to avoid unnecessary transaction aborts, and therefore saves money and time. While it is of course still possible that the abstract service itself encounters an error, it at least provides the capabilities to avoid transaction aborts due to concrete service failures. Moreover, it is possible to mix the new design with existing technology: The new capabilities can be used, but do not have to be, as an abstract service can be employed like any other normal Web service.

However, the new functionality of the design with its advanced compensation abilities has its costs. By introducing additional components like the abstract

services and the adapters, the overall structure of a distributed application becomes more complex. The outsourcing of compensation logic to the abstract services simplifies the business process definitions, but at the same time the distributed compensation logic can make maintenance more difficult. And finally, the added components require additional messaging, and therefore the design increases the total number of messages that have to be sent.

The current implementation is a proof-of-concept of the proposed design architecture, and is still limited regarding certain aspects. The prototype of the abstract service uses only synchronous requests and does not allow parallel requests. Nevertheless, the same principles can be applied in this case, although additional request queue management will be required. Accordingly, the execution of compensation actions is currently performed only sequentially.

5 Related Work

Forward recovery can be realized by using dynamic workflow changes, as described in [17,19], which allow the semi-automatic adaptation of a workflow in case of errors. A change of the workflow process can for example consist of a deletion or jump instruction, or the insertion of a whole new process segment. The change can either be done on a running instance, or it can be performed on the scheme which controls the workflow, and which results in a change in all running instances. Refer to [18] for details. Although this approach is very powerful, it has two major disadvantages. Firstly, it is in most cases only possible to perform these adaptations semi-automatically. Changing a workflow requires a lot of knowledge about the process and the current state it is in, and the implications a change would have. Therefore, it is often necessary for a human administrator to specify and control the change. Secondly, these kinds of workflow changes require a very strict definition of the process, including for example data and control links. Ad-hoc changes of business processes with normal orchestration languages like WS-BPEL (see [6]) is very difficult [9]. [5] provides a mechanism to overcome this difficulty through a compensation handler. Our approach provides a more flexible solution for compensations orthogonal to the business processes, concrete services, and transaction coordination.

Our compensation approach can be used with the Enterprise Service Bus (ESB) [3], a powerful messaging infrastructure for business to business integration with Web services. The abstract service and adapter can be integrated through the ESB flexible extension mechanism. In this way, ESB can serve as a platform to exchange extended messages between business process, abstract services and adapters involved in the compensation conversation. Our approach can be used independently of ESB, employing ESB on top of the introduced infrastructure to integrate abstract services with workflow activities.

[16] introduces a notion of compensable Web services by specifying operations which can revert the execution. In our approach, we allow for a more complex specification of forward recovery compensations, which can be introduced at the client side, mediator side, as well as provider side. Two related approaches to a

flexible compensation mechanism for business processes are proposed in [2010]. In both cases, the focus is put on backward recovery. The compensation logic is treated as a part of coordination logic. In our approach, we separate the coordination from the compensation logic to provide for more flexibility.

6 Conclusions and Further Work

We have described a new design approach for complex compensation strategies in current transaction standards. Two new components have been described, the *abstract service*, which manages replacement services and compensation rules, and the *adapter*, which separates the coordination protocol specific functions from the generic definition of the abstract service. We have also presented the protocol that handles the assessment and processing of externally triggered compensations. The design and the protocol have been successfully validated in a prototype implementation.

Regarding future work, we plan to run additional experiments with different compensation scenarios. Moreover, it will be necessary to further analyze the impact of the new compensation capabilities on the business process definitions. At the moment, it is only assumed that the business process is able to adapt to the signaled compensations. It will be required to analyze possible extensions of existing orchestration languages like BPEL in order to include the new capabilities. The current implementation will be extended to support the management of parallel request processing, and the definition of compensation rules will be adapted accordingly.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services - Concepts, Architectures and Applications. Springer, Heidelberg (November 2003)
2. Alrifai, M., Dolog, P., Nejdl, W.: Transactions Concurrency Control in Web Service Environment. In: ECOWS '06. Proceedings of the European Conference on Web Services, Zurich, Switzerland, pp. 109–118. IEEE Computer Society Press, Los Alamitos, CA, USA (2006)
3. Chappell, D.A.: Enterprise Service Bus. O'Reilly Media, Inc., (2004)
4. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C note, W3C, March (2001)
5. Dobson, G.: Using ws-bpel to implement software fault tolerance for web services. In: EUROMICRO '06: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 126–133. IEEE Computer Society Press, Los Alamitos (2006)
6. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0 (Published online 2007) <http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf>
7. Gray, J.: The transaction concept: Virtues and limitations. In: VLDB 1981: Intl. Conference on Very Large Data Bases (1981)

8. Greenfield, P., Fekete, A., Jang, J., Kuo, D.: Compensation is not enough. In: EDOC 2003. 7th International Enterprise Distributed Object Computing Conference, Brisbane, Australia, pp. 232–239. IEEE Computer Society Press, Los Alamitos (2003)
9. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.P.: Extending bpmel for run time adaptability. In: Ninth IEEE International Enterprise Distributed Object Computing Conference EDOC, Enschede, The Netherlands (2005)
10. Lin, L., Liu, F.: Compensation with dependency in web services composition. In: NWeSP 2005. International Conference on Next Generation Web Services Practices, Seoul, Korea, pp. 183–188. IEEE Press, New York (August 2005)
11. Arjuna Technologies Ltd., BEA Systems, Hitachi Ltd., IBM Corporation, IONA Technologies, and Microsoft Corporation. Web Services Business Activity Framework, Published online (2005) <ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf>
12. Arjuna Technologies Ltd., BEA Systems, Hitachi Ltd., International Business Machines Corporation, IONA Technologies, and Microsoft Corporation. Web Services Coordination (Published online 2005) <ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf>
13. Arjuna Technologies Ltd., BEA Systems, Hitachi Ltd., International Business Machines Corporation, IONA Technologies, and Microsoft Corporation Inc. Web Services Atomic Transaction, Published online (2005) <ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf>
14. Meyer, B.: Applying "Design by Contract". IEEE Computer 25(10), 40–51 (1992)
15. Nielsen, H.F., Mendelsohn, N., Moreau, J.J., Gudgin, M., Hadley, M.: SOAP version 1.2 part 1: Messaging framework. W3C recommendation, W3C (June 2003)
16. Pires, P.F., Benevides, M.R.F., Mattoso, M.: Building reliable web services compositions. In: Aksit, M., Mezini, M., Unland, R. (eds.) NODE 2002. LNCS, vol. 2591, pp. 7–10. Springer, Heidelberg (2003)
17. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. Journal of Intelligent Information Systems 10(2), 93–129 (1998)
18. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive Process Management with ADEPT2. In: ICDE, pp. 1113–1114. IEEE, New York (2005)
19. Rinderle, S., Bassil, S., Reichert, M.: A Framework for Semantic Recovery Strategies in Case of Process Activity Failures. In: Manolopoulos, Y., Filipe, J., Constantinopoulos, P., Cordeiro, J. (eds.) ICEIS, pp. 136–143 (2006)
20. Yang, Z., Liu, C.: Implementing a flexible compensation mechanism for business processes in web service environment. In: ICWS '06. Intl. Conference on Web Services (2006)

Context-Aware Workflow Management*

Liliana Ardissono, Roberto Furnari, Anna Goy, Giovanna Petrone, and Marino Segnan

Dipartimento di Informatica - Università di Torino
Corso Svizzera 185, 10149 Torino - Italy
{liliana, furnari, goy, giovanna, marino}@di.unito.it

Abstract. We describe the CAWE framework for the management of context-aware workflow systems, based on Web Services. The framework is based on a hierarchical workflow representation supporting a synthetic and extensible specification of context-sensitive workflows, which can be executed by standard workflow engines. We have exploited the CAWE framework to develop a prototype application handling a medical guideline specifying the activities to be performed in order to monitor patients treated with blood thinners.

1 Introduction

We present the CAWE (Context-Aware Workflow Execution) framework for the development of context-aware applications based on Web Service technologies, which adapt to the user features and the execution context. In order to support a flexible adaptation of the business logic, we have adopted a workflow-based approach, which enables the declarative specification of such logic. The CAWE framework is based on a hierarchical representation of the workflow, which specifies alternative, context-dependent courses of action, and on a declarative specification of the conditions determining the runtime selection of the appropriate context-dependent part of the workflow.

In order to test the functionality offered by our framework, we developed a prototype system and we have instantiated it on the execution of medical workflows involving the cooperation of human actors playing different roles. The system may be accessed from the internet, by using a PC or a Smart Phone client. The resulting medical application supports the home assistance of patients affected by heart diseases and coordinates the activities of doctors, patient and other technical and administrative personnel during the execution of the clinical guidelines to be applied.

2 Dimensions in Context-Aware Adaptation

Often, context-aware applications proposing personalized products/services to their users do not consider (at least explicitly) the fact that there are two different types of *roles*: the beneficiary of the product/service (henceforth, the *service-user*) and the user who interacts with the system to get the product/service (*interacting-user*). For instance, in an adaptive e-commerce application, a person could buy a product on behalf

* This work is supported by the EU (project WS-Diamond, grant IST-516933) and by MIUR (project QuaDRAnTIS).

of somebody else [11]. Although the same person can play different roles, it is useful to distinguish them. In fact, both the service-user and the interacting-user can be the targets of adaptation, but the features to be taken into account, and what is adapted, differ in the two cases. For instance, a different product might be proposed, depending on the service-user's preferences; however, the User Interface (UI) should be adapted to the device utilized by the interacting-user to connect to the system. The CAWE framework supports UI adaptation, targeted to the interacting-user's context, and workflow adaptation, based on the context of the service-user; the latter is the focus of this paper.

2.1 UI and Workflow Adaptation in the CAWE Framework

In the CAWE framework, the interaction is tailored to the features of the device used to connect to the system (e.g., screen size), and to the role of the interacting-user (e.g., a relative or a doctor), by generating UI pages with different content and layout.

The adaptation of the workflow, described in detail in the following, consists of actuating different courses of action (activity paths) on the basis of the features of the context of execution. These context features belong to the service-user context. For instance, the clinical guideline of our medical application prescribes blood tests which can be performed at home or at the blood test lab, depending on the fact that the patient (service-user) is movable or not. The two alternative behaviors are associated to different activity paths, as the first one requires that a nurse visits the patient at home, while the second can be managed by taking the patient to the lab.

2.2 Service-User's Context in Our Medical Application

In order to provide a more concrete idea of the service-user context, we briefly describe its representation within our medical application. The service-user is the patient, who benefits from the home assistance service. The patient's context includes long-term and short-term information to separate stable data from features which might change during the service execution.

The *long-term* information includes:

(a) The *personal context*: this corresponds to the clinical record, which includes features such as the patient id, contact information, gender, age, prescribed therapy, date of the most recent blood test, and so forth.

(b) The *environment*: this includes features such as the available resources (i.e., the medical equipment available at home).

The *short-term* information includes:

(a) The *Personal context*: this includes physiological data (e.g., blood pressure, hearth pulsation, ...), the patient's mobility state (the "movable" feature specifies whether she can be transported by car, or she needs an ambulance), and the degree of urgency in treatments due to the patient's health state.

(b) The *Environment*: it includes the "social context" (i.e., the people who next to the patient), which can influence the course of action to be selected.

3 The CAWE Framework

The architecture of the CAWE framework includes various modules, wrapped by Web Service interfaces. Specifically, the context-aware workflow execution is supported by two modules: the Context-Aware Workflow Manager and the Context Manager. The Context-Aware Workflow Manager runs a workflow engine which executes the workflow activities by following the flow specification. However, when it encounters a context-dependent portion of the workflow, the engine invokes a Personalization Module which applies adaptation strategies to select the course of action to be performed. The Context Manager provides the other modules with contextual information during the execution of the application.

3.1 Workflow Representation

In order to represent the context-dependent parts of the workflow, we introduce an abstraction hierarchy whose higher-level elements describe the activities to be performed in generic way. Specifically, we introduce the concept of *abstract activity* to denote an activity schema which does not directly specify a piece of business logic of the application. The actions to be executed in order to complete the abstract activity are selected at runtime, depending on the context state.

Moreover, we define *abstract workflow* a workflow schema including at least one abstract activity: the workflow abstracts from the details of execution of at least one (context-dependent) activity. If a workflow does not contain any abstract activity, it represents a *concrete workflow* and it can be executed by the workflow engine without invoking the Personalization Module.

Finally, each abstract activity is associated with a set of *context-dependent implementations*, representing the alternative courses of action which the workflow engine should execute, depending on the context. Each context-dependent implementation is a (possibly abstract) workflow and represents a subprocess to be performed in order to achieve the results of the abstract activity.

Figure 1 depicts the abstract workflow of our medical application, which includes the following activities (abstract/concrete activity start with an uppercase/lowercase letters):

1. A doctor starts the workflow by setting the first blood test that the patient has to undergo (*setFirstBloodTest(patient, date)*).
2. The application books a blood test with a lab at the specified date (*BookBlood-Test(patient, date)*). Then, it evaluates the time interval between the current date and the date of the blood test (*eval(interval)*).
3. If the patient's health state is regular, she waits until the date of the test before doing any actions (*onAlarm(date)*). However, if warning symptoms, e.g., bleeding or fainting, occur before that date (*onMessage ...*), the urgency feature within the patient's context is set to "high" (*setUrgency(patient, "high")*), and the patient skips the wait. This is represented by means of a *pick* scope which includes the two competing courses of action.

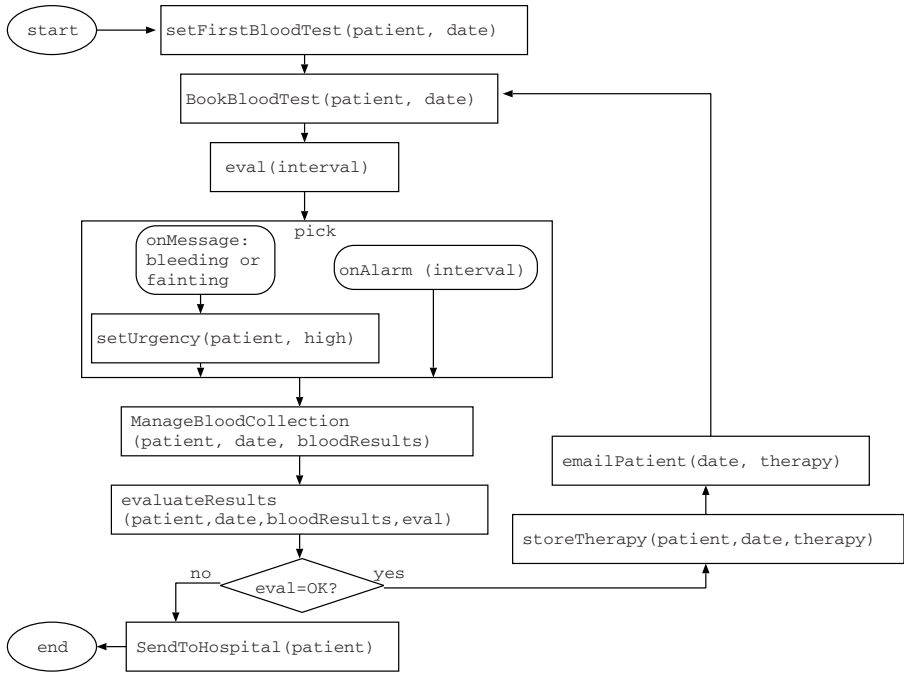


Fig. 1. Abstract Workflow of Medical Guideline

4. At the specified date, or immediately after the occurrence of a warning symptom, a sample of blood is taken from the patient for the lab. Then, the lab analyzes the blood sample and returns the values of the measured blood parameters (*ManageBloodCollection(patient, date, bloodResults)*).
5. A doctor evaluates the results that are then stored, together with the doctor evaluation, into the patient's context (*evaluateResults(patient, date, bloodResults, eval)*).
6. If the test results are good (*eval=OK?*), the doctor selects the date for the next blood test and sets the therapy to be followed until then (*storeTherapy(patient, date, therapy)*). Moreover, the application sends the patient an e-mail with the needed information (*emailPatient(date, therapy)*) and the flow restarts from item 2. Otherwise, the patient is notified to go to the hospital for further analyses (*SendToHospital(patient)*).

Figure 2 shows the workflow specification of a context-dependent implementation suitable for patients who may be taken to the lab by car. The applicability condition of this implementation is *movable=true*, where *movable* is a context variable included in the short-term patient's context. The application requests the appointment with the lab (*requestLabApp(patient, date)*), by invoking the Blood Test Lab Service¹, which returns a notification of the appointment, specifying time and place where to show up

¹ We assume that the actors involved in the workflow have Web Service interfaces.

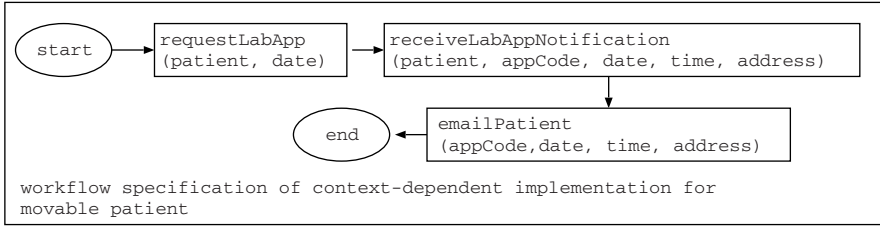


Fig. 2. Workflow Specifications of one of the Context-Dependent Implementations of Abstract Activity “BookBloodTest”

(*receiveLabAppNotification(patient, appCode, date, time, address)*). In turn, the application notifies the patient about the appointment (*emailPatient(appCode, date, time, address)*).²

3.2 Context-Aware Workflow Execution

The workflow engine wrapped by the Context-Aware Workflow Manager creates an instance of the abstract workflow each time the starting activity of the workflow is performed for the first time. The context-aware execution of the workflow is based on the selection of paths, depending on decision points and on the selection of the concrete implementations of the abstract activities to be performed. When the workflow engine has to execute an abstract activity, it first invokes the Personalization Module, which retrieves the context-dependent implementations of the abstract activity and returns the one to be executed, together with the bindings for its variables. Then, the workflow engine executes the selected context-dependent implementation as a subprocess of the main process instance. At subprocess completion, the control returns to the abstract workflow, which moves to the execution of the next activity.

The Personalization Module, invoked on an abstract activity, works as follows: first of all, if the abstract activity is associated to more than one context-dependent implementation, the module evaluates the applicability conditions of each candidate and it selects one of the applicable implementations for execution. Then, the module binds the context-dependent variables of the implementation to their current values, provided by the Context Manager.

4 Related Work

The introduction of context awareness in workflow systems is mainly focused on Quality of Service management (e.g., [2]) and on the adaptation to the user’s device (e.g., [4]). An interesting approach in the area of Web Engineering is presented in [3], where the authors propose an extension to WebML to model multi-channel, context-aware Web applications.

² Another context-dependent implementation, not shown for space reasons, specifies that, if the patient is non-movable, a nurse is requested, to collect the patient’s blood at home.

In this paper, we attempt to look at workflow systems from a more general perspective, and we propose a framework for the management of applications which adapt to user preferences and requirements, as well as to context-aware aspects such as the physical environment or the available resources.

The introduction of hierarchical workflows is usually related to the specification of compositional workflows. For instance, several workflow languages enable the designer to define complex activities which expand in workflows forming a composition hierarchy; e.g., see WS-BPEL [5] and process languages such as Petri Nets [6]. Our proposal differs from those approaches because it introduces a specialization hierarchy supporting the actuation of the same abstract activity in different ways.

5 Conclusions

We have presented the CAWE framework for the development of applications composing Web Service suppliers in complex, long-lasting workflows. CAWE supports the customization of the service offered by the application to the beneficiary of the service, taking the execution context into account; moreover, it personalizes the UI to the user(s) interacting with the application to carry out the assigned tasks.

The abstract activities and contextual conditions can be executed by a standard workflow engine because they simply include the invocation of a piece of code in their body. Indeed, the hierarchical workflow could be pre-processed by translating it to a flat workflow where abstract activities are replaced with decision points which fork on alternative workflow paths. However, our approach supports the conciseness and the readability of the resulting workflows, which could be very hard to understand, if represented as a flat graph including all the alternative courses of action. Moreover, it supports a seamless extension of the business logic of an application to take new contextual conditions and new courses of action into account. Finally, the introduction of the Personalization Module for the selection of context-dependent implementations supports the extension of the framework to the adoption of possibly complex personalization rules, without imposing restrictions on the workflow engines employed.

References

1. Ardissono, L., Goy, A.: Tailoring the interaction with users in Web stores. *User Modeling and User-Adapted Interaction* 10(4), 251–303 (2000)
2. Benlismane, D., Maamar, Z., Ghedira, C.: A view-based approach for tracking composite Web Services. In: *Proc. of European Conference on Web Services (ECOWS-05)*, Växjö, Sweden, pp. 170–179 (2005)
3. Ceri, S., Daniel, F., Matera, M.: Extending webml for modeling multi-channel contextaware web applications. In: *WISE - MMIS'03 IEEE Computer Society Workshop* (2003)
4. Keidl, M., Kemper, A.: Towards context-aware adaptable Web Services. In: *Proc. of 13th Int. World Wide Web Conference (WWW'2004)*, New York, pp. 55–65 (2004)
5. OASIS: OASIS Web Services Business Process Execution Language (2005), http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel
6. van der Aalst, W.: Making work flow: on the application of Petri Nets to Business Process Management. In: *Proc. of 23rd Int. Conf. on Applications and Theory of Petri Nets*, Adelaide, South Australia, pp. 1–22 (2002)

Practical Methods for Adapting Services Using Enterprise Service Bus^{*}

Hyun Jung La, Jeong Seop Bae, Soo Ho Chang, and Soo Dong Kim

Department of Computer Science
Soongsil University, Seoul, Korea

511 Sangdo-Dong, Dongjak-Ku, Seoul, Korea 156-743

{hjla, jsbae, shchang}@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

Abstract. In service-oriented computing (SOC), services are designed not just for a dedicated client but for a family of potential clients. For services to be generic and serviceable to different clients, service variability among the clients must be analyzed and modeled into service components. *Enterprise Service Bus (ESB)* is an architectural framework for service integration, but it does not provide effective adaptation mechanisms. Hence, it is desirable to devise techniques to adapt services on ESB for specific service requests. In this paper, we identify four types of service variability, and we present methods to adapt services provided on ESB. These methods can be practically applied in designing highly adaptable services on ESB.

1 Introduction

In service-oriented computing (SOC), services are designed not just for a dedicated client, but for a family of potential clients. A key problem in developing such common services is to model the variability embedded in common features, and to be able to adapt common services for specific service requests [1][2]. Another case requiring service adaption is when there is a *partial matching* between available services and services expected by clients. That is, an available service can potentially fulfill the service expected by a client, but they do not match in full [3]. Hence, identifying types of variability which may occur on services and how service variability can be modeled into adaptable services are two essential prerequisites to designing highly reusable and applicable services.

Enterprise Service Bus (ESB) is an architectural framework to integrate heterogeneous services and applications in distributed network [4]. It provides the integration functionality through transformation, communication, and routing. Beyond this, however, ESB does not provide methods for service adaptation. Hence, it is desirable to devise techniques to adapt services on ESB for specific service requests.

In this paper, we first identify four types of variability which may occur on services. Then, we present adaptation methods for services on ESB; *Workflow Mediator*, *Service Binder*, *Interface Transformer*, and *Logic Broker*. Each method is presented

^{*} This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (R01-2005-000-11215-0 (2007)).

with its overall scheme and a specific adaptation algorithm based on ESB specification. These methods can be practically applied in designing highly adaptable services on ESB.

2 Related Works

Sam's work proposes a customization framework for dynamic Web services [5]. Based on the comparison to syntactic/semantic aspects and to the constraint on input and output, it suggests a matchmaking process and conflict resolution mechanism for service adaptation. However, this work deals with only interface adaptation. Jiang's work proposes a categorization of variation points and introduces a pattern-based approach for managing the variation points [6]. However, instructions for identifying variability and adaptation are not included. Heralut's work proposes an approach to mediating and executing operations on ESB [7]. It explains how mediation can be achieved on ESB. However, how they mediation is performed is not given in enough details. Schmidt's work addresses a mediation model on ESB, which reconfigures the links between bus service providers and requesters to create dynamic alternations to routing and to modify their behaviors [8]. It treats interface mediations and policy mediations. However, practical adaptation mechanisms for these are not given.

From our survey, we observe that current research works treat service adaptation at conceptual level rather than design level, and adaptation methods on ESB treat only interface and logic variability. In this paper, we treat four types of service variability and suggest design-level practical methods for adapting services on ESB.

3 Types of Service Variability

In this section, we identify four places where service variability may occur [9].

Workflow Variability. A business process consists of a sequence of activities, i.e. *unit services* [10], and this sequence is called a *workflow*. For a given business process, the workflow may slightly vary, depending on different service clients. That is, some unit services in a workflow may be optional, and there can be more than one execution path for the given workflow.

Composition Variability. Services are discovered at runtime, and there may be more than one unit service which fulfills the required functionality. In this case, variation occurs on binding the right services. That is, for each specific request, one of the candidate unit services must be composed.

Interface Variability. *Variability on interfaces* occurs when the interfaces of unit services do not match to the interfaces of published services. Even if the functionality of a unit service is met by the functionality of a registered service, the signatures of interface and the semantics may not fully match. This should be modeled during service engineering, and the mismatch should be resolved by some interface adaptation mechanism.

Logic Variability. Service component includes operations for providing service functionality. There may be minor variation on the business logic or algorithm for the service component. This micro-level logic variability should be modeled into a service component, so that it can be tailored for each invocation.

4 Adaptation Managers for Services on EBS

In SOC, service clients do not have access to the internal details of services, and hence adaptation of a service is performed by an external software agent, which we call it *adaptation manager*. We propose a general scheme of designing ESB-based adaptation managers which can be used in resolving all four types of variability.

To implement adaptation managers on ESB, we define two kinds of components; *listener* and *adaptor*. As shown in Figure 1, a *listener* with «Listener» stereotype listens and intercepts service requests made by clients. And, it determines the required adaptation for each invocation, and invokes appropriate *adaptors*. The *adaptors* actually perform the requested adaptation over *service components* through *end-points* of ESB. Figure 1, shows four types of adaptors, which are denoted with «adaptor» stereotype.

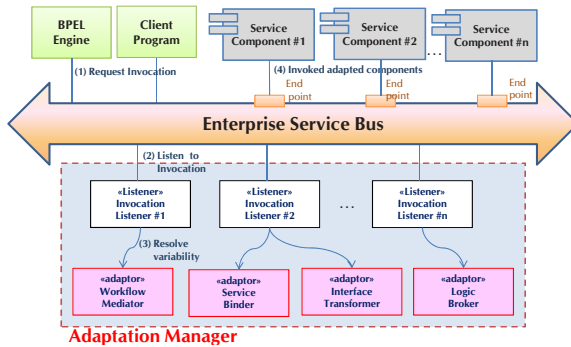


Fig. 1. Adaptation Manager deployed on ESB

4.1 Workflow Mediator for Workflow Variability

Workflow Mediator is to adapt the workflow of services, and it is implemented with *Mediator* pattern. When modeling workflow variability, variation points of workflow type and their relevant variants are identified and stored in the *workflow repository*.

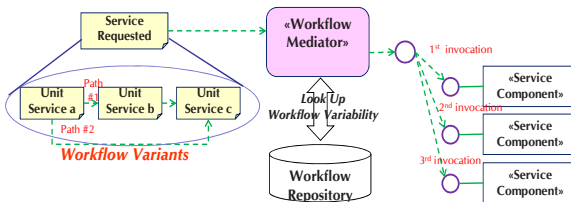


Fig. 2. Pattern of Workflow Mediator

As shown in Figure 2, the workflow mediator analyzes service requirements using user’s preference and context information, determines an appropriate *workflow variant* from the repository, and makes a series of invocations over the participating service components by using the rule repository.

We define two adaptation methods for workflow mediators as shown in Figure 3; *determineWorkflow()* and *controlServiceInvocation()*. The *determineWorkflow()* method is to select the most appropriate workflow from the repository based on the service requirements. This operation returns a path of the BPEL specification which will be executed. The *controlServiceInvocation()* method is to execute the workflow by invoking appropriate service components.

```

public String determineWorkflow(MessageExchange exchange) throws MessagingException {

    String BPELPath; // path of the BPEL documents
    Message sourceMsg = getSourceMsg(exchange); // to get source invocation message
    Vector workflowVariant = getWorkflowVariants(exchange); // to get candidate workflows

    // to get a particular workflow among candidate workflow based on the condition
    for ( int i = 0 ; i < workflowVariant.size() ; i++ ) {
        if (compare (workflowVariant[i], condition) == true) {
            BPELPath = workflowVariant[i].path;
            return BPELPath;
        }
    }
}

public void controlServiceInvocation(MessageExchange exchange) throws MessagingException {

    Message sourceMsg = getSourceMsg(exchange); // to get source invocation message
    Vector serviceComp = getServiceComponents (exchange); // to get the service components
    Rule rule = getRule (exchange) // to get the rule for the exchange

    // to invoke a service component by comparing when the service component is invoked
    for ( int i = 0 ; i < serviceComp.size() ; i++ ) {
        if (compare (serviceComp[i].case, rule.case) == true) CallComponents (serviceComp[i]
    }
}
    
```

Fig. 3. Algorithm of Workflow Mediator

4.2 Service Binder for Composition Variability

Service Binder is to adapt the service compositions, and it is implemented with *dynamic selection* pattern. When modeling composition variability, variation points of composition type and their variants are identified and stored in *composition repository*.

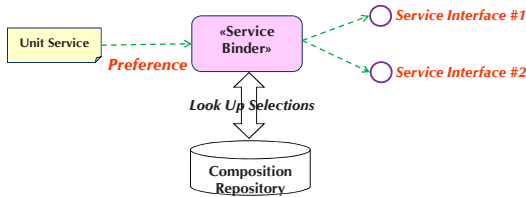


Fig. 4. Pattern of Service Binder

```

public String determineInterface (MessageExchange exchange) throw MessageException{

    Message sourceMsg= getSourceMsg(exchange); // to get source invocation message
    Vector endPoints = getEndPoint(exchange); // to get possible end points
    Rule rule = getRule (exchange); //to get selection rule for the exchange
    String targetEndPoint; //to save a target end point

    // to get an appropriate end point
    for (int i=0;i<endPoints.size(); i++) {
        if (compare(endPoints[i].case, rule.case) == true)
            targetEndPoint = (String)endPoint[i];
    }
}

public void bindInterface (Endpoint ed) throw MessageException{
    if (comparingInterface(sInvc, ed) == match) call(ed); // adaptation is not needed.
    else call(interfaceMediator, sourceMsg) // in order to adapt interface mismatch
}

```

Fig. 5. Algorithm of *Service Binder*

Figure 4 shows the relationship among the service binder, unit services, composition repository and WSDL service interfaces.

We define two methods for service binder; *determineInterface()*, and *bindInterface()*. The *determineInterface()* method is to select an *interface variant* in the repository based on user preferences, characteristics of services, and other context, and to generate a service endpoint type which is specific to the interface variant. The *bindInterface()* method is to bind the specified WSDL interface to the unit service. If interface mismatch occurs, it should be resolved with *Interface Transformer*.

4.3 Interface Transformer for Interface Variability

Interface Transformer is to adapt interfaces, and it is placed between unit services and service providers. When modeling interface variability, variation points of interfaces and relevant interface variants are identified and stored in the *Interface Transform Repository*. Each transformation method takes a service invocation of the interface specified by a unit service, transforms it into the published interface which eventually maps to the interface of a service component. It analyzes the service innovation, determines a transformation method, performs the transformation, and generates a new service invocation of a published interface.

4.4 Logic Broker for Logic Variability

Logic adaptor is a kind of adaption manager use the *plugin* method with *profiles* since the method makes the variable logics more decoupled [2]. When modeling logic variability, client profiles and objects which implement algorithms are identified and stored in *Client Profile* and *Logic Object Repository*.

Message listener recognizes the invocation from Web service client to the service interface written in WSDL. The listener invokes the *variability analyzer* so that the variability analyzer finds out corresponding adaptation types. Then, the listener invokes the service component with the object path through *object finder* so that the service component can use the logic object. For the logic variants of unknown and newly added unit services, new algorithm object should be deployed on the *object container*.

5 Conclusion

A key problem in developing reusable services in SOC is to identify the common features among potential clients and to model them into service components. In addition, *variability* within a common feature should also be modeled. Moreover, *partial matching* between available services and requested services should be identified can resolved. ESB provides an architectural framework for service integration, without providing practical adaptation mechanisms.

In this paper, we identified four types of variability which could occur on services. By extending software adaptation techniques and utilizing the key features of ESB, we presented practical methods to adapt services on ESB. Each method was presented with its overall scheme and a specific adaptation algorithm based on ESB specification. By using the methods, services with high adaptability and applicability on ESB can be effectively developed.

References

- [1] Kim, S., Her, J., Chang, S.: A Theoretical Foundation of Variability in Component-Based Development. *Information and Software Technology (IST)* 47, 663–673 (2005)
- [2] Chang, S.H., Kim, S.D.: A Systematic Approach to Service-Oriented Analysis and Design. In: the proceedings of the 8th International Conference on Product Focused Software Development and Process Improvement (PROFES) (to Appear)
- [3] Min, H., Choi, S., Kim, S.: Using Smart Connectors to Resolve Partial Matching Problems in COTS Component Acquisition. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) *CBSE 2004*. LNCS, vol. 3054, pp. 40–47. Springer, Heidelberg (2004)
- [4] Chappell, D.A.: *Enterprise Service Bus*, O'Reilly (2004)
- [5] Sam, Y., Boucelma, O., Hacid, M.: Web Services Customization: A Composition-based Approach. In: *ICWE'06*. proceedings of the International Conference on Web Engineering, IEEE Computer Society Press, Los Alamitos (2006)
- [6] Jiang, J., Ruokonen, A., Syata, T.: Pattern-base Variability Management in Web Service Development. In: *ECOWS '05*. proceedings of the Third European conference on Web Services, IEEE Computer Society Press, Los Alamitos (2005)
- [7] Heralut, C., Thomas, G., Lalanda, P.: Mediation and Enterprise Service Bus A position paper. In: the proceedings of the First International Workshop on Mediation in Semantic Web Services (*MEDIATE 2005*), pp.1-14 (2005)
- [8] Schmidt, M.T., Hutchison, B., Lambros, P., Phippen, R.: The Enterprise Service Bus: Making Service-oriented Architecture Real. *IBM Systems Journal* 4(4), 781–797 (2005)
- [9] Chang, S.H., La, H.J., Kim, S. D.: A Comprehensive Approach to Service Adaptation, *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)* (to Appear)
- [10] *OMG Business Process Management Initiative, Business Process Modeling Notation (BPMN) Version 1.0, OMG Final Adopted Specification (February 6, 2006)*

On the Quality of Navigation Models with Content-Modification Operations

Jordi Cabot¹, Jordi Ceballos¹, and Cristina Gómez²

¹ Estudis d'Informàtica i Multimèdia, Universitat Oberta de Catalunya
{jcabot, jceballos}@uoc.edu

² Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya
cristina@lsi.upc.edu

Abstract. Initially, web development methods focused on the generation of read-only web applications for browsing the data stored in relational database systems. Lately, many have evolved to include content-modification functionalities. As a consequence, we believe that existing quality properties for web model designs must be complemented with new property definitions. In particular, we propose two new quality properties that take the relationship between navigation models and the related data models into account. The properties check if navigation models include all necessary content-modification operations and whether all possible navigation paths modify the underlying data in a consistent way. In this paper, we show how to determine if a navigation model verifies both properties and also how to, given a data model, automatically generate a preliminary navigation model satisfying them.

1 Introduction

Many web development methods are evolving to cover the definition of full-fledged web applications, including data processing and manipulation functionalities. As a consequence, the models involved in the specification of a web application (that is, the data model to specify the data used by the application, the navigation model to describe the organization of the front-end interface and the presentation model to personalize its graphical aspect) have been extended with new modelling primitives.

One of the most relevant evolutions is the extension of navigation models with content-modification primitives that permit to modify the data managed by the web application. These primitives may be basic operations (insert/delete/update operations, as in WebML [4]) or references to more complex operations defined in the data model (as in OOWS [13]) or in a different model (as in the operational models proposed in [10]).

These new primitives complicate the definition of web model designs. Even for small web applications, data and navigation models can become huge and complex. Consequently, their definition is a time consuming and error prone process. This is a critical issue since their quality is very important, especially when web designs are used to automatically derive the implementation of the web application.

Up to now, quality properties for navigation models are based mainly on a purely syntax consistency analysis of the model structure (for instance, a common

verification is to check the *reachability* of all pages or that there are no *broken* links). These properties do not consider quality issues involving the new content modification operations that may appear in the navigation models. Therefore, existing properties are not suited to assess the quality of such models.

The main goal of this paper is to complement the existing set of quality properties defined for navigation models with two new quality properties that focus on the relationship between the content-management operations appearing in a navigation model and the data model specified for the same web application. Through these properties we can check early in the development process the quality of these extended navigation models. Additionally, we show how these properties help in the automatic generation of a preliminary navigation model once the related data model has been specified. This way we speed up the web development process because designers need not define the navigation model from scratch.

The first property we propose is the *completeness* of a navigation model with respect to its data model. We say that a navigation model is complete if the user can manipulate all the data underlying the web application by means of the modification operations included in the model (except for those parts of the data that the designer defines as derived or read-only). Incomplete navigation models result in web applications with data that a user can never modify. As an example, consider the partial data model shown in Fig. 1. Assuming that all elements of the data model are modifiable, a navigation model with a single page to modify sale objects is incomplete since users are unable to enter or modify sale lines.

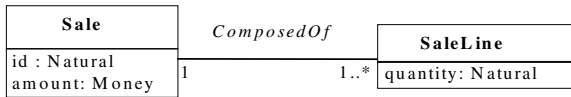


Fig. 1. Example of a data model

The second property is the *correctness* of a navigation model with respect to its data model. A navigation model is correct when all navigation paths admit at least one possible run (i.e. a run-time execution) that leaves the underlying data in a consistent state. Incorrect navigation models result in web applications with some paths that always end up in an inconsistent state. Every time the user interacts with the web application following such paths, an error arises and all user actions carried out until then must be rolled back (or repaired). Following the previous example, since sales must be composed of a minimum of one sale line, navigation models containing a path that permits to insert new sales but where no new sale lines can be created lead to a state where these sales always violate the minimum multiplicity of the *ComposedOf* relationship type. Therefore, such new sales must be discarded.

The rest of the paper is structured as follows. Section 2 reviews some basic concepts of data and navigation models. Section 3 formalizes our quality properties and characterizes the conditions that navigation models must satisfy in order to verify them. Then, Section 4 shows how these properties can be used to derive a preliminary complete and correct navigation model from an initial data model. Section 5 discusses the related work. Finally, Section 6 presents some conclusions and further work.

2 Basic Concepts of Data and Navigation Models

Web modelling languages provide several models to specify a web application. In this section, we review briefly the basic concepts and terminology of data and navigation models, which are the focus of this paper.

2.1 Data Model

A *data model* (also known as *content model*) defines the knowledge about the domain that a web application must have to perform its business functions. Fig. 2 shows an example data model, represented in UML, meant to (partially) model a simple e-commerce application. It contains information about sales, their sale lines and the products they contain. Sales may be associated with the customer purchasing them.

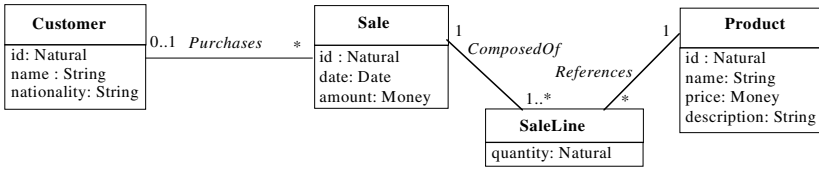


Fig. 2. The data model for the e-commerce application

The most basic constructors in data models are entity types (i.e. classes), relationship types (i.e. associations) and generalizations. Each *entity type* ET contains a set of *attributes*. For instance, in Fig. 2, *Sale* is an entity type with the attributes *id*, *date* and *amount*. Each binary *relationship type* RT has a name and two participants. A participant ET_i in RT may have a minimum and maximum cardinality, determining the minimum (resp. maximum) number of relationships (i.e. links) of RT in which ET_i may participate. We denote by $\min(ET_i, RT)$ and $\max(ET_i, RT)$ these cardinality constraints. For instance, *ComposedOf* states that each sale consists of at least one sale line so $\min(\text{Sale}, \text{ComposedOf})=1$. Each generalization, denoted by $\text{Gens}(ET, ET_1, \dots, ET_n)$, relate a supertype ET with a set of subtypes ET_1, \dots, ET_n . Generalizations may be disjoint and/or complete.

Additionally, the data model may include the definition of several operations to modify the state of the data. The basic operations (i.e., content-modification primitives) we consider are: $\text{InsertET}(x, v_1, \dots, v_n)$ (resp. $\text{DeleteET}(x)$) to perform the addition (removal) of the entity x into (from) entity type ET (optionally, attributes of x may be initialized with values v_1, \dots, v_n), $\text{UpdateA}_i\text{ET}(v, x)$ to set v as the new value for the attribute A_i of entity x and $\text{InsertRT}(x_1, x_2)$ (resp. $\text{DeleteRT}(x_1, x_2)$) to perform the addition (removal) of the fact that entities x_1, x_2 participate in an instance of RT . More complex operations can be defined as a sequence of these basic ones.

2.2 Navigation Model

A navigation model (also known as a *hypertext model*) specifies the organization of the front-end interfaces of a web application. Fig. 3 shows an excerpt of a possible

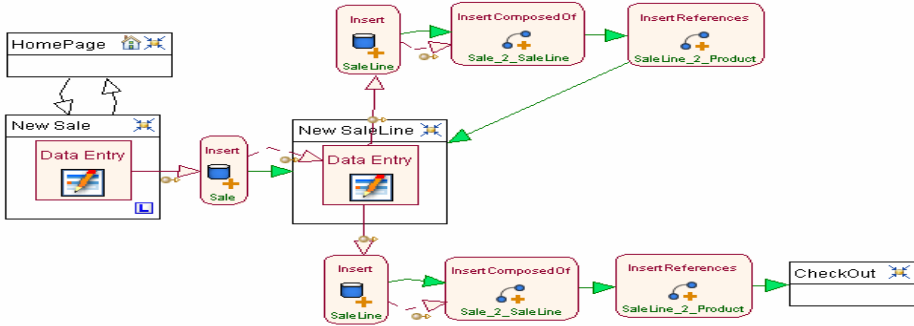


Fig. 3. A fragment of a navigational model for the e-commerce application

navigation model in WebML [4] for the e-commerce application presented in Fig. 2. The model shows the interface to create new sales and their related sale lines.

The most basic constructors of navigation models are pages and links. Pages may include several elements to specify the page contents. For practical purposes, our navigational models do not show the pages internal structure.

Many web modelling languages permit to define navigation models with content-modification operations that are executed as a result of browsing a link. As an example, Fig. 3 shows that when the user navigates to *NewSaleLine* from the *NewSale* page, the operation *InsertSale* is executed (using the parameters provided by the user in *NewSale*). In some languages these operations are simple create/update/delete operations equivalent to the basic operations defined above. Alternatively, other languages allow defining that browsing a link triggers the execution of a complex operation *op* defined in the data model (or in some other model). Then, the basic operations executed during the navigation are the ones specified in the definition of *op*.

Fig. 3 shows the process for creating a new sale. From the home page the user accesses the *NewSale* page. From here, the user may move to the *NewSaleLine* page or return to the *HomePage* again. During the navigation to *NewSaleLine* a new sale (empty or with the selected customer, not shown in the figure) is created because of the *InsertSale* operation attached to the link. In this page, the user selects the product to buy and indicates the quantity. Then, the user may either navigate to the *Checkout* page (the new sale line and the connections between the line and the sale and between the line and the selected product will be created when browsing the link) or to buy additional products by following the link leading to the *NewSaleLine* page again.

3 Complete and Correct Navigation Models

In this section, a navigation model N is formalized as a graph G_N (section 3.1) in order to check whether N satisfies the completeness (section 3.2) and correctness (section 3.3) properties with respect to its corresponding data model D .

3.1 Graph Representation

Given a navigation model N , the graph $G_N = (V_N, A_N)$ is obtained by means of the following rules:

- Every page in N is a vertex in V_N .
- Every link in N from a page X to a page Y is an arc from X (i.e. from the vertex representing X in G_N) to Y in A_N .
- The label of an arc a stores the (possibly empty) ordered sequence of basic operations associated to the link l represented by a in G_N .

Note that G_N is a directed graph (digraph), since being able to navigate from a page X to a page Y does not imply that the navigation from Y to X is also possible. Sometimes G_N turns out to be a multigraph [3] since it may contain multiple arcs with the same orientation between a pair of vertices v_1 and v_2 . This happens when the page corresponding to v_1 contains several links targeting the page represented by v_2 .

Fig. 4 shows the graph corresponding to the navigation model of Fig. 3

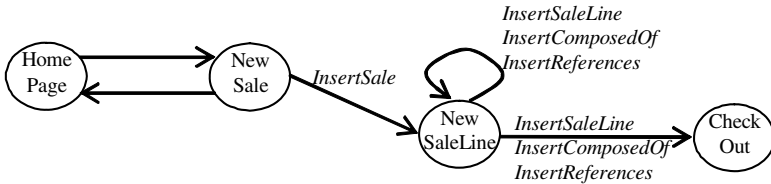


Fig. 4. Graph definition for the navigation model of Fig. 3

3.2 Completeness of a Navigation Model

Intuitively, a navigation model N is complete when the users of the web application may perform all basic operations¹ over the modifiable² elements of the corresponding data model D through interacting with the set of pages in N . Incomplete navigation models result in web applications with parts of the data that users cannot modify.

The set of basic operations that N must contain are the ones explicitly provided by the designer in D (or in some additional model [10]). If no operations are provided, this set of necessary basic operations may be automatically generated from D using a simple set of rules. For instance, we could generate an *InsertET* operation for each entity type ET in D , an *UpdateA_iET* operation for each attribute A_i of ET in D and so forth. We denote by set_{op} the set of operations (either defined or generated) for D .

Definition 3.2.1. N is complete when, for each operation op in set_{op} , it exists, at least, an arc a in A_N where $op \in label(a)$ ³.

¹ When different user groups or roles are defined, we require that at least one of them can perform such an operation.

² Designers can mark parts of the data model as *read-only* or *derived*.

³ $Label(a)$ returns the ordered sequence of operations associated to the arc a .

As an example, the graph of Fig. 4 is an incomplete navigation model regarding the data model of Fig. 2 since, for instance, basic operations to create customers and products are missing (no arc contains those operations).

Definition 3.2.2. N is minimal when it is complete and, for each operation op in set_{op} , there is a single arc a in A_N satisfying that $op \in label(a)$.

It is worth noting that non minimal navigation models may be useful. The designer may decide on purpose to offer several alternatives (i.e. several navigation paths) to execute the same kind of modification in the web application. However, we believe it is worth detecting these cases so that the designer can review and validate them.

Given the previous definitions, verification that a given navigation model N is complete (or minimal) is quite straightforward, we just need to generate the graph G_N for N and check if definition 3.2.1 (or definition 3.2.2) is satisfied by G_N .

3.3 Correctness of a Navigational Model

A navigation model defines the possible navigation paths permitted in the web application. Each navigation path admits several runs (i.e. run-time executions). Each run represents a possible interaction scenario between a user and the application. During a run several modification operations may be applied over the population of the entity and relationships types defined in the data model.

In some particular executions, these operations may turn the data into an inconsistent state (a state where some integrity constraint defined in the data model is not satisfied). This may happen, for instance, when the user does not enter appropriate values in the forms of the pages visited during the navigation. In such cases, all changes performed during the run must be discarded.

It may happen that all possible runs following the same navigation path fail (i.e. leave the data in an inconsistent state due to the execution of the basic operations included in the path). Clearly, such navigation path is completely useless and should be disabled in order to improve the performance and the usability of the web application.

As an example, consider the graph of Fig. 5 representing a simple navigation model consisting of a home page and a page for deleting existing sales. The user selects the sale to be deleted and then browses a link that deletes the sale and returns to the same page again so that additional sales can be deleted. According to the multiplicities of the relationship type *ComposedOf* (Fig. 2), all sales must be related with at least one sale line and all sales lines must be related to a sale. Therefore, when, after deleting a sale, we do not delete the associated sale lines as well (or assign those sale lines to a different sale) these multiplicity constraints will be violated. Hence, every single time the user tries to interact with this navigation model, an error will be raised⁴, independently of the sale the user selects.

Intuitively, correct navigation models are those that do not include navigation paths that always (i.e. for all possible runs) lead to an inconsistent data state, regardless of the parameter values the users provide during the interaction with the pages in the path.

⁴ Obviously, for this particular example we could define the database so that the sale deletion removes all related sale lines in cascade. However, since this information is not expressed in the model, this must be manually done after the initial code-generation.

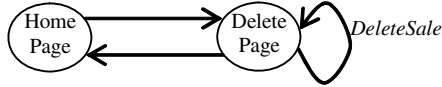


Fig. 5. Graph definition for deleting sales

Definition 3.3.1. A navigation model N is correct iff all navigation paths are correct.

This correctness definition relies on the computation of all navigation paths in N (section 3.3.1) and on the formalization of the correctness property for a given navigation path (section 3.3.2).

3.3.1 Determining the Possible Navigation Paths in a Navigation Model

Definition 3.3.2. Let S_{NavP} be the set of possible navigation paths in N . Then, $S_{NavP} = AllPaths(G_N)$, where $AllPaths(G_N)$ returns the set of all paths in G_N that do not include repeated arcs (these kinds of paths are also known as *trails* [3]).

S_{NavP} considers as valid navigation paths all possible paths. However, in general, not all possible navigation paths in N are valid, since a user cannot start browsing the web application choosing an arbitrary page but beginning in some predefined entry page. Similarly with the exit pages, users are expected to follow the navigation path until they arrive to some predefined exit page; after that they can quit or start from the beginning again. For this purpose, some web modelling languages support the concept of home page, the notion of transaction [4] or the concept of service [11].

When this information is available in the navigation model, we may discard from S_{NavP} those paths that either do not start in an entry page or do not finish in an exit page. As an example, given the navigation model of Fig. 3 we could define that *HomePage* is the entry page and that the exit pages are *HomePage* and *CheckOut* page. Then, the navigation sequence $\{HomePage, NewSale, NewSaleLine, CheckOut\}$ would represent a valid path while $\{HomePage, NewSale\}$ would not.

Clearly, the user may quit the web application before reaching an *exit* page. However, in that case the user is not properly interacting with the web application and thus the correctness of this partial interaction does not affect the correctness of the navigation model.

3.3.2 Correctness of a Navigation Path

Correctness of a navigation path depends on the operations associated to the arcs contained in the path. The basic idea is that some operations require the presence of other operations in a precedent or subsequent arc in the path in order to leave the data in a consistent state. For instance, a path including an *InsertSale* operation on an arc a_i requires that at least an operation *InsertComposedOf* appears further in the path (that is, it must exist an arc a_j , $j \geq i$, where $InsertComposedOf \in label(a_j)$). Otherwise, every run on this navigation path will end up in an inconsistent state due to the insertion of a sale not related with any sale line, thus violating the minimum multiplicity constraint of *Sale* in *ComposedOf*.

When an operation op_1 requires the presence of another operation op_2 in the same path we say that op_1 depends on op_2 . Dependencies for an operation depend on the

type of the operation (insert, update,...) and on the integrity constraints defined in the data model. We just consider graphical constraints (as the cardinality, disjoint and complete constraints) since most web modelling languages do not permit the definition of textual integrity constraints.

A navigation path must satisfy all dependencies of all operations included in the path to have a chance of finishing successfully. We denote by $SeqOp_{nav}$ the ordered sequence of all operations associated to the arcs contained in the path. Given a navigation path nav consisting of the sequence of arcs $a_1...a_n$, the first operation in $SeqOp_{nav}$ is the first operation in $label(a_1)$ and the last operation is the last operation in $label(a_n)$.

Note that the satisfaction of all dependencies is a necessary condition but not a sufficient one to ensure that all runs following the navigation path end successfully (this will depend on the exact parameter values provided by the user at run-time); this just guarantees that a successful run exists at least (i.e. a navigation path including the creation of a sale and the creation of a link between the sale and a sale line may fail if the parameters for the operations are not the appropriate ones; a navigation path not including the link creation after the sale creation will always fail).

Definition 3.3.3. A navigation path nav is correct when, for each operation op_i in $SeqOp_{nav}$, the set of dependencies dep_{op_i} for op_i is satisfied in $SeqOp_{nav}$

It may happen that an operation op_i requires N ($N > 1$) operations of type op_j . This dependency is satisfied in $SeqOp_{nav}$ when the N op_j operations explicitly appear in it. Alternatively, it is also satisfied if the navigation path nav consists of the arcs $a_1...a_n$, op_i is associated to an arc a_i , op_j to an arc a_j and there is a cycle in the graph including a_j but not a_i . Iterating through the cycle N times, the user could generate the required N op_j operations when running the application.

In the following we define how to compute the exact set of dependencies dep_{op} for an operation op . A dependency for an operation op is defined as a tuple $\langle direction, operation, number \rangle$ where *operation* is the name of the operation required by op and *direction* indicates if *operation* must be executed before op (symbol \leftarrow), after⁵ op (symbol \rightarrow) or if the exact position of op is irrelevant (symbol \uparrow). *Number* informs about how many operations of type *operation* are required by op . More complex dependencies are expressed as a sequence of simple ones joined with the logical AND and OR operators (as an example, op may require the existence of the operations op_1 and op_2 or, alternatively, the existence of the operation op_3).

Definition 3.3.4. Let ET be an entity type and op be an operation defined over ET . Dep_{op} is computed as follows:

- If $op = InsertET$, there is a dependency $dep_{RT} = \langle \leftarrow, InsertRT, \min(ET, RT) \rangle$ for each RT where $\min(ET, RT) \geq 1$. Additionally, if ET is the supertype of a complete generalization $Gens(ET, ET_1, \dots, ET_n)$ there is a dependency $dep_{GensSup} = \langle \leftarrow, InsertET_i, 1 \rangle$ for at least one ET_i . If ET is a subtype of a disjoint and complete generalization $Gens(ET', ET, \dots)$ we need a dependency $dep_{GensSub} = \langle \leftarrow, InsertET', 1 \rangle$. Dep_{op} is the union of the dep_{RT} , $dep_{GensSup}$ and $dep_{GensSub}$ dependencies.
- If $op = DeleteET$, there is a dependency $dep_{RT} = \langle \leftarrow, DeleteRT, \min(ET, RT) \rangle$ for each RT where $\min(ET, RT) \geq 1$. Additionally, if ET is a subtype of a disjoint

⁵ But not necessarily *immediately* before or after.

and complete generalization $Gens(ET', ET, \dots)$ there is a dependency $dep_{GensSub} = \langle \leftrightarrow, DeleteET', 1 \rangle$. If ET is the supertype of a complete generalization $Gens(ET, ET_1, \dots, ET_n)$ there is a dependency $dep_{GensSup} = \langle \leftrightarrow, DeleteET_i, 1 \rangle$ for at least an ET_i . Dep_{op} is the union of the dep_{RT} , $dep_{GensSup}$ and $dep_{GensSub}$ dependencies.

Note that no dependencies are needed for the $UpdateA_iET$ operation since changes on attribute values do not either violate cardinality, complete or disjoint constraints.

Definition 3.3.5. Let RT be a relationship type with two participants ET_1 and ET_2 . Let op be an operation defined over RT . Dep_{op} is computed as follows⁶:

- If $op = InsertRT$, there is a dependency $dep_{op} = \langle \hat{\uparrow}, DeleteRT, 1 \rangle$ (if $min(ET_i, RT) = max(ET_i, RT)$ for just one participant ET_i) OR $\langle \leftarrow, InsertET_i, 1 \rangle$ for each ET_i such that $min(ET_i, RT) = max(ET_i, RT) \geq 1$.
- If $op = DeleteRT$, there is a dependency $dep_{op} = \langle \hat{\uparrow}, InsertRT, 1 \rangle$ (if $min(ET_i, RT) = max(ET_i, RT)$ for just one participant ET_i) OR $\langle \leftarrow, DeleteET_i, 1 \rangle$ for each ET_i such that $min(ET_i, RT) = max(ET_i, RT) \geq 1$.

Consider the navigation path nav for the graph of Fig. 4 consisting of the navigation sequence: $\{HomePage, NewSale, NewSaleLine, CheckOut\}$. For this path, $SeqOp_{nav} = \{InsertSale, InsertSaleLine, InsertComposedOf, InsertReferences\}$. To check the correctness of nav we must consider the dependencies for all operations in $SeqOp_{nav}$. According to the previous rules, the dependencies are:

$$\begin{aligned} dep_{InsertSale} &= \langle \leftrightarrow, InsertComposedOf, 1 \rangle \\ dep_{InsertSaleLine} &= \langle \leftrightarrow, InsertComposedOf, 1 \rangle \text{ AND } \langle \leftrightarrow, InsertReferences, 1 \rangle \\ dep_{InsertComposedOf} &= \langle \leftarrow, InsertSaleLine, 1 \rangle \text{ OR } \langle \hat{\uparrow}, DeleteComposedOf, 1 \rangle \\ dep_{InsertReferences} &= \langle \leftarrow, InsertSaleLine, 1 \rangle \text{ OR } \langle \hat{\uparrow}, DeleteReferences, 1 \rangle \end{aligned}$$

$SeqOp_{nav}$ satisfies the dependency for $InsertSale$ since there is an $InsertComposedOf$ operation after the $InsertSale$ operation in the sequence. Dependencies for $InsertSaleLine$ are satisfied as well because $SeqOp_{nav}$ includes the $InsertComposedOf$ and $InsertReferences$ operations after $InsertSaleLine$. Dependencies for $InsertComposedOf$ require that before this operation we find in $SeqOp_{nav}$ an $InsertSaleLine$ operation or (anywhere) a $DeleteComposedOf$. Since $SeqOp_{nav}$ contains the $InsertSaleLine$ operation before the $InsertComposedOf$ operation, this OR-dependency is also satisfied. This is likewise with the dependencies for $InsertReferences$. Therefore, all dependencies for the operations in the path are satisfied and we may conclude that this navigation path is correct.

Given all these previous definitions, the verification that a navigation model N is correct can be summarized in the following steps: 1 – Generation of the graph G_N , 2 – computation of the function $AllPaths$, 3 – Computation of $SeqOp_{nav}$ for each navigation path, 4 – Determining the dependencies for all operations in $SeqOp_{nav}$ with the rules presented above and 5 – Checking that all paths satisfy the correctness definition 3.3.3.

⁶ For some (unusual) cardinality combinations additional (longer) sets of dependencies could be defined.

4 Generating a Complete and Correct Navigational Model

Given a data model D , we show in this section how to automatically derive a preliminary navigation model that is guaranteed to be complete and correct with respect to D (idea of *correctness-by-construction* [9]). Designers may complement this initial navigation model (for instance, adding all details on the internal page structure) to obtain the full navigation model for the web application.

Clearly, such automatic generation offers three main benefits. Firstly, designers save time and effort by not defining the navigation model from scratch. Secondly, it minimizes costly design errors since designers depart from an already complete and correct model. Finally, the generated model may present several alternative navigation designs that go beyond the one/s the designer had in mind and that may be worth exploiting in the final navigation model.

The construction of this initial model is split in several phases:

1. Generation of basic pages and links to ensure the completeness of the navigation model
2. Combining basic pages to create correct navigation paths
3. Refactorings to improve the structure of the generated navigation model

As a result of the process, we obtain a graph G_N representing the new navigation model. Then, this graph can be translated into the actual navigation model by means of reversing the rules introduced in section 3.1.

Each of the steps is described as follows.

4.1 Generation of a Complete Navigation Model

To be complete, the navigation model must contain all necessary data manipulation operations to modify the underlying web application data. Thus, for each required operation op , the graph G_N must include at least an arc a verifying that $op \in label(a)$.

Additionally, for each created arc a we define a new vertex v_l in the graph representing the page where the user can select/enter all the parameters required by op . The link represented by a is anchored in that page, that is, v_l is the origin vertex for a . At this stage of the process, we could use as a target page any other page of the model. The easiest option is to assume that either the target page is the same page (so that the user can apply again the same operation) or the home page.

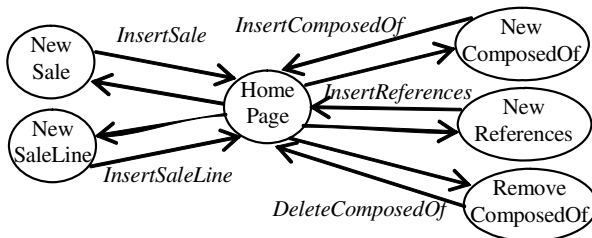


Fig. 6. Graph corresponding to a complete navigation model

Assuming that the only operations that can be executed over the data model of Fig. 2 are: *InsertSale*, *InsertSaleLine*, *InsertComposedOf*, *InsertReferences* and *DeleteComposedOf*, a complete navigation model could be the one corresponding to the graph shown in Fig. 6 From the home page, the user can access five different pages. Each page permits to enter the necessary information to execute the operation attached to the single exit link in the page. The exit link leads to the home page again.

4.2 Generation of a Complete and Correct Navigation Model

The previous graph corresponds to a complete navigation model but not a correct one since, for instance, the user can insert a new sale without inserting the corresponding sale lines thus leaving the data in an inconsistent state. In this section we extend this initial graph to ensure its correctness. Due to space limitations, we focus on the correctness of the subgraph including the home page and the new sale page.

As seen in section 3.3, correctness depends on the dependencies among the operations contained in the navigation paths of the graph. For instance, the occurrence of an *InsertSale* operation in a navigation path *nav* requires that, to be correct, *nav* includes an *InsertComposedOf* operation as well.

Therefore, a first step to ensure the correctness of the graph of Fig. 6 is to extend the graph by adding a new vertex and a new arc with the *InsertComposedOf* operation after the *NewSale* vertex (see Fig. 7). Now after inserting a new sale the user is redirected to a page to create a *ComposedOf* link.

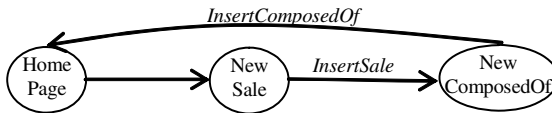


Fig. 7. Graph corresponding to a partially correct navigation model

However, this navigation path is not yet correct. The addition of the *InsertComposedOf* operation forces us to consider also the dependencies of this new operation. *InsertComposedOf* requires a previous operation *InsertSaleLine* or, alternatively, the existence of a *DeleteComposedOf* operation. None of them already appear in the path so we need to add one of them as well. Since we have two different alternatives to satisfy the correctness (either to add a new vertex and link for the *InsertSaleLine* or for the *DeleteComposedOf* operation), we duplicate the path created up to now. Each path takes one of the possible options so that we can cover all possible scenarios. After, we continue the process with each path separately.

In the path with the *InsertSaleLine* operation we need to further add a new *InsertReferences* operation. Dependencies for this latter operation are already satisfied by the path so we can stop the process for this path. The path with the *DeleteComposedOf* operation does not need new extensions (dependencies for *DeleteComposedOf* are satisfied if earlier in the path we find a *DeleteSale* or a *InsertComposedBy* as it is the case).

Fig. 8 shows the final aspect for this part of the graph. From the home page, we can insert a new sale in two different ways. We can either insert the sale and the related sale line or, alternatively, we can insert the sale and assign an existing sale line (previously related to a different sale) to the sale. The designer may decide if this second alternative makes sense in this particular domain. The decision cannot be automated.

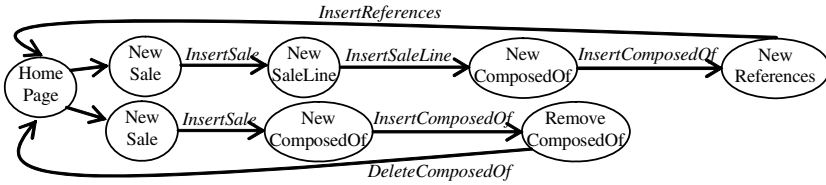


Fig. 8. Graph corresponding to a correct and complete navigation model

The generation process may add some extra arcs to create cycles in the graph. Cycles are created when the multiplicities of relationship types modified in the path are greater than one. In the previous example the graph should offer the option of adding several sale lines for the same sale as permitted by the multiplicities of *ComposedOf*.

More formally, the generation of a correct navigation model can be summarized in the following steps:

1. Compute the function *AllPaths* (see section 3.3.1) for the complete graph obtained in the previous section. *AllPaths* in this case just needs to consider the arcs between the home page and each specific page created for the required operations.
2. For each path compute the *transitive closure* of the dependencies of the operations in the path. The *transitive closure* can be computed by means of recursively applying the dependency rules of section 3.3.2 over the new operations added to the path until no more operations are added. When finding alternative dependencies (none of them already satisfied in the path) we create an additional path for each alternative.
3. Extending the graph with the new vertices and arcs that are necessary to satisfy all dependencies appearing in the transitive closure.

4.3 Refactorings for Navigation Models

Refactorings were initially proposed at the code level [8] as a disciplined technique for improving the structure of existing code (using simple transformations) without changing the external observable behavior. More recently, some work has been carried out to apply this technique on software models instead of on source code [12].

In this section we propose three simple refactorings to automatically improve the structure of the navigation model obtained at the end of the previous step. In our case, “without changing the external observable behavior” means without turning the model into an incomplete or incorrect model, that is, refactorings are transformations of the graph G_N representing a navigation model N that keep the completeness and correctness properties of N .

Refactoring 1. Removal of redundant navigation paths. Given two navigation paths n_1 and n_2 , we say that paths n_1 and n_2 are equivalent when $SeqOp_{n_1} = SeqOp_{n_2}$, where $SeqOp_{n_i}$ represents the ordered sequence of operations associated to the arcs of n_i . If two navigation paths are equivalent they are also redundant since the removal of one of them does not affect the completeness or the correctness of the navigation model. This refactoring removes all redundant navigation paths. The removal of a path consists in the removal of all arcs and nodes in the path not appearing in any other (non-redundant) path.

Refactoring 2. Head-Merging of navigation paths. Given two navigation paths $n_1 = \{ \langle v_1, a_1, v_2 \rangle, \langle v_2, a_2, v_3 \rangle, \dots, \langle v_{x-1}, a_{x-1}, v_x \rangle \}$ and $n_2 = \{ \langle v'_1, a'_1, v'_2 \rangle, \langle v'_2, a'_2, v'_3 \rangle, \dots, \langle v'_{y-1}, a'_{y-1}, v'_y \rangle \}$ we can merge the beginning of the two paths when there is an interval $1..i$ where for all $k, 1 \leq k \leq i, label(a_k) = label(a'_k)$ (that is when the first part of the path coincides in n_1 and n_2). After applying the refactoring, n_2 becomes $n_2 = \{ \langle v_1, a_1, v_2 \rangle, \langle v_2, a_2, v_3 \rangle, \dots, \langle v_{i-1}, a_{i-1}, v_i \rangle, \langle v_i, a'_i, v'_{i+1} \rangle, \dots, \langle v'_{y-1}, a'_{y-1}, v'_y \rangle \}$

Refactoring 3. Tail-Merging of navigation paths. Given two navigation paths $n_1 = \{ \langle v_1, a_1, v_2 \rangle, \langle v_2, a_2, v_3 \rangle, \dots, \langle v_{x-1}, a_{x-1}, v_x \rangle \}$ and $n_2 = \{ \langle v'_1, a'_1, v'_2 \rangle, \langle v'_2, a'_2, v'_3 \rangle, \dots, \langle v'_{y-1}, a'_{y-1}, v'_y \rangle \}$ we can merge the end of the two paths when there is an interval $1..i$ where for all $k, 1 \leq k \leq i, label(a_{x-k}) = label(a'_{y-k})$ (that is when the last part of the path coincides in n_1 and n_2). After applying the refactoring, n_2 becomes $n_2 = \{ \langle v'_1, a'_1, v'_2 \rangle, \langle v'_2, a'_2, v'_3 \rangle, \dots, \langle v'_{y-i}, a'_{y-i}, v_{x-i+1} \rangle, \dots, \langle v_{x-1}, a_{x-1}, v_x \rangle \}$

The application of the second refactoring over the graph of Fig. 8 results in the new graph shown in Fig. 9.

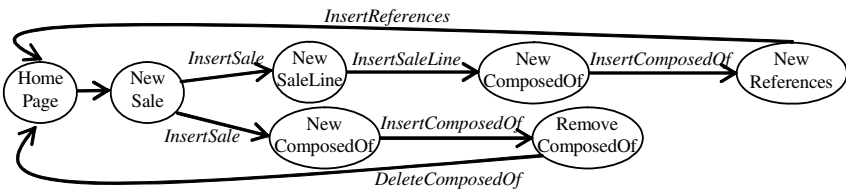


Fig. 9. Graph after applying the second refactoring

Apart from these automatic refactorings we could also provide several manual refactorings (i.e. refactorings that the designer must decide where and when to apply them). For instance, we could merge the nodes *NewSaleLine*, *NewComposedOf* and *NewReferences* in a single page with an outgoing link including the (ordered) sequence of operations included in the outgoing links for the three pages.

5 Related Work

Two kinds of related research are relevant to this paper: methods proposing properties to determine the quality of navigation models and methods devoted to the automatic generation of navigation models from data models.

Regarding quality aspects of navigation models, current proposals are rather limited. CASE tools provide limited verification facilities, mainly purely syntactic analysis of the correctness of the models regarding the syntax and semantics of the modelling languages. Other common supported properties include basic verifications of the navigation structure, as the reachability of all pages from the home page [4], [5]. [11] accepts the definition of a *route* for each conceptual user service. A route represents the sequence of steps that the user must follow to complete the service. Then, it checks that the structure of the navigation model is consistent with the defined routes. A few powerful formal verifiers also exist (see [6] as an example) though they are not yet fully integrated with current web development methods, which hamper their practical usability. Moreover, none of these proposals consider the specificities of navigation models with content-modification operations or the relationship between navigation models and their related data models as the quality properties we have defined in this paper.

With respect to the automatic generation of navigation models, existing approaches (as [1], [7] and [2]) derive the structure of the navigation model based on the relationships among the entity types in the data model (outside the web community, we find similar proposals, devoted to the automatic generation of the application graphical-user interface, see [14] as an example). Nevertheless, the generated models are just read-only navigation models for browsing the data; they do not include data modification functionalities.

Other important kinds of quality properties, as usability and accessibility of web applications [16] are outside the scope of this paper.

6 Conclusions and Further Work

We have presented two new quality properties (completeness and correctness of navigation models) that focus on the relationship between navigation and data models defined for the same web application. With these properties we can check whether a navigation model conforms to its data model so that inconsistencies between them can be detected in the early stages of the web application development process.

Our properties complement current quality checks for navigation models, which do not consider the data modification operations that may appear in those models. We believe that our properties are relevant to current web development methods addressing the definition and generation of fully-fledged web applications.

Also, we have shown how these properties can be used to automatically generate a preliminary version of a navigation model once the data model has been specified. This initial navigation model can then be refined by the designer in order to obtain the final navigation model for the web application.

Regarding further work, we would like to extend our quality assessment by detecting not only errors in the navigation model but also other kinds of problematic situations (“warnings”) that should be revised and by exploring the applicability of our graph-based representation to check additional properties (some properties may be reduced to path problems over our graph [15]) Besides, we also plan to improve our

current generation method for navigation models to include additional patterns and refactorings that make the model closer to the final one expected by the designer. Finally, we plan to validate our approach using an industrial case study.

Acknowledgments

We would like to thank the people of the GMC group and the anonymous reviewers for their many useful comments in the preparation of this paper. This work has been partially supported by the Ministerio de Ciencia y Tecnología under the project TIN2005-06053 and the integrated action HI2006-0208.

References

1. Albert, M., Pelechano, V., Fons, J., Rojas, G., Pastor, O.: Extracting Knowledge from Association Relationships to Build Navigational Models. LA-WEB'03, pp. 2–10 (2003)
2. Assossou, D., Wack, M.: Transformation Rules from Conceptual Model to Navigational Model in Hypermedia Applications. WEBIST'06 (1) pp. 428–434 (2006)
3. Bollobás, B.: Modern Graph Theory, p. 394. Springer-Verlag, Heidelberg (1998)
4. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks* 33(1-6), 137–157 (2000)
5. Comai, S., Matera, M., Maurino, A.: A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 339–350. Springer, Heidelberg (2002)
6. Deutsch, A., Marcus, M., Sui, L., Vianu, V., Zhou, D.: A Verifier for Interactive, Data-driven Web Applications, SIGMOD'05, pp. 539–550 (2005)
7. Falquet, G., Guyot, J., Nerima, L., Park, S.: Design and analysis of active hypertext views on databases, *Information Modeling for Internet Applications*, pp. 40–58. Idea Group Publishing (2003)
8. Fowler, M.: *Refactoring: Improving the design of existing code*, p. 464. Addison-Wesley, London, UK (1998)
9. Hall, A., Chapman, R.: Correctness by construction. *IEEE Software* 19(1), 18–25 (2002)
10. Jakob, M., Schwarz, H., Kaiser, F., Mitschang, B.: Modeling and Generating Application Logic for Data-Intensive Web Applications, ICWE'06, pp. 77–84 (2006)
11. Lucas, F.J., Molina, F., Toval, A., de Castro, M.V., Cáceres, P., Marcos, E.: Precise WIS Development. ICWE'06, pp. 71–76 (2006)
12. Mens, T., Tourwé, T.: A Survey of Software Refactoring. *IEEE Trans. Software Eng.* 30(2), 126–139 (2004)
13. Pastor, O., Fons, J., Pelechano, V., Abrahao, S.: Conceptual Modelling of Web Applications: The OOWS approach. In: *Web Engineering*, pp. 277–302. Springer-Verlag, Heidelberg (2006)
14. Pizano, A., Shirota, Y., Iizawa, A.: Automatic Generation of Graphical User Interfaces for Interactive Database Applications. CIKM'93, pp. 344–355 (1993)
15. Tarjan, R.E.: Fast algorithms for solving path problems. *Journal of the ACM* 28(3), 594–614 (1981)
16. Vanderdonckt, J., Beirekdar, A.: Automated Web Evaluation by Guideline Review, *Journal of Web Engineering* 4(2), 102–117 (2005)

Metamodeling the Quality of the Web Development Process' Intermediate Artifacts

Cristina Cachero¹, Coral Calero², and Geert Poels³

¹ University of Alicante. Spain
ccachero@dlsi.ua.es

² ALARCOS Research Group. University of Castilla-La Mancha. Spain
Coral.Calero@uclm.es

³ Faculty of Economics and Business Administration, Ghent University. Belgium
geert.poels@ugent.be

Abstract. WE practices lack an impact on industry, partly due to a WE field that is not quality-aware. In fact, it is difficult to find WE methodologies that pay explicit attention to quality aspects. However, the use of a systematic process that includes quality concerns from the earliest stages of development can contribute to easing the building up of quality-guaranteed Web applications without drastically increasing development costs and time-to-market. In this kind of process, quality issues should be taken into account while developing each outgoing artifact, from the requirements model to the final application. . Also, quality models should be defined to evaluate the quality of intermediate WE artifacts and how it contributes to improving the quality of the deployed application. In order to tackle its construction while avoiding some of the most common problems that existing quality models suffer from, in this paper we propose a number of WE quality models to address the idiosyncrasies of the different stakeholders and WE software artifacts involved. Additionally, we propose that these WE quality models are supported by an ontology-based WE measurement meta-model that provides a set of concepts with clear semantics and relationships. This WE Quality Metamodel is one of the main contributions of this paper. Furthermore, we provide an example that illustrates how such a metamodel may drive the definition of a particular WE quality model.

Keywords: web quality process, web quality, web process quality, web measurement, ontology, metamodel, navigational model.

1 Introduction

It is an avowed fact that WE practices lack a big impact on industry [1]. This situation is at least partly caused by a WE field that is not quality-aware. In fact, it is difficult to find WE methodologies that include explicit support for quality aspects among their characteristics. In order to change this situation and assess the quality of the different WE artifacts, we need to define specific evaluation instruments. Such instruments may come in the shape of a *Quality Model*, defined by the ISO as the *set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality* [2].

At this point, a question may arise: what should be the objective of such quality instruments? Garvin [3] defines five different quality perspectives, among which two have been widely adopted when defining quality models. The first one is 'quality as the degree of compliance with respect to certain specifications'. This is the most widespread perspective in Software Engineering due to the fact that in this kind of quality assurance the end user is not involved and therefore measures are easier and cheaper to take. The second perspective is quality as 'meeting customer needs'. Even if much more complex to evaluate, it is this second perspective the one that, according to the ISO/IEC 9126 [2] and ISO/IEC 14598 [4] standards, should make up the overall objective of any quality evaluation process. Provided that we narrow the term 'customer' to that of 'end-user', this concept of quality from the end-users' perspective is what the ISO/IEC 9126 standard defines as 'quality in use', that is, the *efficiency, productivity, security and satisfaction with which users use the application to satisfy specific goals under specific conditions.*

This ISO recommendation agrees with the fact that quality in use, even if it has not been tackled in a systematic way, is a widespread concern among Web developers, due to the necessity for most applications to keep the audience coming back to the site [5]. However, this concern is not reflected in current Web quality evaluation practices. If we examine the myriad of Web design guidelines [6] and automated measures [7] that can be gathered in literature we observe that, as it happens in Software Engineering, such Web evaluation effort reflects a 'conformance to specification perspective', that is, implicitly assumes the assessment of quality before the user is actually interacting with the application. One second problem of using such measures and guidelines is that, talking in terms of the OMG Standard Metapyramid [8] (see main subdivisions in Figure 1), the Web quality evaluation effort is concentrated on the M1-Implementation level (measures over the application code, without running it) and M0-test level of abstraction (code running under testing conditions). Only log analysis techniques have strived to evaluate the end-user actual behavior.

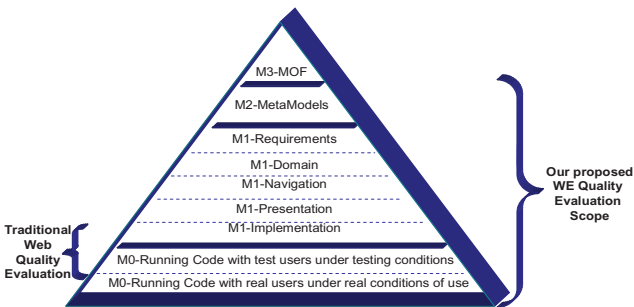


Fig. 1. The OMG standard metapyramid with additional WE subdivisions to distinguish among different levels of abstraction at M1 level (adapted from ISO10027)

Starting to assess quality at such a late stage of development is avowed to have a negative impact on the final product cost and quality [9]. In fact, according to Moody

and Shanks [10], the cost associated with removing a defect during design is on average 3.5 times greater than during requirements; at implementation stage the effort associated with removing the same defect can be up to 50 times greater, and up to 170 times greater after delivery. Other empirical studies have shown that moving quality evaluation effort up to the early phases of development can be 33 times more cost effective than testing done at the end of the development.

The solution to these two problems requires therefore to conciliate an early evaluation of the main internal Web products (that is, a ‘conformance to specification’ evaluation that is not just centered on code, but also makes use of early models, from requirements to implementation, see Figure 1) with the necessity of ‘meeting customer needs’, which implies taking into account the actual user behaviour under real conditions of use. Fortunately, the ISO set of quality standards establishes that the ‘conformance to specifications’ degree of a given software product (such as intermediate artifacts generated as part of a WE process) may be a valid predictor of the ability of the product to meeting user needs. This assumption means that is it possible to improve the Web quality in use by working on the quality of each outgoing artifact that participates in a typical WE process, from the requirements model to the final application to be delivered. The, only requisite is that the translation from the ‘meeting user needs’ (whose fulfillment is the final objective of the development process) quality perspective to the set of specific ‘conformance to specifications’ requirements defined for each model (which analysts/designers can systematically check) is accurately defined.

In order to perform such inclusion of quality concerns in existing WE methodologies in a sensible a consistent way, we have developed a proposal that has three main elements: (1) a quality-aware Web development process, (2) a set of general-purpose WE quality models specific for each stakeholder and/or WE artifact and, (3) a WE-Software Measurement Metamodel (SMM) that permits to operationalize and, if needed, also tailor, those quality models according to a particular domain and/or application. All three elements are based on principles and achievements that, uncovered in different quality lines of research, provide insights into how to deal with quality in each of the different workflows that a typical WE process defines, from requirements to implementation.

In this paper we are centering on the last two elements of our approach, that is, the definition of WE quality models and their adaptation to particular Web applications. To justify the necessity for our proposal, in Section 2 we present a brief overview of the main problems that existing quality models suffer from. How (1) the use of the Software Measurement Ontology (SMO) and (2) the definition of an associated WE measurement metamodel that guides the construction and adaptation of the quality model, can contribute to palliate such problems is presented in Section 3. This WE Measurement Metamodel must be instantiated to reflect a certain WE quality model as well as any necessary tailoring. An example of such instantiation that reflects an hypothetical Navigational quality model is developed in Section 4. Last, conclusions and future work are explained in Section 5.

2 Related Work

Quality models for software products are far from scarce. Well known pioneer models include McCall [12], Boehm[13], Dromey [14] and ISO/IEC 9126 [2]. All of them center on measurable elements over the implementation of the software product on one hand, and on the (abstract) quality characteristics on the other hand, and try to establish relationships among both dimensions.

There are various proposals of specific Web quality models, most of them tackling the Web idiosyncrasy from the 'meet the user needs' perspective [15, 16, 17, 18, 19]. From them, only [20] and [19] promote considering other artifacts (apart from code) that may take part in the WE development cycle, and none of them provide independent quality models for each level of abstraction. These approaches can however be refined and complemented by research in conceptual modeling quality (e.g. Lindland et al. framework [12], Krogstie et al. framework [21] and Moody and Shanks framework [11]), which provides further insight into how the quality concept can be dealt with at higher levels of abstraction. Last but not least, Web quality evaluation needs to be performed following a well defined quality evaluation process. Some well known Web quality evaluation processes are WebQUEM [15] and WebTango [7]. The main drawback of these processes is that they assume that Web quality evaluation is performed on the deployed application. Only [11] and [19] present a broader perspective and try to conciliate Web quality evaluation with a general WE development process, even if they only provide general guidelines and not specific proposals. Next, we present the challenges all these fields pose, and how we propose to integrate them in a single, consolidated proposal in the context of WE.

2.1 Research Issues

When trying to operationalize all the myriad of different quality models and quality evaluation processes that have been proposed in literature, several theoretical and practical issues arise: [10, 22]:

- **P1: Terminology inconsistencies.** Most approaches (the exception being those based on theoretical grounds) lack a definition for quality concepts that is precise and concise. For instance, while in the ISO/IEC 9241-11 usability refers to the end-user perception as a whole (and therefore encompasses efficiency effectiveness and satisfaction), in the ISO/IEC 9126 end-user perception is referred to as 'quality in use', and usability is only one of the internal characteristics that may affect such quality in use.
- **P2: Partially defined.** Most quality models are outlined but not fully developed. All define measurable concepts, some of them also attributes, few of them include (most often partial) measures and scarcely any defines decision criteria or indicators.
- **P3: Lack of focus.** Most quality models provide an extensive (and mostly tangled) coverage of stakeholders and levels of abstraction. An example of such assertion is the QUIM model [22], which aims at being a consolidated usability model that integrates all possible perspectives. As another example, WQM [16] covers 10 factors, 26 subfactors and 127 measures that may be related to any WE artifact, from analysis to implementation.

- **P4: Disregard for process quality.** Most quality models define criteria and, in some cases, measures for evaluating products (error detection), but not how to develop products in a way that assures a certain level of quality (error prevention).
- **P5: Lack of integration with current practices.** Quality management is not integrated into current WE practices.
- **P6: Lack of simplification and validation.** Quality models that include measures usually pay little attention to the theoretical/empirical validation of the included measures. Furthermore, although empirical research has shown that a few measures most times suffice to obtain significant gains in quality ([11]), quality models usually include an extensive, even redundant set of measures. Such verbosity unnecessarily increases the complexity and therefore hampers the potential usefulness of the quality models.
- **P7: Interdependencies and measure interpretations not clear.** In most quality models (again the notable exception being those that are based on theory), the degree of influence of individual internal quality factors on the quality in use of the application, as well as their interdependencies, are not well established. For example, the role of learnability versus understandability in the usability model presented in [19] is an open issue. Also, little information is provided on how to interpret measurement results.
- **P8: Lack of tool support.** Although most Web measures are automated, tool support for the definition of quality models and, even more important, for the automation of the measurement process on a given Web application is still an open issue.
- **P9: Lack of guidelines for improvements.** Even in the case of being able to evaluate a certain Web characteristic, to our knowledge extent no quality model provides a clue about how (by means of which changes in the artifacts) such evaluation could be improved, let alone to which extent such changes may affect the evaluation of other characteristic included in the quality model.

In order to overcome these problems, certain requirements should be achieved when defining WE quality models and integrating them with WE development processes:

Requirement 1. WE quality models should be expressed using a set of clear concepts with clear semantics and relationships, in order to ease their understanding and assure a structural coherence. This palliates problems P1, P2 and P3.

Requirement 2. WE quality models should be defined taking into account a specific stakeholder and a specific software artifact. This palliates problem P3.

Requirement 3. WE quality models should be accompanied by a WE quality evaluation process. Such process must be defined and integrated with the WE development process. This contributes to overcome problems P3, P4, and P5. Furthermore, for the definition of the WE quality evaluation process, standards should be followed when possible. This alleviates problem P6.

Requirement 4. Guidelines should be provided when possible to improve WE artifacts according to the WE Quality Model under consideration. Such guidelines should also if possible preserve the semi-automatic nature of the WE process. This contributes to solve problem P9.

Requirement 5. The integration of WE quality models in the WE process should preserve the semi-automated nature of such process. This contributes to alleviating problem P8.

Requirement 6. WE quality models should be empirically validated before being included in the WE process. This palliates problems P6 and P7.

In this paper, we will center on how the definition of WE Quality models that preserve *Requirement 1* and *Requirement 2* can be achieved if we base them on an ontology-based WE measurement meta-model. The use of a meta-model assures the syntactic correctness of the WE Quality Models (including completeness restrictions and focus control on specific stakeholders and specific WE artifacts), while the fact that this meta-model is based on an ontology contributes to avoiding terminology inconsistencies.

3 Definition of WE Quality Models Following an Ontology and a Meta-model

As we presented previously, one of the problems that existing Measurement quality models face is terminology inconsistencies. In order to overcome such problem we need a common vocabulary both to express WE concepts and to express quality concepts. Such common vocabulary usually comes in ontology form.

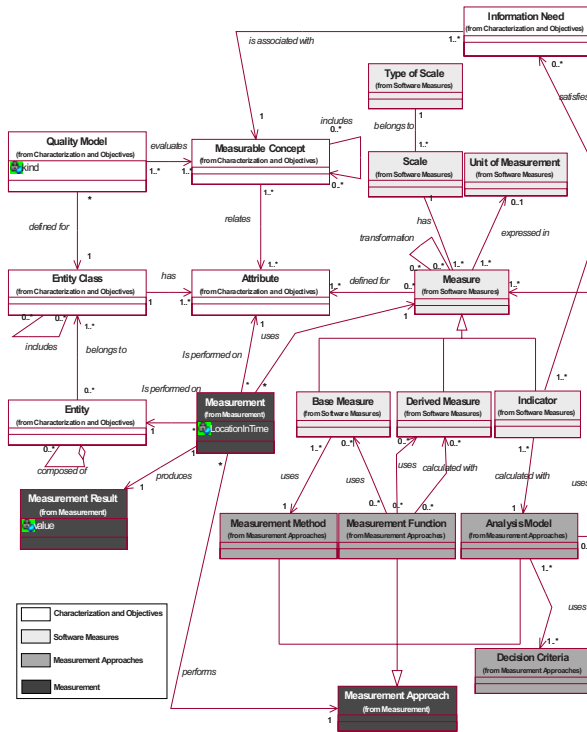


Fig. 2. The UML diagram of the SMO

Ontologies, defined as explicit, formal and shared specifications of a conceptualization, have been widely used in Software Engineering [23]. The use of an ontology not only avoids vocabulary conflicts and inconsistencies but also establishes the adequate level of detail for the definition of each concept.

While the definition of a WE ontology is still being worked on and remains out of the scope of this paper, the greater maturity of the measurement field causes some proposals for measurement ontologies to co-exist. From them, the Software Measurement Ontology (SMO)[24] is, to our knowledge extent, the most complete one, which is the reason why we have chosen it as the basis for our approach. The SMO ontology is structured around four packages, namely:

- **Software Measurement Characterization and Objectives**, which includes the concepts required to establish the scope and objectives of the software measurement process.
- **Software Measures**, which aims at establishing and clarifying the key elements in the definition of a software measure.
- **Measurement Approaches**, which introduces the concepts necessary for reflecting measurement results.
- **Measurement**, which establishes the terminology related to the act of measuring software.

Table 1. SMO terms definition

Term	Definition
Measurement Approach	Sequence of operations aimed at determining the value of a measurement result. (A measurement approach is either a measurement method, a measurement function or an analysis model)
Measurement	A set of operations having the object of determining the value of a measurement result, for a given attribute of an entity, using a measurement approach
Measurement Result	The number or category assigned to an attribute of an entity by making a measurement
Information Need	Insight necessary to manage objectives, goals, risks, and problems
Measurable Concept	Abstract relationship between attributes of entities and information needs
Entity	Object that is to be characterized by measuring its attributes
Entity Class	The collection of all entities that satisfy a given predicate
Attribute	A measurable physical or abstract property of an entity, that is shared by all the entities of an entity class
Quality Model	The set of measurable concepts and the relationships between them which provide the basis for specifying quality requirements and evaluating the quality of the entities of a given entity class
Measure	The defined measurement approach and the measurement scale. (A measurement approach is either a measurement method, a measurement function or an analysis model)
Scale	A set of values with defined properties
Type of Scale	The nature of the relationship between values on the scale
Unit of Measurement	Particular quantity, defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity
Base Measure	A measure of an attribute that does not depend upon any other measure, and whose measurement approach is a measurement method
Derived Measure	A measure that is derived from other base or derived measures, using a measurement function as measurement approach
Indicator	A measure that is derived from other measures using an analysis model as measurement approach
Measurement Method	Logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale. (A measurement method is the measurement approach that defines a base measure)
Measurement Function	An algorithm or calculation performed to combine two or more base or derived measures. (A measurement function is the measurement approach that defines a derived measure)
Analysis Model	Algorithm or calculation combining one or more measures with associated decision criteria. (An analysis model is the measurement approach that defines an indicator)
Decision Criteria	Thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result

In Figure 2 the UML diagram of the ontology is presented, while in Table 1 the concepts defined in the ontology are shown.

This ontology is the basis on which a namesake (and structure-equivalent) meta-model has been defined [25]. Next, we present how we have adapted such meta-model to meet our detected needs.

3.1 The WE Software Measurement Meta-Model (WE-SMM)

The Software Measurement Meta-Model (SMM) presented in [25] is a mirror of the Software Measurement Ontology presented in Figure 2, and may be instantiated to define in a systematic and non-ambiguous way a quality model that includes all the necessary concepts for its operationalization. The main advantage of using meta-models instead of ontologies in the context of a software development process stems in their prescriptive rather than descriptive nature, what permits the designer to make assumptions on the quality models that are not possible with ontologies. Furthermore, while ontologies need to be general, meta-models can be tailored to meeting specific needs. In our case, and given the fact that we aim at simplifying as much as possible the definition of WE Quality Models, we have adapted the SMM to the WE environment, with the aim of making its instantiation more intuitive for Web designers. Such WE-SMM is presented in Figure 3.

Summarizing, the construction of this WE-SMM has implied the following actions over the original SMM:

- We have limited the risk for inconsistencies in the measurement model by eliminating SMM redundant relationships.
- We have limited the set of valid Entity Classes to the outgoing artifacts of the WE development process. In this way, measurable concepts that are to be measured on different WE artifacts are forced to belong to different quality models.
- We have introduced a global Information Need that is connected with the WE-quality model as a whole to justify its definition. For the structure of this Global Information Need we propose to use the GQM template for goal definition [13].
- In order to keep the quality model simple, we have restricted the number of Information Need objects that can be associated with each Measurable Concept (1)
- For the same reason, we have established that each Information Need be satisfied by a single Indicator, implying that the Measurable Concept connected with the Information Need is also (transitively) associated with that indicator.
- In order to assure that every Attribute is measurable, every attribute defined in a WE quality model should be associated with at least one Measure that is devoted to measuring such Attribute. This restriction makes sure that the evaluation model is operationally defined by means of Measures, that is, no reliant on subjective interpretations of concepts [10].
- In order to further contextualize the WE quality model and help to keep the focus, we have added a 'Stakeholder' element to the original SMM. Stakeholders are usually not explicitly identified in existing quality models. However, as stated in [11], they are important in any quality model, as different Stakeholders will generally be interested in different Measurable Concepts. The instantiation possibilities of this

concept in the context of WE are (1) Analysts/Designers, (2) Developers/Maintainers and (3) Customers (subdivided into Acquirers and End-Users)

- Finally, we have omitted from the WE-SMM the Measurement package, due to the fact that their elements do not contribute to the definition of quality models but rather to the results of their operationalization.

Additionally, and although not directly reflected in the WE-SMM, in order to control the quality model complexity we recommend the limitation of the hierarchy depth of Measurable Concepts to two levels of detail. Also, following the ISO/IEC 9126 example, these two levels should be characterized by familiar labels and concise definitions. Similarly, attributes associated with Entity Classes should also be familiar and provide concise definitions. Finally, in order to facilitate a hypothetical merging of quality models at different levels of abstraction into a general, well-structured WE Global Quality Model, we recommend that attributes for the different models have unique names in the context of the WE field.

The concepts and relationships included in this meta-model, together with the additional recommendations, force a certain structure similarity among any quality model

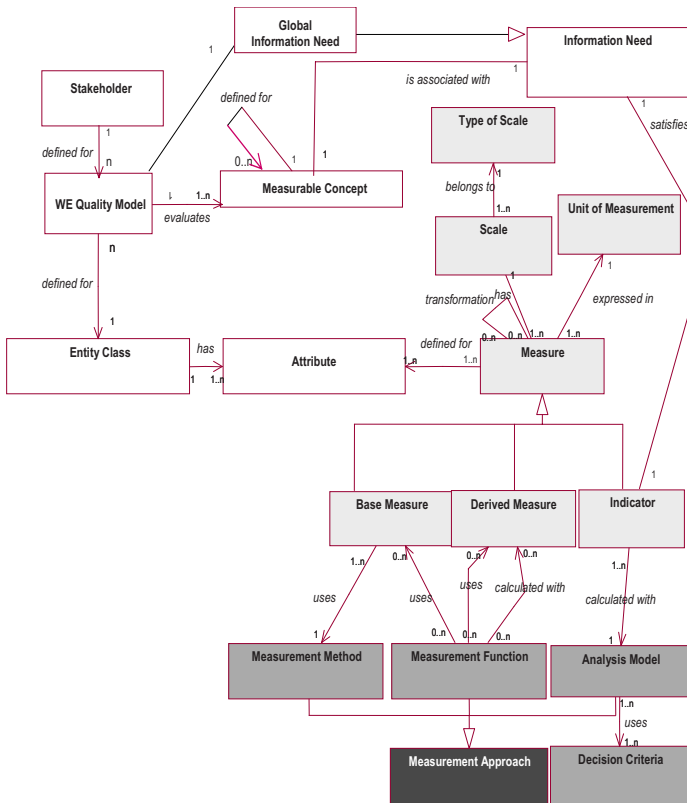


Fig. 3. WE-Measurement metamodel

defined based on it, what in turn facilitates the understanding and discussion of WE quality models among both researchers and practitioners.

With a WE-SMM that is based on a SMO we are fulfilling Requirements 1 and 2 (see Section 2). Next we present an example of how to use these instruments to define particular WE Quality Models.

4 WE-SMM Based Definition of WE-Quality Models

The WE-SMM defined above can be easily instantiated to define a complete and structurally sound quality model over any WE intermediate artifact with the ultimate objective of assuring certain characteristics that may contribute to improving the quality in use of the application. Such quality model can be general (when it does not take into account the particularities of the domain and/or application under evaluation) or particular, if such knowledge introduces any divergence in the elements that are taken into account for the evaluation task. This means that, while Quality Models in literature usually reflect a global view of the set of measurable concepts, measures and so on that may be applied at a certain level of abstraction (that required by the measures included), our WE-SMM instantiation may furthermore involve a tailoring process over the original Quality Model to adapt it to a given application in a given domain. During this tailoring process certain fine-tuning actions can be performed. For instance, more specific decision criteria that better reflect the domain knowledge could be defined, or certain attributes/measures may be dismissed to even further simplify the measurement process. Furthermore, we say that the WE-SMM instantiation operationalizes a Quality Model, due to the fact that (1) it obligues the QM to present certain characteristics (e.g. to provide measures for every attribute) and (2) it permits to express the Quality Model in a machine-readable format, which in turn opens the path to applying automation techniques.

In order to illustrate this fact, next we are presenting an instantiation example that operationalizes a hypothetical quality model devoted to evaluating the navigability of a Web application. Although the whole definition of this WE Quality Model is out of the scope of this paper, for illustration purposes let's assume that this Navigability WE-QM includes an Understandability characteristic that can be measured based on two attributes: Navigation Node Complexity and Navigation Path Complexity. The definition of the meaning of these concepts was presented in Table 1.

This measurement model intends to reflect the viewpoint of the end-user of the application, that is, the **Stakeholder** involved is the End-User. Therefore, only model qualities that are bound to contribute to increasing the end-user quality in use of the application should be included in this instantiation. Such instantiation aims at assessing the navigability problems that may arise due to a low-quality definition of navigational paths and navigational nodes.

Due to the fact that navigation paths and nodes are both defined by means of a navigation model, the **Entity Class** will be such Navigation Model, which is part of any WE methodology.

The WE-Quality Model is associated to a **Global Information Need To Know** how good Navigability is. Recall that the description of such global information need

must follow the GQM template, and therefore could be defined as follows: analyzing the WE Navigational Model for the purpose of evaluating it with respect to the navigability of the final application from the viewpoint of the end-user of the application in the context of a testing environment.

The **Navigability WE-Quality Model** contains a set of **Measurable Concepts**. If we consider Navigability as ‘Usability of the navigation’, we can assume that the main characteristics included in the ISO 9126-1 for Usability apply. These characteristics are Understandability, Learnability, Operability, Attractiveness and Compliance. We agree with [19] in that the first three Measurable Concepts (understandability, learnability, operability) are related with the user performance and can be therefore quantified using objective measures, some of which can be taken over navigational models. Attractiveness is not relevant at this stage of development, where final users are not yet present. Last, as far as we know there are no widely accepted standards or conventions regarding the definition of navigation structures in WE navigational models, and therefore Compliance is not relevant either.

Table 2. WE-Measurement Meta-model instantiation example

Term	Instantiation for the Understandability Measurable Concept
Stakeholder	End-User
Global Information Need	To know how good navigability is
Information Need	To know how good understandability is
Measurable Concept	Understandability
Entity Class	Navigation model
Attribute	(1) Navigation Node Complexity – (2) Navigation Path Complexity
WE-Quality Model	Navigability WE-Quality Model
Base Measure	(1) Number of attributes (NA) - (2) Number of Navigational Links (NNL)
Scale	Natural Number
Type of Scale	Ratio
Measurement Method	(1) Count the number of attributes of the model– (2) Count the number of links of the model
Indicator	UND_IND (NA, NNL)
Scale	Acceptable-NonAcceptable
Type of Scale	Ordinal
Analysis Model	f(UND_IND)=NA+NNL
Decision Criteria	If f(UND_IND) <50 then Acceptable else NonAcceptable

Each one of these Measurable Concepts must be related to an **Information Need**. The Information Need covered by Understandability in the context of the Navigability WE-Quality Model is To Know how good Understandability is. The description associated with such concept could be ‘the capability of the Web application navigational structure to enable the user to understand whether the application is suitable for her, and how it can be used for particular tasks under certain conditions of use’. Learnability and Operability can be defined similarly.

The next step consists in defining the **attributes** that may influence each characteristic. The navigational model has two main purposes in the WE development process. On one hand it defines the set of abstract pages, that is, the basic information nodes that make up the application. On the other hand, it provides the navigation paths and the navigation facilitators (menus, indexes, guided tours and so on) to improve the user experience. With these two purposes in mind, we have identified two Navigation Model Attributes:

1. Navigation Node Complexity: possibly related to learnability, and understandability. Such relationship is not in the meta-model because it is derived from the relationship between measures-attributes on one hand and the empirically validated relationship measure- measurable concepts on the other hand.
2. Navigation Path Complexity: possibly related to learnability, understandability and operability. Again, such relationship must be empirically validated (it would be a derived relationship)

The partial instantiation of the WE-SMM that gathers all the concepts presented so far is presented in Figure 4.

According to the meta-model, each model attribute must be related to at least one **measure**. For the sake of the example, let's suppose that we have determined that only two measures are relevant for the evaluation purposes of this Navigability WE-Quality Model: the number of navigational links (NNL) and the number of attributes (NA).

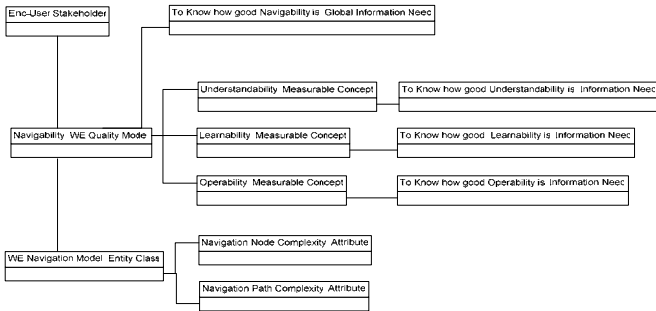


Fig. 4. Partial instantiation of WE_Navigability model (Part 1)

The definition of the Number of Navigational Links (NNL) **Base Measure** includes the **Scale** Natural Number, the **Type of Scale** Ratio and the **Unit of Measurement** Links. This measure is associated with the Navigation Path Complexity Attribute. The **measurement method** is 'to count the number of links of the model'. Similarly, the definition of the Number of Attributes (NA) Base Measure is associated with the Scale Natural Number, the Type of Scale Ratio and the Unit of Measurement Attributes. This measure is related with the Navigation Node Complexity Attribute. The **measurement method** is 'to count the number of attributes of the model'.

Also, each information need requires at least one **Indicator**. Indicators can be regarded as special kinds of measures that are related to decision criteria via an **Analysis Model**.

As an example, let's define the Understandability Indicator UND_IND. Let's suppose that the Analysis Model associated to this indicator is a function that involves the two measures presented above: $F(UND_IND)=NNL+NA$.

Let's also assume that this indicator belongs to the Scale {Acceptable, Non Acceptable} and Type of Scale Ordinal (Acceptable is better than Non Acceptable).

Last, for the definition of the decision criteria let's assume that the Trellis number applies, and that models with less than 50 elements are understandable enough. This

decision criteria is expressed in the meta-model instantiation as if $f(\text{UND_IND}) < 50$ then Acceptable else NonAcceptable.

Figure 5 presents a WE-SMM instantiation that reflects all these new elements of the Navigability WE-Quality Model.

As the reader may have already noticed, this measurement model is quite straightforward to use by any designer familiar with Navigation Models. The fact that we specifically consider the stakeholder helps to focus the measurable concepts, attributes and measures that must be taken in consideration.

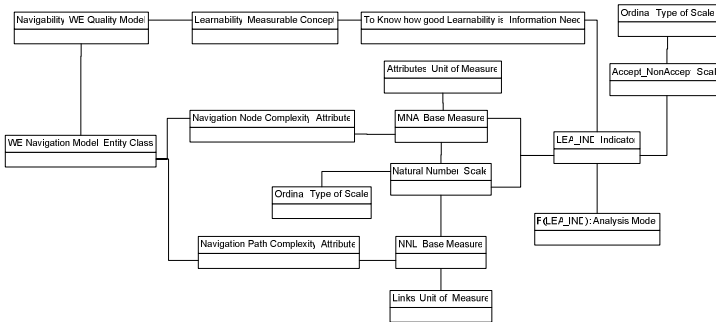


Fig. 5. Partial instantiation of WE_Navigability Model (Part 2)

5 Conclusions and Future Work

The systematic integration of quality issues in the WE field is mandatory if we aim at setting the focus on preventing rather than detecting errors and therefore decreasing maintainability costs. Even more important, we believe that providing practitioners with WE methodologies that assure a certain degree of quality of the application delivered is likely not only to support some of the WE traditional claims of providing better results than creative practices, but also to increase the acceptance rate of the WE technology in industry.

These quality issues should be taken into account while developing each outgoing artifact, from the requirements model to the final application to be delivered. In order to perform such inclusion of quality concerns in existing WE methodologies in a sensible and consistent way, we have developed a proposal that enriches the traditional WE development process with a set of quality activities and instruments that serve to evaluate the quality of intermediate WE artifacts as a means to improving the quality of the deployed application. Our proposal has three main elements: (1) a quality-aware Web development process (out of the scope of this paper), (2) a set of general-purpose WE quality models specific for each stakeholder and/or WE artifact and, (3) a WE software measurement metamodel (SMM) that permits to operationalize and, if needed, also tailor, those quality models according to a particular domain and/or application. All three elements are based on principles and achievements that, uncovered in different quality lines of research, provide insights into how to deal with quality in each of the different workflows that a typical WE process defines, from requirements to implementation.

This paper presents two contributions. On one hand, it provides an overview of the main problems associated with quality models in Software Engineering, and a set of requirements that should be met by any quality model proposal if it aims at being of use. On the other hand, we clarified how some of these requirements can be covered by defining quality models with certain elements and restrictions. In order to assure this, we have defined a WE-SMM that tailors an existing SMM. This WE-SMM and the way we propose to use it to instantiate sound and complete WE quality models is the second contribution of this paper.

If we review the list of problems presented in Section 2, the fact that such WE-SMM is based on an underlying ontology contributes to avoiding terminology inconsistencies (P1). Also, the use of this metamodel turns the descriptive nature of ontologies into prescriptive, and therefore assures that a set of syntactic and semantic constraints are met by any quality model defined as an instantiation of such meta-model. One of such constraints is the set of elements that must be present in any syntactically correct WE quality model, which partially solves P2. The focus on a given field (in our case the WE field) as the application context of the quality models facilitates the task of constructing consolidated, exhaustive yet specialized models. This fact also contributes to alleviating P2 and P3. Additionally, the consideration of the WE development process with its related stakeholders and outgoing artifacts allows us to univocally define (1) the particular stakeholder at which each WE quality model is aimed and (2) the measurable concepts, attributes and measures that are applicable to each specific artifact. This fact also contributes to improving P3. Last but not least, this way of representing WE quality models by means of a meta-model instantiation is a machine-readable way, which helps to preserve the development advantages provided by the (semi-)automatic nature of WE processes. This in turn leverages P8. As a proof of concept we have presented a navigational quality model by using the meta-model, together with an indicator. This quality model proposal must be validated empirically in order to determine if the selected indicator and decision criteria are valid (which is one of the requirements presented in Section 2 that are not covered in this work). In fact, this is one of the main further lines of research: defining **empirically validated quality models** that include all the elements referenced in the WE-SMM for each level of abstraction is far from easy, yet it is an unavoidable step if we eventually aim at assuring that the results of the WE quality evaluation process are trustworthy. Another important future line of research is how the improvement of the intermediate models based on the quality evaluation results should be tackled in order to actually assure a good quality of the final product.

Acknowledgments

This paper has been supported by the MEC Spanish Ministry for Staff Stages at foreign Universities (PR2006-0374) and projects CALIPSO (TIN20005-24055-E), MEC-FEDER (TIN2004-03145), ESFINGE (TIN2006-15175-C05-05), METASIGN (TIN2004-00779) and DSDM (TIN2005-25866-E)). Also, is part of the DADASMECA project (GV05/220), financed by the Valencia Government and DIMENSIONS (PBC-05-012-1) financed by the Castilla-La Mancha government.

References

- [1] Lang, M., Fitzgerald, B.: Hypermedia Systems Development Practices: A Survey. *IEEE Software* 22(2), 68–75 (2005)
- [2] ISO/IEC 9126. Software engineering – Product quality – Part 1: Quality model. International Organization for Standardization, Geneva (2001)
- [3] Garvin, D.: What Does “Product Quality” Really Mean?, *Sloan Management Review*, Fall 1984, pp. 25–45 (1984)
- [4] ISO/IEC 14598. Information technology – Software Product Evaluation. International Organization for Standardization. Geneva (1999)
- [5] Fraternali, P., Paolini, P.: Model-driven development of Web applications: the Autoweb System. *ACM Transactions on Information Systems (TOIS)* 18(4), 323–382 (2000)
- [6] Nielsen, J.: *Designing Web Usability: The Practice of Simplicity*. New Riders, Berkeley (2000)
- [7] Ivory, M.Y.: *Automated Web Site Evaluation*. Kluwer Academic Publishers, Norwell (2004)
- [8] ISO/IEC 10027. Information technology – Information Resource Framework. International Organization for Standardization. Geneva (1990)
- [9] Briand, L., Morasca, S., Basili, V.: Defining and Validating Measures for Object-based High-level Design. *IEEE Transactions on Software Engineering* 25(5), 722–743 (1999)
- [10] Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering* 55, 243–276 (2005)
- [11] Moody, D.L., Shanks, G.G.: Improving the quality of data models: empirical validation of a quality management framework. *Information Systems*, vol. 28, pp. 619–650. Elsevier Science Ltd (2003)
- [12] McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality, *Nat’l Tech. Information Service*, vol. 1, 2 and 3 (1977)
- [13] Bohem, B.W.: *Software Engineering Economics*. Prentice Hall Inc., Englewood Cliffs, NJ (1981)
- [14] Dromey, R.G.: A model for software product quality. *IEEE Transactions on Software Engineering* 21(2), 146–162 (1995)
- [15] olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. *IEEE Multimedia Magazine* 9(4), 20–29 (2002)
- [16] 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)
- [17] Moraga, M., Calero, C., Piattini, M.: Ontology Driven Definition of a Usability Model for Second Generation Portals. In: *MATES 2006*, vol. 155, ACM Press, NewYork (2006)
- [18] Comai, S., Matera, M., Maurino, A.: A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) *Advanced Conceptual Modeling Techniques*. LNCS, vol. 2784, pp. 339–350. Springer, Heidelberg (2003)
- [19] Abrahao, S., Insfran, E.: Early usability evaluation in Model-Driven Architecture Environments. In: *Proceedings of the Sixth IEEE International Conference on Quality Software*. IEEE Press, Wiley, Chichester (2006)
- [20] Comai, S., Matera, M., Maurino, A.: A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) *Advanced Conceptual Modeling Techniques*. LNCS, vol. 2784, pp. 339–350. Springer, Heidelberg (2003)
- [21] Krogstie, J., Lindland, O.I., Sindre, G.: Defining quality aspects for conceptual models. *ISCO 1995*: 216-231 (1995)

- [22] Seffah, A., Donyaee, M., Kline, R.B., Padda, H.K.: Usability measurement and metrics: a consolidated model. *Software Quality Journal* 14, 159–178 (2006)
- [23] Ruiz, F., Hilera, J.R.: Using Ontologies in Software Engineering and Technology. In: Calero, C., Ruiz, F., Piattini, M. (eds.) *Ontologies for Software Engineering and Software Technology*, Springer, Heidelberg (2006)
- [24] García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M.: Towards a consistent terminology for software measurement. *Information and Software Technology* 48(8), 631–644 (2005)
- [25] Ferreira, M., García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Braga, J.L.: *Medición del Software: Ontología y Metamodelo*. Technical Report, University of Castilla-La Mancha (2006)

The Use of a Bayesian Network for Web Effort Estimation

Emilia Mendes

The University of Auckland, Private Bag 92019
Auckland, New Zealand, 0064 9 3737599 ext. 86137
emilia@cs.auckland.ac.nz

Abstract. The objective of this paper is to describe the use of a probabilistic approach to Web effort estimation by means of a Bayesian Network. A Bayesian Network is a model that embodies existing knowledge of a complex domain in a way that supports reasoning with uncertainty. Given that the causal system relative to Web effort estimation has an inherently uncertain nature the use of Bayesian model seemed a reasonable choice. We used a cross-company data set of 150 industrial Web projects volunteered from Web companies worldwide, which are part of the Tuketuku database. Results showed that the effort estimates obtained using a Bayesian Network were sound and significantly superior to the prediction based on two benchmark models, using the mean and median effort respectively.

Keywords: Web effort estimation, Bayesian Networks, Effort accuracy, Web cost estimation.

1 Introduction

A cornerstone of Web project management is sound resource estimation, the process by which resources are estimated and allocated effectively, enabling projects to be delivered on time and within budget. Resources are factors, such as cost, effort, quality, ‘problem size’, that have a bearing on a project’s outcome. Within the scope of resource estimation, the causal relationship between factors is not deterministic and has an inherently uncertain nature. E.g. assuming there is a relationship between development effort and an application’s quality, it is not necessarily true that increased effort will lead to improved quality. However, as effort increases so does the *probability* of improved quality. Resource estimation is a complex domain where corresponding decisions and predictions require reasoning with uncertainty.

In Web project management the complete understanding of what factors affect a project’s outcome and the causal relationships between factors is unknown. In addition, as Web development differs substantially from software development 0, there is very little research on resource estimation for software projects that can be readily reused.

Web development, despite being a relatively young industry, initiated just 13 years ago, currently represents a market that increases at an average rate of 20% per year, with Web e-commerce sales alone surpassing 95 billion USD in 2004 (three times the

revenue from the world's aerospace industry)¹[33]. Unfortunately, in contrast, most Web development projects suffer from unrealistic project schedules, leading to applications that are rarely developed on time and within budget [33].

To understand resource estimation for Web projects, previous studies have developed models that use as input, factors such as the size of a Web application, and cost drivers (e.g. tools, developer's quality, team size), and provide an effort estimate as output. The differences between these studies were the number and type of size measures used, choice of cost drivers and occasionally the techniques employed to build resource estimation models. Despite previous studies, to date no complete understanding of which factors affect a Web project's outcome, their causal relationships, and the uncertainty inherent to such relationships has been achieved.

Important reasons for this gap in knowledge are: i) the use of techniques to build resource estimation models that fail to represent the causal relationship between factors and their corresponding uncertainty, and require the use of large amounts of data that is often difficult to obtain; ii) a strong reliance on obtaining the "correct" causal model using simple statistical models, which are inadequate to accommodate complex relationships between all the relevant factors [8]; iii) until recently, the non-existence of appropriate algorithms and corresponding software tools [11] to enable the building of large causal models that allow for uncertainty and probabilistic reasoning; iv) the relatively new research area of resource estimation for Web projects with the first study published in 2000 [19].

There have been numerous attempts to model resource estimation of Web projects, but none yielded a complete causal model incorporating all the necessary component parts. Mendes and Counsell [19] were the first to investigate this field by building a model that used machine-learning techniques with data from student-based Web projects, and size measures harvested late in the project's life cycle. Mendes and collaborators also carried out a series of consecutive studies [10],[18],[19]-[28] where models were built using multivariate regression and machine-learning techniques using data on industrial Web projects. Recently they also proposed and validated size measures harvested early in the project's life cycle, and therefore better suited to resource estimation [22].

Other researchers have also investigated resource estimation for Web projects. Reifer [34] proposed an extension of an existing software engineering resource model, and a single size measure harvested late in the project's life cycle. None were validated empirically. This size measure was later used by Ruhe et al. [35], who further extended a software engineering hybrid estimation technique to Web projects, using a small data set of industrial projects, mixing expert judgement and multivariate regression. Later, Baresi et al. [2], and Mangia et al. [17] investigated effort estimation models and size measures for Web projects based on a specific Web development method. Finally, Costagliola et al. [4] compared two types of Web-based size measures for effort estimation.

The goal of our research is therefore to create and evaluate a large-scale Bayesian network [11] (BN) that represents the causal model for resource estimation of Web projects, incorporating all the fundamental factors and their causal relationships. A BN is a model that embodies existing knowledge of a complex domain in a way that supports reasoning with uncertainty [11][31]. It is a representation of a joint probability

¹ http://www.aia-aerospace.org/stats/aero_stats/stat08.pdf
http://www.tchidagraphics.com/website_ecommerce.htm

distribution over a set of variables, and is made up of two parts. The first, the qualitative part, represents the structure of a BN as depicted by a directed acyclic graph (digraph) (see Fig. 1). The digraph’s nodes represent the relevant variables (factors) from the domain being modelled, which can be of different types (e.g. observable or latent, categorical, numerical). A digraph’s arcs represent probabilistic relationships, i.e. they represent the causal relationships between variables [11][29][41]. The second, the quantitative part, associates a node probability table (NPT) to each node, its probability distribution. A parent node’s NPT describes the relative probability of each state (value) (Fig. 1 “NPT for node Total Effort”); a child node’s NPT describes the relative probability of each state conditional on every combination of states of its parents (Fig. 1 “NPT for node Quality delivered”). So, for example, the relative probability of Quality delivered (QD) being ‘Low’ conditional on Total effort (TE) being ‘Low’ is 0.8, and represented as:

- $p(QD = \text{‘Low’} \mid TE = \text{‘Low’}) = 0.8$

Each column in a NPT represents a conditional probability distribution and therefore its values sum up to 1 [11].

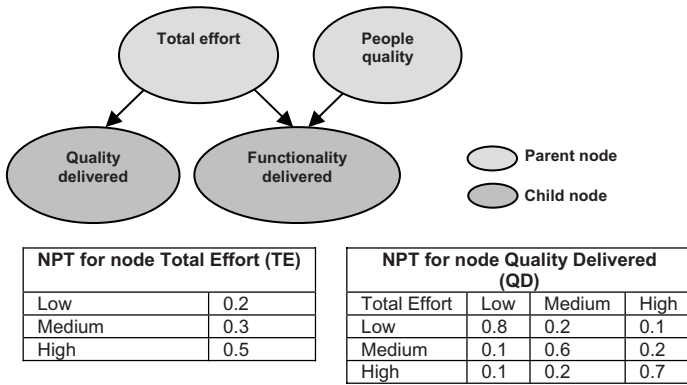


Fig. 1. A small BN model and two NPTs

Formally, the relationship between two nodes is based on Bayes’ rule [11][31]:

$$p(X \mid E) = \frac{p(E \mid X)p(X)}{p(E)} \tag{1}$$

where:

- $p(X \mid E)$ is called the *posterior* distribution and represents the probability of X given evidence E ;
- $p(X)$ is called the *prior* distribution and represents the probability of X before evidence E is given;
- $p(E \mid X)$ is called the *likelihood* function and denotes the probability of E assuming X is true.

Once a BN is specified, evidence (e.g. values) can be entered onto any node, and probabilities for the remaining nodes are automatically calculated using Bayes' theorem [31][41]. Therefore BNs can be used for different types of reasoning, such as predictive and "what-if" analyses to investigate the impact that changes on some nodes have upon others [37].

The BN described and validated in this paper focuses on Web effort estimation. This BN comprises a subset of a more complete BN, and was chosen since this is the only BN within the scope of our research that was built from data on Web projects, as opposed to being elicited from interviews with domain experts. We had the opportunity to gather data on 150 industrial Web projects as part of the Tukatuku Benchmarking project [22], and to use these data to build and validate the BN presented herein. The project data characterises Web projects using size measures and cost drivers targeted at effort estimation. Since we had a dataset of real industrial Web projects, we were also able to compare the accuracy of our Web effort BN to that provided using a mean and median effort models, which are used here as a benchmark. To do so we computed point forecasts for the BN using the method described in [32], to be detailed later.

Prediction accuracy was measured using de facto measures such as the Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdmRE) and Prediction at 25% (Pred(25)) [4].

The remainder of the paper is organised as follows: Section 2 describes the procedure used to build and validate the Web effort BN. Section 3 presents the results using for the Web effort BN, and for the mean and median effort models. Finally, conclusions and comments on future work are given in Section 4.

2 Building the Web Effort BN

2.1 Dataset Description

The analysis presented in this paper was based on data from 150 Web projects of the Tukatuku database [22], which aims to collect data from completed Web projects, to be used to develop Web cost estimation models and to benchmark productivity across and within Web Companies. The Tukatuku includes data on 150 Web hypermedia systems and Web applications [3] where:

- Projects come from 10 different countries, mainly New Zealand (56%), Brazil (12.7%), Italy (10%), Spain (8%), United States (4.7%), England (2.7%), and Canada (2%).
- Project types are new developments (56%) or enhancement projects (44%).
- The applications are mainly Legacy integration (27%), Intranet and eCommerce (15%).
- The languages used are mainly HTML (88%), Javascript (DHTML/DOM) (76%), PHP (50%), Various Graphics Tools (39%), ASP (VBScript, .Net) (18%), and Perl (15%).

Each Web project in the database was characterized by 25 variables, related to the application and its development process (see Table 1). These size measures and cost drivers have been obtained from the results of a survey investigation [22], using data from 133 on-line Web forms aimed at giving quotes on Web development projects. In addition, these measures and cost drivers have also been confirmed by an established Web company and a second survey involving 33 Web companies in New Zealand. Consequently it is our belief that the 25 variables identified are measures that are meaningful to Web companies and are constructed from information their customers can provide at a very early stage in project development.

Table 1. Tukutuku database variables

Variable name	Scale	Description
COMPANY DATA		
Country	Categorical	Country company belongs to.
Established	Ordinal	Year when company was established.
nPeopleWD	Ratio	Number of people who work on Web design and development.
PROJECT DATA		
TypeProj	Categorical	Type of project (new or enhancement).
nLang	Ratio	Number of different development languages used
DocProc	Categorical	If project followed defined and documented process.
ProImpr	Categorical	If project team involved in a process improvement programme.
Metrics	Categorical	If project team part of a software metrics programme.
Devteam	Ratio	Size of project's development team.
Teamexp	Ratio	Average team experience with the development language(s) employed.
TotEff	Ratio	Actual total effort in person hours used to develop the Web application.
estEff	Ratio	Estimated total effort in person hours necessary to develop the Web application.
Accuracy	Categorical	Procedure used to record effort data.
WEB APPLICATION		
TypeApp	Categorical	Type of Web application developed.
TotWP	Ratio	Total number of Web pages (new and reused).
NewWP	Ratio	Total number of new Web pages.
TotImg	Ratio	Total number of images (new and reused).
NewImg	Ratio	Total number of new images created.
Fots	Ratio	Number of features reused without any adaptation.
HFotsA	Ratio	Number of reused high-effort features/functions adapted.
Hnew	Ratio	Number of new high-effort features/functions.
totHigh	Ratio	Total number of high-effort features/functions
FotsA	Ratio	Number of reused low-effort features adapted.
New	Ratio	Number of new low-effort features/functions.
totNHigh	Ratio	Total number of low-effort features/functions

Within the context of the Tukutuku project, a new high-effort feature/function requires at least 15 hours to be developed by one experienced developer, and a high-effort adapted feature/function requires at least 4 hours to be adapted by one experienced developer. These values are based on collected data.

Table 2 summarises the number and percentages of projects for the categorical variables, and summary statistics for the numerical variables from the Tukutuku database are given in Table 3.

Table 2. Summary of Number of Projects and Percentages for Categorical variables

Variable	Level	Num. Projects	% Projects
<i>TypeProj</i>	Enhancement	66	44
	New	84	56
<i>DocProc</i>	No	53	35.3
	Yes	97	64.7
<i>ProImpr</i>	No	77	51.3
	Yes	73	48.7
<i>Metrics</i>	No	85	56.7
	Yes	65	43.3

Table 3. Summary Statistics for Numerical variables

	Mean	Median	Std. Dev.	Min.	Max.
nlang	3.75	3.00	1.58	1	8
DevTeam	2.97	2.00	2.57	1	23
TeamExp	3.57	3.00	2.16	1	10
TotEff	564.22	78.00	1048.94	1	5000
TotWP	81.53	30.00	209.82	1	2000
NewWP	61.43	14.00	202.78	0	1980
TotImg	117.58	43.50	244.71	0	1820
NewImg	47.62	3.00	141.67	0	1000
Fots	2.05	0.00	3.64	0	19
HFotsA	12.11	0.00	66.84	0	611
Hnew	2.53	0.00	5.21	0	27
totHigh	14.64	1.00	66.59	0	611
FotsA	1.91	1.00	3.07	0	20
New	2.91	1.00	4.07	0	19
totNHigh	4.82	4.00	4.98	0	35

As for data quality, we asked companies how their effort data was collected (see Table 4). At least for 83% of Web projects in the Tukutuku database effort values were based on more than guesstimates.

Table 4. How Effort data was gathered

Data Collection Method	# of Projects	% of Projects
Hours worked per project task per day	93	62
Hours worked per project per day/week	32	21.3
Total hours worked each day or week	13	8.7
No timesheets (guesstimates)	12	8

2.2 Procedure Used to Build the BNs

The BN presented in this paper was built and validated using an adapted Knowledge Engineering of Bayesian Networks (KEBN) process [6][16][41] (see Fig. 2). In Fig. 2

arrows represent flows through the different processes, depicted by rectangles. Such processes are executed either by people – the Knowledge Engineer (KE) and the Domain Experts (DEs) [41] (white rectangles), or automatic algorithms (dark grey rectangles). Within the context of this research project this author is the knowledge engineer, and Web project managers from Web companies in Rio de Janeiro and Auckland are the domain experts.

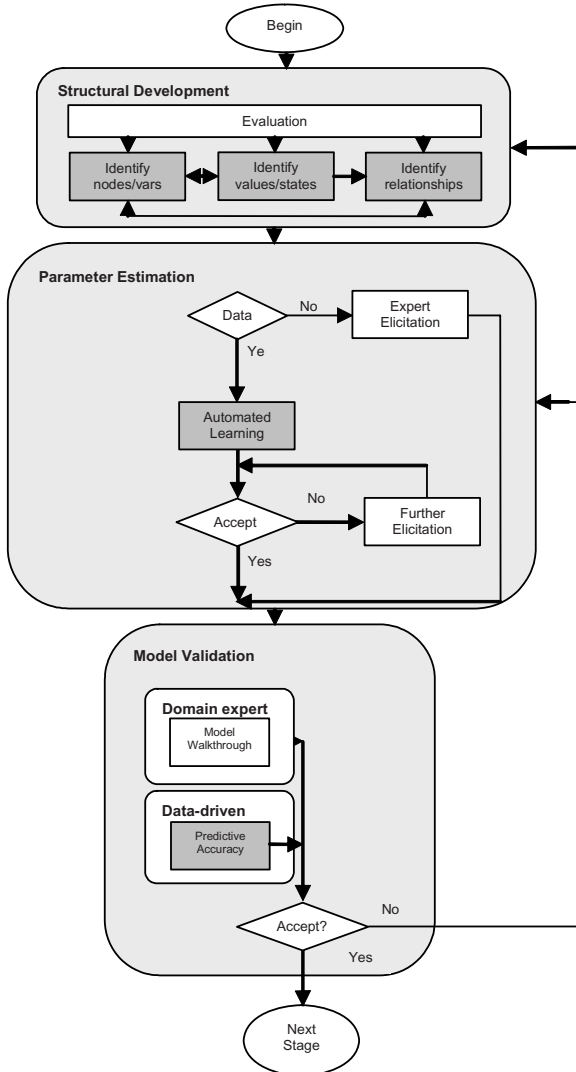


Fig. 2. KEBN, adapted from [40]

The three main steps within our KEBN process are the Structural Development, Parameter Estimation, and Model Validation. This process iterates over these steps until a complete BN is built and validated. Each of these three steps is detailed below:

Structural Development: This step represents the qualitative component of a BN, which results in a graphical structure comprised of, in our case, the factors (nodes, variables) and causal relationships identified as fundamental for resource estimation of Web projects. This is an iterative process where independent BN's sub-models are identified. This model construction process has been validated in previous studies [7][9][16][29][41] and uses the principles of problem solving employed in data modelling and software development [39]. Also, the BN tool we used (Hugin Expert) allows for the representation of sub-models, thus facilitating the application of our modelling approach. Existing literature in Web resource estimation, data from the Tukutuku database and current knowledge from domain experts are employed to elicit the BN's structure. In the context of this paper we have used data from the Tukutuku database and current knowledge from a domain expert who works in a well-established Web company in Rio de Janeiro (Brazil). The identification of nodes, values and relationships was initially obtained automatically using Hugin, and later modified once feedback was obtained from the domain expert and the conditional independences were checked. In addition to identifying variables, their types (e.g. query variable, evidence variable) and relationships, domain experts in general also choose what states (values) each variable should take, and if they are discrete or continuous. In practice, currently available BN tools require that continuous variables be discretised by converting them into multinomial variables [14], also the case with Hugin Expert. Hugin offers two discretisation algorithms – equal-width intervals [36], whereby all intervals have equal size, and equal-frequency intervals, whereby each interval contains n/N data points where n is the number of data points and N is the number of intervals (this is also called maximal entropy discretisation [40]). We used equal-frequency intervals as suggested in [13], and five intervals. Automatic discretisation frees domain experts and knowledge engineers from having to statically discretise variables manually [30]. Throughout this step the knowledge engineer also evaluated the structure of the BN in two stages. The first entailed checking if [14]: variables and their values have a clear meaning; all relevant variables for that cycle have been included; variables are named conveniently; all states are appropriate (exhaustive and exclusive); a check for any states that can be combined. The second stage entailed reviewing the graph structure of the BN to make sure any identified d-separation dependencies comply with the types of variables used and causality assumptions. D-separation dependencies are used to identify variables influenced by evidence coming from other variables in the BN [11][31]. Once the BN structure is assumed to be close to final we may still need to optimise this structure to reduce the number of probabilities that need to be assessed for the network. If optimisation is needed then we employ techniques that change the graphical structure (e.g. divorcing [11]) and the use of parametric probability distributions (e.g. noisy-OR gates [7][31]). In the case of the Web effort BN we changed its original graphical structure to maintain the conditional independence of the nodes (see Section 2.3), however divorcing was not employed in order to keep only nodes that had been elicited from the Tukutuku data.

Parameter Estimation: This step represents the quantitative component of a BN, which results in conditional probabilities, obtained via Expert Elicitation or automatically, which quantify the relationships between variables [11][14]. For the Web effort BN, they were obtained using two steps: first, by automatically fitting a sub-network to a subset of the Tuketuku dataset (Automated learning); second, by obtaining feedback from the domain expert regarding the suitability of priors and conditional probabilities that were automatically fitted. No previous literature was used in this step since none reported probabilistic information. Of the 150 projects available in the Tuketuku database we used 120 (80%) to build the Web effort BN and later employed the remaining 30 for the Model Validation step to assess the BN's effort prediction accuracy.

Model Validation: This step validates the BN that results from the two previous steps, and determines whether it is necessary to re-visit any of those steps. Two different validation methods are used - Model Walkthrough and Predictive Accuracy, which specifically verifies if resource predictions provided by a BN are, on average, better than those currently used by Web companies. Predictive Accuracy is normally carried out using quantitative data, thus this was the validation approach we employed to validate the Web effort BN. Accuracy was measured using de facto measures such as the Mean MRE, median MRE and Pred(25), and estimated effort for each of the 30 projects in the validation set was obtained using a point forecast, computed using the method described in [32]. This method calculates the joint probability distribution of effort using the belief distribution [31], and computes estimated effort as the sum of the probability of a given effort scale point multiplied by its related mean effort. Within the context of our Web effort BN, effort was discretised using a five-scale point (see Section 2.3).

Model walkthrough represents the use of real case scenarios that are prepared and used by domain experts to assess if the predictions provided by a BN, or BN's sub-model, correspond to the predictions experts would have chosen based on their own expertise. Success is measured as the frequency with which the BN's predicted value for a target variable (e.g. quality) that has the highest probability corresponds to experts' own assessment. We did not employ a model walkthrough to validate the Web effort BN because we had already carried out a Predictive accuracy procedure using real data volunteered by numerous Web companies worldwide.

2.3 The Web Effort BN

Fig. 3(a) shows the original Web effort BN obtained from fitting the data on Web projects. We used the entire Tuketuku database when building the structure, however for parameter estimation we only employed the 120 projects in the training set, otherwise the point estimates would be biased.

Once this structure was obtained using the Necessary Path Condition (NPC) algorithm [38], it was validated with a domain expert, resulting in the structure presented in Figure 3(b). The main changes to the original structure were related to node *TypeProj*, from which all causal relationships, except for *TotalEffort*, were removed. There were also several changes relating to the three categorical variables *Documented Process*, *Process Improvement* and *Use Metrics*. In the validated structure (see Figure 3(b)), *Process Improvement* presents a relationship with both *Use Metrics*

and *Documented Process*, indicating that it is an important factor determining whether a Web company adheres to the use of metrics and to the use of a documented process. This structure also relates *Use Metrics* to *Documented Process*, indicating that companies that measure attributes to some extent document their processes. The number of languages to be used in a project (*numLanguages*) and the average number of years of experience of a team (*Team Experience*) are also related with the size of the development team (*sizeDevTeam*). The nodes relative to Web size measures (e.g. *NewWP*) remained unchanged as the data already captured the strong relationship between size and effort.

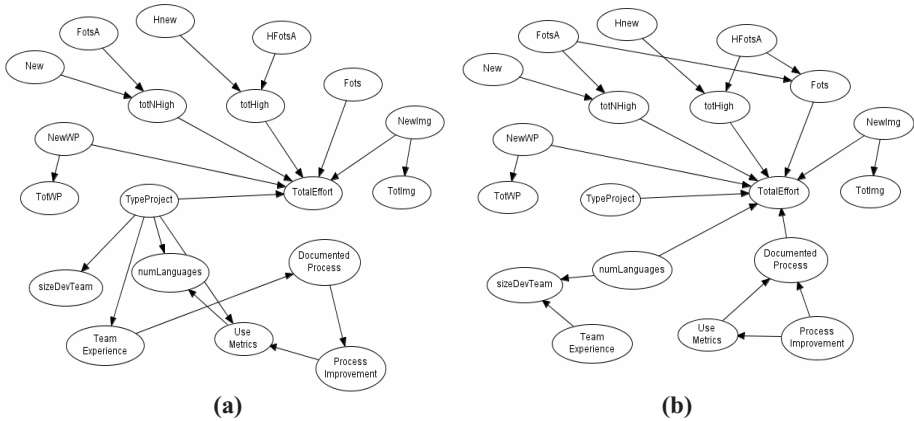


Fig. 3. Original BN (a) and BN after evaluation with DE (b)

Once the structure had been validated, our next step was to ensure that the conditionally independent variables (nodes) in the Web effort BN were really independent of each other [31]. Whenever two variables were significantly associated we also measured their association with effort, and the one with the strongest association was kept. For example, *Process improvement* was significantly associated with *Fots*, so one of these nodes had to be removed from the BN. Given that *Process Improvement* had a significant association with *TotalEffort* stronger than the association between *TotalEffort* and *Fots*, we kept *Process Improvement* in the model. This was an iterative process given that once nodes are removed (e.g. *FotsA*, *New*), other nodes become conditionally independent (e.g. *totNHigh*) and so need to be checked as well. The associations between the numerical variables were assessed using a non-parametric test - Spearman's rank correlation test; the associations between numerical and categorical variables were checked using the one-way ANOVA test, and the associations between categorical variables were checked using the Chi-square test. All tests were carried out using SPSS 12.0.1 and $\alpha = 0.05$.

Fig. 4 shows the Web effort BN after all conditional independences were checked. This was the Web effort BN used as input to the *Parameter estimation* step, where prior and conditional probabilities were automatically generated using the EM-learning algorithm [15], and later validated by the DE.

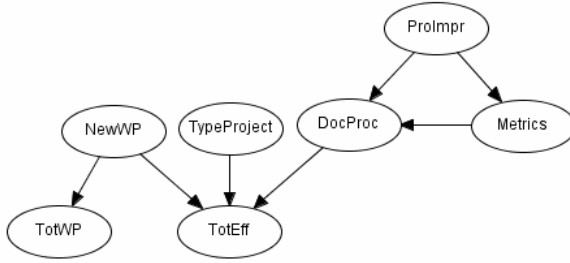


Fig. 4. BN after conditional independences were checked

Effort was discretised into five discrete approximations, described in Table 5.

Table 5. How Effort data was gathered

Categories	Range (person hours)	Mean Effort
Very low	≤ 12.55	5.2
Low	> 12.55 and ≤ 33.8	22.9
Medium	> 33.8 and ≤ 101	63.1
High	> 101 and ≤ 612.5	314.9
Very High	> 612.5	2,238.9

TotWP and NewWP were also discretised into five discrete approximations. There are no strict rules as to how many discrete approximations should be used. Some studies have employed three [32], others five [9], and others eight [37]. We chose five. However, further studies are necessary to determine whether a different number of approximations leads to results significantly different. The NPTs for the seven nodes used in the Web effort BN are not presented here due to lack of space.

3 Measuring the Prediction Accuracy of the Web Effort BN

The 30 Web projects in the validation set were used to measure the prediction accuracy of the Web effort BN model. In addition, we also used the mean (526.9) and median (59.1) effort models as benchmark. Prediction accuracy was measured using MMRE, MdMRE, and Pred(25) and Table 6 shows that the MMRE and MdMRE obtained using the BN model was very close to the baseline predictions suggested in the literature (MMRE and MdMRE $\leq 25\%$). However, Pred(25) was lower than the suggested baseline of 75% or above. In addition, Table 6 also shows that the prediction accuracy for the Web Effort BN model was superior to the accuracy obtained with either the mean or median effort models.

In order to assess if the difference in accuracy between the Web Effort BN model and the mean & median models was not due to chance we also used a statistical significance test to compare the absolute residuals (actual effort – estimated effort) between these three models. Since none of the residuals were normally distributed,

confirmed used the One-Sample Kolmogorov-Smirnov Test, they were compared using the non-parametric Wilcoxon Signed Paired Test. This test confirmed that the predictions obtained using the Web Effort BN model were significantly superior to the predictions from both the median and mean models. In addition, this test also showed that there were no significant differences between the median and mean effort models.

Table 6. Accuracy Measures for the Web Effort BN and Benchmarking models

Accuracy (%)	BN model	Mean model	Median model
MMRE	34.26	1106.31	132.76
MdMRE	27.42	252.36	85.90
Pred(25)	33.33	6.67	10.00

Fig. 5 shows boxplots of absolute residuals for the Web effort BN (ResBN), mean (Mean) and median (Median) models. The median of ‘ResBN’ is much lower than the median of ‘Mean’, and also lower than the median of ‘Median’. All boxplots present outliers, however those for ‘Mean’ and ‘Median’ are much worse than the ones for ‘ResBN’. The boxes for ‘ResBN’ and ‘Mean’ are flatter than the box for ‘Median’. What these results suggest is that using a model that allows the representation of uncertainty, which is inherent in effort estimation, can outperform other commonly used benchmarking models, based on the mean or median effort. In addition, these results also suggest that Web companies that either volunteered projects to the Tukutuku database, or develop similar projects to those in that database, would benefit from using a Bayesian Network to obtain effort estimates, compared to simply relying on estimated based on the mean or median effort of past projects.

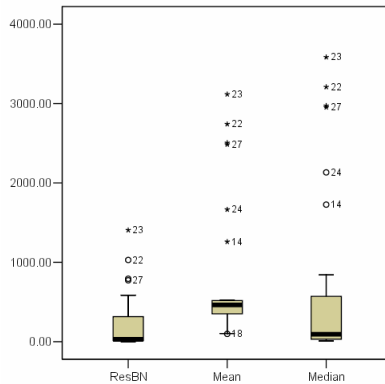


Fig. 5. Boxplots with distribution of residuals

The Web effort BN model presented in this paper is a very simple model, built using a dataset that does not represent a random sample of projects, therefore these results have to be interpreted with care. In addition, we chose to use only the nodes identified using the Tukutuku dataset, i.e., other nodes that could have been identified

by the DE were not included. We also wanted to investigate to what extent a BN model and probabilities generated using automated algorithms available in HUGIN would provide predictions comparable to those obtained using mean and median models.

There are several issues regarding the validity of our results: i) the choice of discretisation, structure learning, parameter estimation algorithm, and the number of categories used in the discretisation all affect the results and there are no clear-cut guidelines on the best combination to use. This means that further investigation is paramount; ii) the Web effort BN presented in this study might have been quite different had it been entirely elicited from DEs, and this is part of our future work; iii) the decision as to what conditional independent nodes to retain was based on their strength of association with TotalEffort, however other solutions could have been used, e.g. ask a DE to decide; iv) obtaining feedback from more than one DE could also have influenced the BN structure in Fig. 3(b), and this is also part of our future work .

Finally, the use of BN tools by practitioners may still prove to be a challenge given that there are still many interface and technical issues that do not make their use straightforward.

4 Conclusions

This paper has presented the results of an investigation where a dataset containing data on 120 Web projects was used to build a Bayesian model, and the predictions obtained using this model were compared to those obtained using the mean and median effort models, based on a validation set with 30 projects.

The predictions obtained using the Web effort BN was significantly superior to the median-based and mean-based predictions, despite the use of a simple BN model. Future work entails: the building of a second Web effort BN based solely on domain experts' knowledge, to be compared to the BN presented in this paper; aggregation of this BN to our large Web resource BN, to obtain a complete causal model for Web resource estimation.

Acknowledgments. We would like to thank all the Web companies that volunteered the data used in this study. We would also like to thank Dr. N. Mosley for his comments on a previous version of this paper and Associate Professor Filomena Ferrucci for volunteering data on 15 projects to the Tukutuku project. This work is sponsored by the Royal Society of New Zealand, under the Marsden Fund research grant 06-UOA-201 MIS.

References

- [1] Baresi, L., Morasca, S., Paolini, P.: An empirical study on the design effort for Web applications. In: Proceedings of WISE 2002, pp. 345–354 (2002)
- [2] Baresi, L., Morasca, S., Paolini, P.: Estimating the design effort for Web applications. In: Proceedings of Metrics 2003, pp. 62–72 (2003)
- [3] Christodoulou, S.P., Zafiris, P.A., Papatheodorou, T.S.: WWW2000: The Developer's view and a practitioner's approach to Web Engineering. In: Proc. Second ICSE Workshop on Web Engineering, 4 and 5 June 2000, Limerick, pp. 75–92 (2000)

- [4] Conte, S.D., Dunsmore, H.E., Shen, V.Y.: *Software Engineering Metrics and Models*, Benjamin-Cummins (1986)
- [5] Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Vitiello, G.: Effort estimation modeling techniques: a case study for web applications. In: *Procs. Intl. Conference on Web Engineering (ICWE'06)*, pp. 9-16 (2006)
- [6] Druzdel, M.J., Onisko, A., Schwartz, D., Dowling, J.N., Wasyluk, H.: Knowledge engineering for very large decision-analytic medical models. In: *Proceedings of the 1999 Annual Meeting of the American Medical Informatics Association*, pp. 1049-1054 (1999)
- [7] Druzdel, M.J., van der Gaag, L.C.: Building Probabilistic Networks: Where Do the Numbers Come From? *IEEE Trans. on Knowledge and Data Engineering* 12(4), 481-486 (2000)
- [8] Fenton, N., Krause, P., Neil, M.: Software Measurement: Uncertainty and Causal Modeling, *IEEE Software*, pp. 116-122 (2002)
- [9] Fenton, N., Marsh, W., Neil, M., Cates, P., Forey, S., Taylor, M.: Making Resource Decisions for Software Projects. In: *Proc. ICSE'04*, pp. 397-406 (2004)
- [10] Fewster, R., Mendes, E.: Measurement, Prediction and Risk Analysis for Web Applications. In: *Proceedings of IEEE Metrics Symposium*, pp. 338-348 (2001)
- [11] Jensen, F.V.: *An introduction to Bayesian networks*. UCL Press, London (1996)
- [12] Kitchenham, B.A., Mendes, E.: A Comparison of Cross-company and Single-company Effort Estimation Models for Web Applications. In: *Proceedings EASE 2004*, pp. 47-55 (2004)
- [13] Knobbe, A.J., Ho, E.K.Y.: Numbers in Multi-Relational Data Mining. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) *PKDD 2005*. LNCS (LNAI), vol. 3721, Springer, Heidelberg (2005)
- [14] Korb, K.B., *Bayesian, A.E.N: Artificial Intelligence*. CRC Press, USA (2004)
- [15] Lauritzen, S.L.: The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis* 19, 191-201 (1995)
- [16] Mahoney, S.M., Laskey, K.B.: Network Engineering for Complex Belief Networks. In: *Proc. Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 389-396 (1996)
- [17] Mangia, L., Paiano, R.: MMWA: A Software Sizing Model for Web Applications. In: *Proc. Fourth International Conference on Web Information Systems Engineering*, pp. 53-63 (2003)
- [18] Mendes, E., Kitchenham, B.A.: Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In: *Proc. IEEE Metrics*, pp. 348-357 (2004)
- [19] Mendes, E., Counsell, S.: Web Development Effort Estimation using Analogy. In: *Proc. 2000 Australian Software Engineering Conference*, pp. 203-212 (2000)
- [20] Mendes, E., Mosley, N., Counsell, S.: Web Effort Estimation. In: Mendes, E., Mosley, N. (eds.) *Web Engineering*, pp. 29-73. Springer-Verlag, Heidelberg (2005)
- [21] Mendes, E., Mosley, N.: Further Investigation into the Use of CBR and Stepwise Regression to Predict Development Effort for Web Hypermedia Applications. In: *Proc. ACM/IEEE ISESE*, Nara, Japan, pp. 79-90 (2002)
- [22] Mendes, E., Mosley, N., Counsell, S.: Investigating Web Size Metrics for Early Web Cost Estimation. *Journal of Systems and Software* 77(2), 157-172 (2005)
- [23] Mendes, E., Mosley, N., Counsell, S.: A Replicated Assessment of the Use of Adaptation Rules to Improve Web Cost Estimation. In: *Proc. ISESE*, pp. 100-109 (2003)
- [24] Mendes, E., Mosley, N., Counsell, S.: Early Web Size Measures and Effort Prediction for Web Costimation. In: *Proceedings of the IEEE Metrics Symposium*, pp. 18-29 (2003)
- [25] Mendes, E., Mosley, N., Counsell, S.: Comparison of Length, complexity and functionality as size measures for predicting Web design and authoring effort. In: *IEE Proc. Software* 149(3), 86-92 (2002)

- [26] Mendes, E., Mosley, N., Counsell, S.: Web metrics - Metrics for estimating effort to design and author Web applications. *IEEE MultiMedia*, pp. 50–57 (January-March, 2001)
- [27] Mendes, E., Mosley, N., Watson, I.: A Comparison of Case-Based reasoning Approaches to Web Hypermedia Project Cost Estimation. In: *Proc. WWW'02* (2002)
- [28] Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S.: A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. *ESE* 8(2), 163–196 (2003)
- [29] Neil, M., Fenton, N., Nielsen, L.: “Building Large-scale bayesian networks”, *The knowledge Engineering Review*. *KER* 15(3), 257–284 (2000)
- [30] Neil, M., Taylor, M., Marquez, D., Fenton, N., Hearty, P.: Modeling Dependable Systems using Hybrid Bayesian Networks. In: *Proc. BND Workshop*, pp. 817–823 (2006)
- [31] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco (1988)
- [32] Pendharkar, P.C., Subramanian, G.H., Rodger, J.A.: A Probabilistic Model for Predicting Software Development Effort. *IEEE Trans. Software Eng.* 31(7), 615–624 (2005)
- [33] Reifer, D.J.: Web Development: Estimating Quick-to-Market Software. *IEEE Software*, 57–64 (2000)
- [34] Reifer, D.J.: Ten deadly risks in Internet and intranet software development. *IEEE Software*, 12–14 (2002)
- [35] Ruhe, M., Jeffery, R., Wieczorek, I.: Cost estimation for Web applications. In: *Proceedings ICSE 2003*, pp. 285–294 (2003)
- [36] Silverman, B.W.: *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, Sydney (1986)
- [37] Stamelos, I., Angelis, L., Dimou, P., Sakellaris, E.: On the use of Bayesian belief networks for the prediction of software productivity. *Information and Software Technology* 45(1), 51–60 (2003)
- [38] Steck, H., Tresp, V.: Bayesian Belief Networks for Data Mining. In: *Proceedings of The 2nd Workshop on Data Mining und Data Warehousing, Sammelband* (September 1999)
- [39] Studer, R., Benjamins, V.R., Fensel, D.: Knowledge engineering: principles and methods. *Data & Knowledge Engineering* 25, 161–197 (1998)
- [40] Wong, A.K.C., Chiu, D.K.Y.: Synthesizing Statistical Knowledge from Incomplete Mixed-mode Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9(6), pp. 796–805 (1987)
- [41] Woodberry, O., Nicholson, A., Korb, K., Pollino, C.: Parameterising Bayesian Networks. In: *Proc. Australian Conference on Artificial Intelligence*, pp. 1101–1107 (2004)

Sequential Pattern-Based Cache Replacement in Servlet Container

Yang Li, Lin Zuo, Jun Wei, Hua Zhong, and Tao Huang

Technology Center of Software Engineering, Institute of Software
Chinese Academy of Sciences, Beijing 100080, P.R. China

{fallingboat, martin_zl, wj, zhongh, tao}@otcaix.iscas.ac.cn

Abstract. Servlet cache can effectively improve the throughput and reduce response time experienced by customers in servlet container. An essential issue of servlet cache is cache replacement. Traditional solutions such as LRU, LFU and GDSF only concern some intrinsic factors of cache objects regardless of associations among cached objects. For higher performance, some approaches are proposed to utilize these associations to predict customer visit behaviors, but they are still restricted by first-order Markov model and lead to inaccurate predication. In this paper, we describe associations among servlets as sequential patterns and compose them into pattern graphs, which eliminates the limitation of Markov model and achieve more accurate predictions. At last, we propose a discovery algorithm to generate pattern graphs and two predictive probability functions for cache replacement based on pattern graphs. Our evaluation shows that this approach can get higher cache hit ratio and effectively improve the performance of servlet container.

Keywords: Servlet Cache, Sequential Patterns, Cache Replacement.

1 Introduction

Recently, Java EE has become mainstream middleware platform for large-scale enterprise applications. As the core part, servlet container [1] provides runtime environment for servlet components and plays a key role in the performance of Web applications. However, servlet performance report [2] presented by Web Performance Inc. in 2004 showed that challenges still exist to improve the performance of servlet container. Many solutions based on cache technology are proposed to gain performance improvement in web server and proxy server [4] [5]. But they are hard to adapt to servlet container because responses generated by servlets are dynamic and dependent on values of input parameters.

In servlet container, there exist lots of servlets which are frequently accessed, and their outputs or responses keep unchanged or stable in some period if their parameter values are unchanged, e.g. catalog pages within a shopping application or an events calendar on a university web site [5]. We call them as cacheable servlets. Therefore, the responses of these cacheable servlets can be cached and accessed by multiple clients to significantly improve response time experienced by customers. Cacheable servlets have been supported in Websphere [6].

Cache replacement is an essential issue in servlet cache. Traditional cache replacement algorithms such as LRU [3], LFU [7] and GDSF [12] mainly utilize a combination of several intrinsic factors of cache objects (e.g. file size, the recentness and frequency) to design replacement cost functions, which can enhance cache hit ratio and performance of servers to a certain extent. However, they ignore business functionalities represented by servlets and business associations among them.

Some researchers have observed these associations among business components and adopted them in different areas. Access patterns [8] are firstly introduced to database system to represent associations among data items. Access patterns are also utilized to prefetch cached web objects by mining server logs to achieve the associations among web pages [9]. Qiang [14] also proposes one discovery algorithm for proxy server based on web access pattern mining and gives the n-gram replacement algorithm to improve cache hit ratio. In general, access patterns mainly includes association rule [9] and sequential pattern [9][10][11], and sequential pattern is widely used in Web system.

Markov model is often adopted to describe sequential patterns. Some researchers have made use of associations among web pages to predict customer visit behaviors for link prediction and path analysis [21][22], but most of them are under first-order Markov model, which assumes the future visit behaviors of customers are only influenced by their current behaviors. Therefore, first-order Markov model can not accurately reflect customer visit behaviors at all time because it brings some inexistent behaviors from multiple steps predictions and leads to invalid replacements. For example, suppose there exist five servlets S_1, S_2, S_3, S_4, S_5 , and customer visit behaviors are $S_1 \rightarrow S_3 \rightarrow S_4$ (10 times) and $S_2 \rightarrow S_3 \rightarrow S_5$ (10 times). The predictions based on first-order Markov model are described in Fig.1, where direct predictions are accurate (Figure1. a). However, the two steps predictions (Figure1.b) include inaccurate behaviors $S_1 \rightarrow S_5$ and $S_2 \rightarrow S_4$, which never happened in actual behaviors.

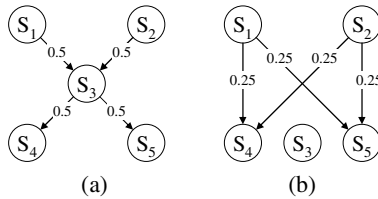


Fig. 1. The Predictions Based on First-order Markov Model

In this paper, we proposed a novel cache replacement approach to support servlet cache. We first describe the business associations among servlets as sequential patterns and compose them to form pattern graphs, which do not suffer the invalid predictions of first-order Markov model and reflect customer visit behaviors more accurately. Then, we present a discovery algorithm to discover above pattern graphs according to customer visit histories, and two predictive probability functions based on pattern graphs are presented, which can be combined with traditional cost functions to improve the effectiveness of cache replacement. We discuss this approach in the context of servlet container. However, we believe the principle idea can be applied

to other dynamic environments, such as ASP and PHP. Our evaluation shows that this approach greatly enhances cache hit ratio, reduces redundant reexecution of cacheable servlets and gains more attractive performance in servlet container.

In the next section, we briefly review related work. Some basic definitions and cache system model of servlet container are presented in section 3. The definitions of sequential patterns and sequential patterns discovery algorithm is described in section 4. Section 5 presents sequential pattern-based replacement cost functions. In section 6, we provide simulation and performance evaluation of the proposed approach and conclude the paper in section 7.

2 Related Works

Many cache replacement algorithms use a combination of several factors such as the visit time and visit frequency, the size, and the cost of fetching a document, which lead to a significant improvement in cache hit ratio and latency reduction in proxy server and web server. LRU [3] replaces cache object with the oldest visit time. LFU [7] evicts cache object with minimal visit frequency. LRU-threshold [13] extends LRU and only caches the object whose size is less than a threshold value. GDSF [12][14] is based on visit time, file size, visit frequency and loading cost etc, which evicts cache object with minimal replacement cost. However, all these algorithms only utilize some factors of cache objects itself (visit time, visit frequency, file size, loading cost, and so on) as criterion to replace cache objects, and ignore associations among cache objects which can be applied to cache replacement and greatly enhance cache hit ratio. In addition, they mainly suit for static files and are hard to handle dynamic servlet responses.

Some approaches are proposed to utilize sequential patterns to describe business association among cache objects. Agrawal [15] first adopts sequential patterns to describe the associations among data items in database and present three algorithms to discover them. Huang [16] analyzes user access sequential patterns and design a prediction-based proxy server to improve hit ratio of accessed documents. Sarukkai [21] uses Markov chain to solve probabilistic link prediction and path analysis in web server. To improve the efficiency of link prediction, Zhu [22] uses Markov model to construct the structure of a Web site based on past visitor behaviors. However, these approaches only adapt to static objects and can not support cacheable servlet responses which depend on the values of parameters. In addition, some of them are base on the first-order Markov model in which the objects that a client visits in future are only determined by its current position. Therefore, these approaches can not accurately reflect business association among servlets on multiple steps predictions.

Compared to current approaches, our contributions are summarized as follows: 1) we describe business associations among servlet as sequential patterns. 2) We utilize pattern graphs to compose above sequential patterns, which can reflect multiple steps associations and get rid of the inaccuracy of Markov model. 3) We design a discovery algorithm discoverTPG to gather pattern graphs at runtime. 4) Based on pattern graphs, we propose two sequential pattern-based replacement cost functions to makes a supplement for traditional replacement cost functions. To the best of our knowledge, this is the first attempt to utilize sequential pattern in servlet cache.

3 Servlet Cache Model

In this section, we present some basic definitions about servlet cache and discuss servlet cache model in servlet container.

3.1 Basic Definitions

Definition 1. Servlet s can be defined by $s = \langle id, name, parameters, type \rangle$ where id and $name$ are identity and name of s , $parameters$ is a group of parameters of s , $type$ is the type of s and $type \in \{cacheable, non-cacheable\}$.

When $type$ is *cacheable*, it means responses generated by s can be cached. On the contrary, responses of s can not be cached if $type$ is *non-cacheable*. In servlet container we represent all servlets by S , all cacheable servlets by S^C and non-cacheable servlets by S^{NC} . For convenience, we put all cacheable servlet before non-cacheable servlets in S , which means $s_i.type=cacheable$ when $1 < i \leq L$ and $s_i.type=non-cacheable$ when $L < i \leq N$ if $S = \{s_1, s_2, \dots, s_i, \dots, s_N\}$ and $|S| = N, |S^C| = L$.

Cacheable servlets are often defined in a configure file according to their business functionalities by application developers, which generally only return some query information such as product information or history of stock. They are generally visited frequently but generated responses are changed rarely when values of parameters are identical. We can cache their responses to service multiple customers, which makes that servlet container need not execute cacheable servlets to generate same responses for each customer at each time.

Definition 2. Servlet cache object is defined as a five-tuples: $scache = \langle id, s, values, response, factors \rangle$ where

- id denotes a unique identity of cache object
- s denotes a cacheable servlet
- $values$ denotes the values of $s.parameters$
- $response$ denotes the response generated by s when the values of $s.parameters$ are values
- $factors$ denotes factors of cost function

Comparing with static files, servlet cache object depends on not only cacheable servlet but also its parameter values. The $scache.factors$ is different according to specific cache replacement algorithms. For LRU, the $scache.factors$ is $\{age\}$, and the $scache.factors$ for GDSF is $\{age, frequency, cost, size\}$. The meanings of the factors will be detailed in section 5.

Web Characterization Activity (WCA) defines a user session as the click-stream of page views for a single user across the entire web site [23]. In servlet container we also define a session just like what they do.

Definition 3. A session represents a sequence of servlets visited by a customer, which can be defined as $session = \{id_1, id_2, id_m, \dots, id_{length}\}$, where id_m is id of a servlet and $length$ represents the length of $session$.

Next, we will detail the cache model of servlet container.

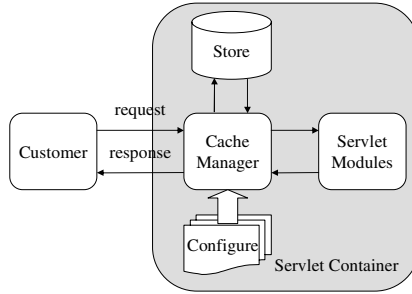


Fig. 2. Servlet Cache Model

3.2 Servlet Cache Model

Fig.2 describes the cache model of servlet container. Here **configure** defines the cacheable servlets; **cache manager** stores **servlet cache objects** into **store** and manages their consistency; **servlet modules** generate response to service customers.

Algorithm 1 explains how above cache model works. `ServletContainer.execute(s, values)` means execution of *s* with values of *s.parameters* (line 5). `CacheManager.get(s.name, values)` means to lookup corresponding *scache* in store according to *s.name* and its parameters values (line 9). `CacheManager.updateFactors(scache)` means to update some factors of *scache* such as visit time and frequency (line 11).

From Algorithm 1, we can see that the process flows are different according to servlet type. If *s.type* is non-cacheable, servlet container will directly execute *s* with the values of *s.parameters* (line 5) and return generated response to customer (line 6). But when *s.type* is cacheable, servlet container will first lookup corresponding *scache* from store (line 9). If *scache* exists, some factors of *scache* are updated, and *scache.response* is returned to customer (line 12). If *scache* does not exist, servlet container does what it does for non-cacheable servlet (line 15) and creates new *scache* object to store into store (line 16, 17). When the store overflows upper limit, some *scaches* are evicted according to replacement cost function `rank(scache)` (line 19).

Obviously, replacement cost function **rank (scache)** is the key of cache replacement, which determinates which *scaches* should be evicted. Next we will detail our approach, and describe two more effective replacement cost functions based on this approach in section 5.

```

1.  s: servlet that client requests
2.  values: the values of parameters of s
3.  function request( s, values){
4.    if( s.type == non-cacheable ){
5.      response = ServletContainer.execute( s,values);
6.      return response;
7.    }
8.    else if( s.type == cacheable ){
9.      scache= CacheManager.get(s.name, values)
10.     if(scache != null){
11.       CacheManager.updateFactors( scache);
12.       return scache.response;
  
```

```

13.     }
14.     else{
15.         response = ServletContainer.execute(s, val-
ues);
16.         scache= new scache(new(id), s, val-
ues,response);
17.         CacheManager.push(scache);
18.         while(Store.size > Store.maxsize ){
19.             Run cache replacement algorithm to evict the
scache with minimal values of rank(scache).
20.         }
21.         return scache.response;
22.     }
23. }
24. }

```

Algorithm 1. The Principle of Servlet Cache Model

4 Sequential Patterns in Servlet Container

4.1 Basic Definitions

Srikant [8] defines sequential patterns in database as “5% of customers bought ‘Foundation’ and ‘Ring world’ in one transaction, followed by ‘Second Foundation’ in a later transaction. In servlet container, we use similar description to define sequential patterns as “15% of customers visited servlet ‘searchBook’ followed by servlet ‘showBookDetail’ in a later visit”. Some basic definitions of sequential pattern are presented as follows.

Definition 4. Transition, if a *session* contains a sequence like $\dots id_m, \dots, id_{m+d} \dots$ and $id_m = s_{source}.id$, $id_{m+d} = s_{target}.id$, we think that there exists a transition from servlet *source* to servlet *target* and transition distance is d , which is defined as: $S_{source} \xrightarrow{d} S_{target}$, where S_{source} , S_{target} and d are source, target and distance of the transition, respectively.

A transition is called **self transition** if its source and target are the same servlet. For example, if a servlet s returns product information according to *productid*, a self transition happens when a customer visits s with different *productid* continuously.

Definition 5. Sequential pattern is defined as the probability that transition $S_{source} \xrightarrow{d} S_{target}$ happens in servlet container.

Sequential patterns in servlet container represent the business associations among servlets. We can use them to predict the probability that a cacheable servlet response is visited again and evict cache objects with minimal probability.

4.2 Pattern Graphs

Definition 6. Transition graph TG(d) is a directed weighted graph, which comes from a group of session *sessions*: session(1), session(2), \dots , session(n). Vertex i of

TG(d) represents servlet s_i , edge $e<i, j>$ represents transition $s_i \xrightarrow{d} s_j$ and weight of edges $W_{TG(d)}(i, j)$ represents the times that $s_i \xrightarrow{d} s_j$ happened in sessions.

Definition 7. $P(i, j, d)$ denotes the probability that transition $s_i \xrightarrow{d} s_j$ happened in a group of session sessions. Obviously $P(i, j, k) = W_{TG(k)}(i, j) / \sum_{m \in EndS} W_{TG(k)}(i, m)$, where $W_{TG(k)}(i, j)$ represents the weight of edge $e<i, j>$ of TG(k) and $EndS$ is collections contained all end of transition $s_i \xrightarrow{k} s_m$ in sessions. $P(i, j, d)$ is sequential pattern in servlet container according to definition 5.

Definition 8. Transition probability graph TPG(d) is also a directed weighted graph from a group of session sessions: session(1), session(2), , session(n). Its vertices and edges have the same meaning with TG(d), but the weight of edge $W_{TPG(d)}(i, j) = P(i, j, d)$.

TG(d) and TPG(d) are called pattern graphs (PG), they record customer visit behaviors from different viewpoints. The former records customer visit behaviors honestly and latter records the trends of customer visit behaviors. Before using them to design replacement cost functions, we first take an example to illustrate pattern graphs ($d=1, 2$) and compare them with Markov model.

Suppose there are 5 servlets and $S^C = \{s_1, s_2, s_3\}$, $S^{NC} = \{s_4, s_5\}$ in servlet container. The sessions of three customers are as follows:

- session (1) = { $s_1, s_2, s_3, s_5, s_5, s_2, s_3, s_4$ }
- session (2) = { $s_1, s_2, s_4, s_3, s_4, s_1, s_2, s_3, s_5, s_5$ }
- session (3) = { $s_2, s_3, s_5, s_1, s_2, s_4$ }

Obviously we can get $N=5, L=3$. We represent cacheable servlets by grey nodes and non-cacheable servlets by white nodes. Then, the corresponding pattern graphs can be illustrated in Fig.3 (a) ~ (d).

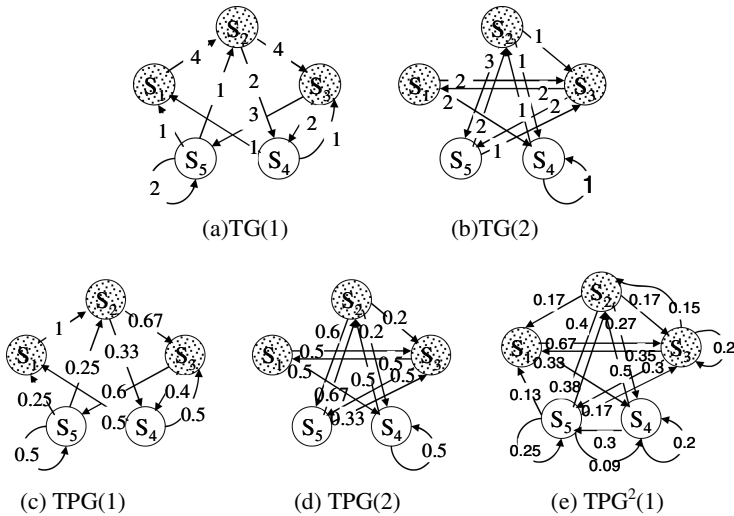


Fig. 3. Pattern Graphs

Current approaches mainly adopt first-order Markov model to predict the future customer visit behaviors according to visit histories [21][22]. First-order Markov model gets TPG (n) by calculating the power of TPG (1), namely $TPG(n) = TPG^n(1)$. However, these approaches have some defects comparing with ours. On the one hand, they base on first-order Markov assumption, in which those servlets that customer will visit in future are only determined by servlets that customer are visiting currently. This model can not accurately reflect customer visit behaviors because it will bring some inexistent multiple steps visit behaviors. For example, as shown in Fig.3(e), there exist some transitions such as $s_3 \xrightarrow{2} s_2$, $s_5 \xrightarrow{2} s_7$ and $s_5 \xrightarrow{2} s_4$ in $TPG^2(1)$, but they never happened in three customer sessions. The inexistent transitions make visit patterns from Markov model are imprecise and may lead to invalid predictions. Our approach obtains pattern graphs from original customer sessions, so it overcomes the inaccuracy of Markov model. In addition, computation of $TPG^n(1)$ will bring more time-complexity.

4.3 Sequential Patterns Discovery Algorithm

Based on above definitions, we present a discovery algorithm discoverTPG to discover pattern graphs. For convenience of discussion, we define operation: session [k] = id_k , which gets the identity of servlet at position k in a session.

Algorithm 2 describes our sequential patterns discovery algorithm. First we use the function discoverTG to compute all transition graphs TG(k) (line 5). Secondly we compute $\sum_{m \in Ends} W_{TG(k)}(i, m)$ for different distances (line 10). At last, we get all transition probability graphs with different distances (line 13).

```

1.   S: the set of all servlets in servlet container
2.   sessions: sessions represented clients behaviors
3.   d: maximal transition distances
4.   function discoverTPG(S, sessions, d){
5.       int[][][] TG = discoverTG(S, sessions, d);
6.       double[][][] TPG = new double[|S|][|S|][d];
7.       int[][] Temp = new int[|S|][d];
8.       for( int m = 0; m < d; m ++){
9.           for( int j = 0; j < |S|; j ++ )
10.              Temp[i][m] += TG [i][j][m];
11.           for( int j = 0; j < |S|; j ++ ) {
12.               for( int i = 0; i < |S|; i ++ ){
13.                   TPG [i][j][m] = TG [i][j][m]/ Temp[i][m];
14.               }
15.           }
16.       }
17.       return TPG;
18.   }
19.   function discoverTG(S, sessions, d){
20.       int[][][] TG = new int[|S|][|S|][d];
21.       for( each session ){
22.           for( int i = 0; i < d; i ++ ){
23.               for( int j = 0; j < session.length -d; j ++){
24.                   TG [session[j]][session[i + d]][d]=1;
25.               }

```

```

26.     }
27.   }
28.   return TG;
29. }

```

Algorithm 2. discoverTPG Algorithm

If there are N sessions and the average length of them is $avglength$, the time-complexity of $O(N*d*avglength)$ is required for the function `discoverTG`. On the other hand, the function `discoverTPG` needs $O(|S|*|S|*d)$ time-complexity to compute all transition probability graphs. Therefore this discovery algorithm has a time-complexity of $O(max(N*d*avglength, |S|*|S|*d))$. While there are a large number of sessions and servlets in servlet container, this algorithm is time-consuming and may occupy a lots of system resources. Therefore, we should choose suitable time to run discovery algorithm to avoid the influence on the normal running of servlet container.

In addition, Berkhin [17] has confirmed that there are no associations between two servlets if transition distance is higher than 6. Therefore, we only concern the transitions with less 7 distance.

5 Replacement Cost Functions

In this section, we first briefly review some traditional replacement cost functions. Secondly the concept of predictive probability function will be presented. At last two replacement cost functions based predictive probability function are introduced.

5.1 Traditional Replacement Cost Functions

Traditional replacement cost functions concern some factors such as visit time, visit frequency, file size and file loading cost in web server and proxy server. The same factors can also be concerned in servlet container, but they have different meanings for servlet cache object. Some factors of scache are as follows:

- age: the last visit time of scache.
- frequency: the visit times of scache.
- size: the size of scache.response
- cost: the time spending on executing scache.s to generate scache.response with scache.values.

Definition 9. Above factors are called as **original factors** because they belong to servlet cache object (scache) itself. Some typical replacement cost functions of traditional cache replacement algorithms are listed in Table.1, which contain one or more original factors.

In Table.1, the factor age and frequency make the cache objects with minimal revisited possibility evicted from the cache; the factor size and cost make the cache objects with minimal replacement cost removed from the cache. These replacement cost functions only consider visit history of single servlet regardless of associations among

Table 1. The Factors of Cache Replacement Algorithms

	age	frequency	size	cost	replacement cost function
LRU	√				age
SIZE			√		size
LFU		√			frequency
GDSF	√	√	√	√	cost*frequency/size +age

servlets. Next, we will consider some features of servlet container and utilize aforementioned transition probability graph TPG(d) to design predictive probability function, which makes a supplement for traditional replacement cost functions.

5.2 Predictive Probability Function

Definition 10. Predictive probability function represents the probability that a cacheable servlet is revisited in near future, which is regarded as a supplementary factor for original factors and comes from associations among servlets. In this section we design two predictive probability functions as follows, where K represents quantity of scache objects in cache system.

- Session-based predictive probability function

$$rank_{session}(j, D) = \sum_{d=1}^D \sum_{i=1}^{|sessions|} TPG(d)[session[length], j, d] \tag{1}$$

- Scache-based predictive probability function

$$rank_{scache}(j, D) = \sum_{d=1}^D \sum_{i=1}^K TPG(d)[scache(i).s.id, j, d] \tag{2}$$

Equations (1) and (2) predict the probability that a cacheable servlet s_j is revisited in near future according to the different historic information. The function $rank_{session}(j, D)$ bases on session information and reflects realtime customer behaviors. The function $rank_{scache}(j, D)$ only considers the transitions starting from cacheable servlets and reflects the status of cache system, which maybe makes its prediction lesss accurate than equation (1), but less time-complexity of computation will be gained.

It needs the time-complexity $O(L \times D \times |sessions|)$, $O(L \times D \times K)$ to compute the prediction probability of all cacheable servlets through equations (1) and (2), respectively. Although predictive probability functions induce some computation cost, they can evict cache objects with minimal cost and reduce the times of cache replacement and redundant reexecution of cacheable servlets. In result, a higher cache hit ratio and better performance of servlet container are achieved by adoption of predictive probability functions.

If we define traditional cost functions and predictive probability functions as $rank_{traditional}(scache)$ and $rank_{predictive}(scache)$ respectively, we can represent new replacement cost function as:

$$rank_{new}(scache) = rank_{traditional}(scache) \times rank_{predictive}(scache). \tag{3}$$

In the section 6 we will confirm the effectiveness of new replacement cost function $\text{rank}_{\text{new}}(\text{scache})$ through evaluations.

6 Evaluation

The servlet cache model of servlet container and replacement cost functions have been implemented in the application server OnceAS [18] developed by Institute of Software, Chinese Academy of Sciences.

OnceAS application server runs on a PC with CPU of P4 2.8G, memory of 512M, and windows 2000 professional operating system. The simulating customers visit OnceAS through a 100M LAN. We adopt a Java EE blueprint program Pet Store Demo [19] as our web application. For experiment simulation, we add 10000 pets including 2000 cats, dogs, fish, birds and reptiles respectively into pet store as our simulation data.

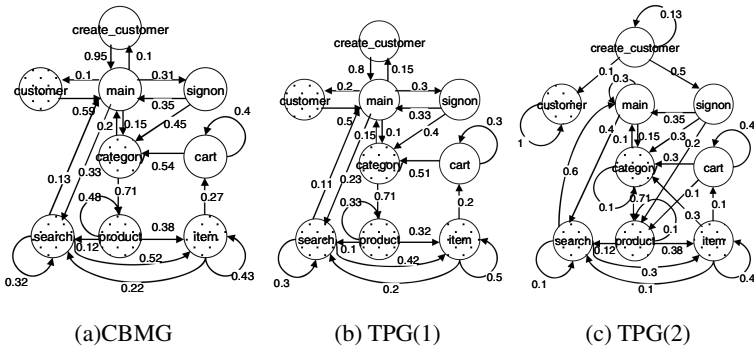


Fig. 4. CBMG of Simulating Customers and Pattern Graphs

Daniel [20] proposes customer behavior model graph (CBMG) and gives different category CBMG for three kinds of customers to simulate the behaviors of real customers in TPC-W. We use Daniel's method to compute the CBMG of Pet Store for one category customer in Fig.4 (a) where white nodes denote non-cacheable servlets and grey nodes denote cacheable servlets. We regard Fig.4 (a) as the behaviors of simulated customers, every simulated customer visits different servlets continually according to the transition probability of Fig.4 (a).

Servlet cache system running in OnceAS has provided the solution on cache consistency, which will evict invalid cache objects in time. In addition, since the response of servlets search, categories, products and items are relatively stable unless

Table 2. The Cacheable Servlets of Pet Store

Servlet name	category	product	item	customer	search
Parameter	category_id	product_id	item_id	customer_id	keywords

we maintain the database of Pet Store, we define them as cacheable servlets. Their name and parameters are shown in Table.2.

The evaluation has been performed in two steps. First of all, 100 customers are simulated to visit Pet Store for six hours according to the CBMG in Fig.4 (a) so that we can collect the customer sessions and run discovery algorithm discoverTPG to achieve pattern graphs. In a second stage, we utilize the results from first stage to perform some experiments to confirm the efficiency of our approach.

The results of first stage are presented in Fig.4 (b)(c). Obviously, TPG(1) is similar to Fig.4 (a), which has covered customer visit tendency. In addition, we can see that TPG(2) can discover some sequential patterns that can not be discovered by TPG(1), such as $customer \xrightarrow{2} customer$ and $item \xrightarrow{2} category$.

Experiment 1 is conducted with different cache sizes and replacement cost functions. We still simulate 100 customers with the behaviors described in Fig.4 (a) to visit Pet Store for two hours. The settings and results of experiment 1 are presented in Table 3 and Fig.5. (The function whose $rank_{predictive} = 1$ represents a traditional algorithm, and $mrank_{session}(j,1)$ and $mrank_{session}(j,2)$ represent the one step and two steps replacement cost functions from Markov model, respectively.) First, experiment 1 compares cache hit ratios of traditional LRU algorithm and sequential pattern-based cache replacement algorithms adopted equations (1) and (2), respectively. Secondly experiment 1 shows the influence on cache hit ratio from different transition distances. At last, experiment 1 also compares the prediction effect of our approach with Markov model at different distance.

Table 3. The Settings of Experiment 1

Cost function	$rank_{traditional}$	$rank_{predictive}$
Func1	age	1
Func2	age	$rank_{scache}(j,1)$
Func3	age	$rank_{session}(j,1)$
Func4	age	$mrank_{session}(j,1)$
Func5	age	$rank_{session}(j,2)$
Func6	age	$mrank_{session}(j,2)$

From Fig.5 (a) we can see that traditional LRU algorithm (Func1) has the worst cache hit ratio, only 52% when cache size is 1.2MB. Then, cache hit ratio has increased remarkably when predictive probability functions are applied. In addition, cache hit ratio increment gained from equation (1) is higher than that from equation (2) (Func3>Func2) just like what we expect in section 5. Although there exist several exceptions caused by additional computation, the cache hit ratio of Func5 is higher than that of Func3 at most time. It shows that the longer distance is considered in replacement cost function, the higher cache hit ration can be gained. At last, we can see that Markov model (Func4) has the almost same increment comparing to our approach (Func3) when transition distance is 1. However, our approach has the more attractive enhancement of cache hit ratio than Markov model when transition distance is 2, because the latter introduces some inexistent visit behaviors at this time.

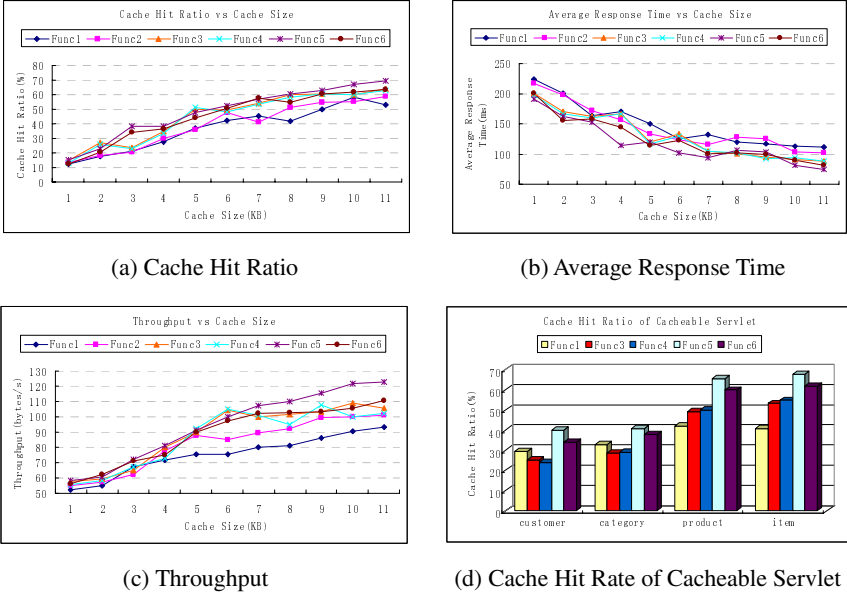


Fig. 5. The Result of Experiment 1

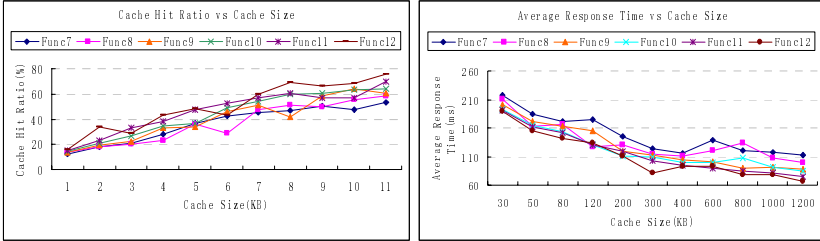
Fig.5(b) and (c) show that our approach also gains much less average response time and higher throughput of servlet container than traditional algorithms and those based on Markov mode. However, we notice that there exists an abnormal case where the cache hit ratio of Func3 is lower than that of Func1 (traditional LRU) for servlets *customer* and *category*, as shown in Fig.5 (d). The reason is that TPG(1) in Fig.4 (b) misses some customer behaviors, which makes cached responses of servlets *customer* and *category* are evicted improperly and impairs cache hit ratio. But we are glad to see that Func5 (TPG(2)) has erased these abnormalities successfully.

Experiment 2 is also conducted with different cache sizes, which compares cache hit ratio of original LRU, LFU and GDSF algorithms and the corresponding algorithms using predictive probability function $\text{rank}_{\text{session}}(j,2)$ respectively. Table.4 and Fig.6 show the settings and the result of experiment 2.

Table 4. The Cost Functions of Experiment 2

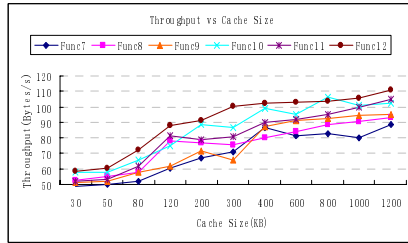
Cost function	$\text{rank}_{\text{traditional}}$	$\text{rank}_{\text{predictive}}$
Func7	age	1
Func8	age	$\text{rank}_{\text{session}}(j,2)$
Func9	frequency	1
Func10	frequency	$\text{rank}_{\text{session}}(j,2)$
Func11	$\text{cost} * \text{frequency} / \text{size} + \text{age}$	1
Func12	$\text{cost} * \text{frequency} / \text{size} + \text{age}$	$\text{rank}_{\text{session}}(j,2)$

From Fig.6(a) we can observe that cache hit ratios of three cache replacement algorithms have been improved after adopting predictive probability function ($\text{rank}_{\text{session}}(j,2)$). Although there are some inconsistencies in Fig.6 (a) because the computation of $\text{rank}_{\text{session}}(j,2)$ consumes some system resources such as CPU and memory, Fig.6 (b) and (c) still show that predictive probability functions have improved the performance of servlet container in response time and throughput.



(a) Cache Hit Ratio

(b) Average Response Time



(c) Throughput

Fig. 6. The Result of Experiment 2

7 Conclusion

To resolve the servlet cache replacement problem and improve the performance of servlet container, in this paper, we first present pattern graphs to describe business associations among servlets as sequential patterns. Secondly we present our discovery algorithm to discover the pattern graphs and propose two predictive probability functions to mend traditional replacement cost functions according to pattern graphs. Finally, evaluation shows that our approach can effectively improve cache hit ratio and the performance of servlet container.

However, our approach still has some limitations. Firstly, the discovery algorithm discoverTPG has a higher time-complexity ($O(\max(N \times d \times \text{avlength}, |S| \times |S| \times d))$), which makes us have to choose suitable time to run discovery algorithm. Secondly pattern graphs maybe become stale with the change of customer visit behaviors so that we have to rerun discovery algorithm to capture these changes and adjust our predictive probability functions appropriately. In addition, to apply this approach into other environments such as EJB and Web service, it is need to be extended to solve the consistency of cached objects. We will solve above issues in the future.

Acknowledgments. The work was supported partially by the National Natural Science Foundation of China under Grant No. 60573126; the National Basic Research Program of China (973) under Grant No. 2002CB312005; the National High-Tech R&D Plan of China (863) under Grant No.2006AA01Z19B; the National Key Technology R&D Program of China under Grant No. 2006BAH02A01.

References

1. Coward, D.: Java™ Servlet Specification Version 2.4 Sun Microsystems Inc. 2003-11-24 <http://jcp.org/aboutJava/community/process/final/jsr154/index.html>
2. Christopher, L.: Servlet Performance Report: Comparing The Performance of J2EE Servers <http://www.webperformanceinc.com/library/reports/ServletReport/index.html>
3. Bonchi, F., Giannotti, F., Gozzi, C., Manco, G., Nanni, M., Pedreschi, D., Renso, C., Ruggieri, S.: Web Log Data Warehousing and Mining for Intelligent Web Caching. *Data & Knowledge Engineer* 39(2), 165–189 (2001)
4. Li, K., Shen, H., Tajima, K.: Cache Design for Transcoding Proxy Caching. In: Jin, H., Gao, G.R., Xu, Z., Chen, H. (eds.) *NPC 2004*. LNCS, vol. 3222, pp. 187–194. Springer, Heidelberg (2004)
5. Turner, D.: Web Page Caching in Java Web Applications. In: *Proc. of International Conference on Information Technology Coding and Computing, Las Vegas (2005)*
6. Shupp, R., Andy, C., Chuck, F.: Web Sphere Dynamic Cache: Improving J2EE application performance. *IBM System Journal* 43(2), 351–370 (2004)
7. Podlipnig, S., Böszörményi, L.: A survey of web cache replacement strategies. *ACM Computing Surveys* 35(4), 374–398 (2003)
8. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: *Proc. of the Fifth Int. Conference on Extending Database Technology, Avignon, France: Springer-Verlag Berlin and Heidelberg GmbH*, pp.18–32 (1996)
9. Cooley, R., Mobasher, B., Srivastava, J.: Web mining: information and pattern discovery on the World Wide Web. In: *Proc. of the 9th IEEE International Conference on Tools with Artificial Intelligence, Newport Beach*, pp. 558–567. IEEE Computer Society, Los Alamitos, CA, USA (1997)
10. Garofalakis, M., Rastogi, R., Shim, K.: Spirit, Sequential pattern mining with regular expression constraints. In: *Proc. of the ICVL D*, pp. 223–234. Morgan Kaufmann Publishers, Edinburgh (1999)
11. Han, J., Pei, J., Mortazavi-Asl, B.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: *ICDE01*, pp. 215–224. Springer, Heidelberg (2001)
12. Cherkasova, L.: Improving WWW Proxy Performance with Greedy-Dual-Size Frequency Caching Policy, HP Labs: Computer Systems Laboratory: HPL-98-69R1 (1998)
13. Abrams, M., Stanbridge, C., Abdulla, G., Williams, S.: Caching Proxies: Limitation and Potentials. In: *Proc. of the 4th WWW Conference*, pp. 119–133. O’Reilly, Boston (1995)
14. Yang, Q., Zhang, H.: Web-log mining for predictive web caching. *IEEE Transactions on Knowledge and Data Engineering* 15(4), 1050–1054 (2003)
15. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Yu, P.S., Chen, A.S.P. (eds.) *ICDE*. In: *Proc. of the 11th ICDE*, pp. 3–14. IEEE Computer Society Press, Washington DC (1995)
16. Yin-Fu, H., hao Min, J.: Mining Web Logs to Improve Hit Ratios of Prefetching and Caching. In: *Proc. of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, Compiegne France, Sep. 19-22*, pp. 577–580 (2005)

17. Pavel, B., Becher, D.J., Randall, D.J.: Interactive Path Analysis of Web Site Traffic. In: Proc. of ACM SIGKDD Int. KDD01, San Francisco, CA, pp. 419–441. ACM Press, New York (2001)
18. Huang, T., Chen, N.J., Wei, J., Zhang, W.B., Zhang, Y.: OnceAS/Q: A QoS-enabled Web application server. *Journal of Software* 15(12), 1787–1799 (2004)
19. JavaTM Pet Store Demo, <http://java.sun.com/developer/releases/petstore/>
20. Daniel Menascé, TPC-W: A Benchmark for E-commerce *IEEE Internet Computing*, 6(3), pp. 83–87 (2002)
21. Sarukkai, R.: Link Prediction and Path Analysis Using Markov Chains. In: Proc. of the 9th Intl. World Wide Web Conf. Amsterdam, May (2000)
22. Zhu, J., Hong, J., Hughes, J.: Using Markov Chains for Link Prediction in Adaptive Web Sites. In: Proc. of Software, Belfast, Northern Ireland, pp. 60–73 (2002)
23. Web Characterization Activity. <http://www.w3.org/WCA>

A Hybrid Cache and Prefetch Mechanism for Scientific Literature Search Engines

Huajing Li¹, Wang-Chien Lee¹, Anand Sivasubramaniam¹, and C. Lee Giles^{1,2}

¹ Department of Computer Science and Engineering

² The School of Information Sciences and Technology

Pennsylvania State University

State College, PA 16802, USA

{huali, wlee, anand}@cse.psu.edu, giles@ist.psu.edu

Abstract. *CiteSeer*, a scientific literature search engine that focuses on documents in the computer science and information science domains, suffers from scalability issue on the number of requests and the size of indexed documents, which increased dramatically over the years. *CiteSeer^x* is an effort to re-architect the search engine. In this paper, we present our initial design of a framework for caching query results, indices, and documents. This design is based on analysis of logged workload in *CiteSeer*. Our experiments based on mock client requests that simulate actual user behaviors confirm that our approach works well in enhancing system performances.

1 Introduction

Due to the rapid growth of the Web in recent years, Internet has changed our everyday life style dramatically. Various types of web applications have come into shape and begun to service the public, among which maybe the most successful story is the popularity of search engines (i.e. Google¹, AltaVista²). As one important branch of search engine applications, scientific literature search engines are welcomed because of their dedication in a specific domain, which in result improve the quality of services. Unlike generic search engines such as Google, scientific literature search engines limit their scope in a set of tightly-correlated topics. *CiteSeer* [2,6,7] is a web-based scientific literature search engine which focuses on computer and information science. On average, *CiteSeer* receives over 800,000 requests daily, is accessed by over 100,000 unique users monthly, and serves approximately 30 gigabytes of data daily. As the document corpus grows larger and the workload increases, it is observed that *CiteSeer* suffers from its original design deficiencies, causing long service latencies and larger maintenance costs.

A perceivable performance metric for end-users of search engines is the query response time. Namely, the time it takes for a user to wait before an issued query to be answered and returned. Inside a search engine, queries are sent to the server, which accesses the indices (in the form of *inverted lists*) to obtain the result set. In addition to the query performance, *CiteSeer* has another concern as well: the document retrieval

¹ <http://www.google.com>

² <http://www.altavista.com>

efficiency, which represents the time it takes for the system to generate the detail page of a document and present it to the user. As we click on a specific item in a search result, CiteSeer will provide a summary page describing the document, including title, author, abstract, related documents, bibtex, etc. The required data are located in distributed sources, some of which have to be generated by internal algorithms on-the-fly. Considering the huge volume of requests for documents, it is critical to improve the document retrieval efficiency.

Caching techniques have been used to address performance issue of information systems, including search engines. Two levels of caches exist for current search engine applications. *Result cache* [9][13] stores previous search results, while *index cache* [4], on the other hand, stores inverted lists for query terms that have been used. In addition, as an alternative to existing cache replacement policies, *prefetching* can improve the system performance by fetching result list beforehand.

To better understand the characteristics of CiteSeer, usage logs are analyzed to gain insights into typical user behaviors. In our analysis, we found high shareness and locality in user requests, with popular requests repeating frequently. Existing works mainly apply a cache management policy to the system. With detailed analysis into the request distribution characteristics, we believe such one-policy scheme lacks the flexibility to reflect the distribution features. Hence, we propose a hybrid cache management scheme, whose performance can be optimized by tuning parameters according to the summarized request distributions. In addition, our workload analysis reveals that high correlation exists for requests to CiteSeer, which motivates us to incorporate a correlation-based prefetching mechanism into the framework. This unique function is not available in existing caching mechanisms for search engines. Also, we include document caching in improving document retrieval latency.

Previously proposed caching approaches do not fully exploit the potential of the unique running characteristics exhibited by CiteSeer logs to optimize the system performance. In this paper, we propose and analyze a hybrid cache and prefetch framework for CiteSeer^x. Our contributions can be summarized as follows:

- We propose an integrated hybrid caching framework, which is based on our analysis to the the logged CiteSeer trace. The framework is comprised of multiple components, each of which utilizes a combined cache management policy and is dedicated to improve the latencies for a specific group of requests.
- We include the term and document correlation into consideration in predicting future user requests and thus provide prefetch facility before a user actually requests documents.
- We experimentally evaluate the proposed hybrid cache to test its effect based on CiteSeer’s actual data and workload. Also, we demonstrate the tuning of cache’s running parameters to achieve optimized performance.

The rest of the paper is organized as follows. Our analysis to CiteSeer’s usage logs is presented in Section 2. We propose our hybrid caching framework for CiteSeer^x in Section 3. Section 4 gives our evaluation results and discussions. In Section 5, we briefly review and compare proposed caching mechanisms in the literature. Finally, Section 6 provides the concluding remarks.

2 Workload Analysis

In order to understand the working nature of CiteSeer, usage logs are parsed and analyzed so that we can have insights into typical user behaviors and evaluate the effectiveness of a system cache. This section summarizes our workload analysis results. Basically, Section 2.1 introduces the preliminary tasks we performed to the usage logs. Section 2.2 gives a statistical summary to basic attributes of the logs. Afterwards, Section 2.3 and Section 2.4 respectively give the request frequency distribution analysis and locality analysis. Next, we studied the correlation between user requests, which is presented in Section 2.5. Finally, we give the analysis summary.

2.1 Data Preparation

A typical CiteSeer logging entry consists of five parts: time stamp, request type, parameter (query terms, requested document identifiers, etc), IP address, and agent type. An example is given below:

```
1114070813.127 event context 0 1782747 0 446930 ip: 128.255.54.* agent:
Generic
```

Although the log format contains all the information needed for the research, some data preparation tasks are necessary. First, irrelevant request types of the study are filtered out. Only three request types are kept, which are (1) document query, (2) document detail, and (3) document download. The latter two request types are treated as the same type in the analysis because they are both requests for document records. The second task we performed is to *sessionize* the logged trace. In CiteSeer, the system does not record user identities as well as session information. We use the following heuristics to differentiate sessions: (1) The IP address is used as the user identity. (2) A time threshold is used to measure the request interval from a same IP address. In our experiment, we set the time threshold to be 1,800 seconds (30 minutes). Finally, it is found the robot requests form a considerable portion of the network traffic. To better understand a user's behavior, robot requests are removed. Basically, we used a robot agent string list⁴ to identify obvious robots. To those robots that do not declare themselves, statistical analysis is performed to identify them. If we find that either the average session length is extremely long (more than 500) or a portion (10%) of intra-session intervals are particularly small (less than 0.5 second), the IP address is tagged with "robots" because they exhibit odd behaviors from normal human users.

2.2 Statistical Summary

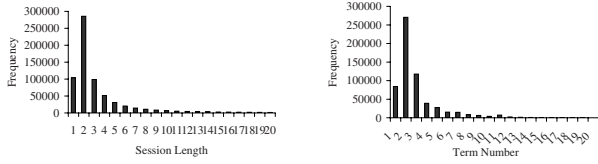
Two-week-length CiteSeer logs are collected and analyzed in this paper. Table 1 gives the statistical summary of the usage logs. From the number of requests and the number of the unique requests we can infer that many requests do not repeat frequently (also shown in Section 2.3). Also, we can see that the for most sessions, the session length is limited in a relatively small value. The detailed session length distribution can be found in Figure 1(a), from which we can observe that most sessions have a length of 2 or 3.

³ We suppressed the last section of the IP address.

⁴ <http://www.pgtsj.com.au/pgtsj/pgtsj0208d.html>

Table 1. Statistical summary of the analyzed logs

Summary	Trace
Number of Document Query Requests	601, 337
Number of Document Retrieval Requests	2, 273, 233
Number of Distinct Document Query Requests	334, 731
Number of Distinct Document Retrieval Requests	393, 416
Average Session Length	2.839



(a) Session length distribution (b) Term number distribution for document queries

Fig. 1. Statistical analysis

We also performed an analysis on the distribution of term numbers in document queries. From Figure 1(b), it can be observed that most queries only contain a small number of terms (less than 10). This finding implies that there is a high possibility that some unique terms are frequently queried.

2.3 Frequency Distribution

As we look through the logs, we find that there exist a set of *hot* query terms and documents in the system. The requests to these terms and documents are very frequent. Correspondingly, the large portion of the requested term corpus and document corpus only contributes a small portion of the entire requests. We rank terms and documents according to their request frequencies and plot the distributions in Figure 2, where Y-axis shows the log-scaled request frequency, while X-axis shows the log-scaled ranking of the request, sorted by their frequencies. The distributions shown in Figure 2 are close to straight lines, suggesting the existence of the Zipf distribution. This finding shows that system cache can improve the service efficiency, considering some query terms and documents are highly popular. If results are stored in a high-speed storage device or in main-memory, a large portion of requests can be answered without accessing the indices or the data store.

The above study does not tokenize query terms. However, many queries have some popular terms in common. Remember for most search engines the query service will tokenize the query string into separate terms and send them to the indices. In addition, full-text indices are arranged as inverted lists corresponding to individual terms. Therefore, inverted lists of popular single-terms can be cached as long as we find similar Zipf distribution for single term frequencies. We examine the logs and plot the finding in Figure 2(c), from which we can see the existence of an approximate straight line, confirming our expectations.

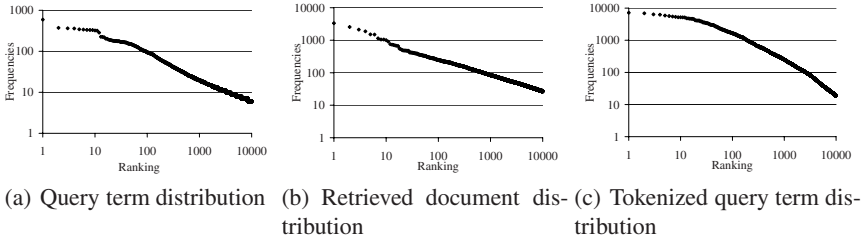


Fig. 2. Log-log scale request distribution

2.4 Request Locality

Previous studies indicate that there is a high shareness in user requests [10,13,16]. However, the effectiveness of a system cache will be greatly decreased without temporal locality in highly repetitive requests. To answer this question, we analyze the logs to find request localities.

Firstly, we try to reveal the *distance* between subsequent resubmission of same requests, where distance represents the number of other requests in the interval of a request resubmission. Figure 3(a) and 3(b) shows the distance distribution (in log-log scale) for document query and retrieval respectively. From the plots we can see that most repeated requests have a relatively small distance. In other words, the same requests are often issued with small intervals. This statistics suggest that if we cache temporary query results or documents, it has a high possibility that the cached data will be requested in a short period of time. With a good cache replacement policy deployed, high cache hit rate can be expected.

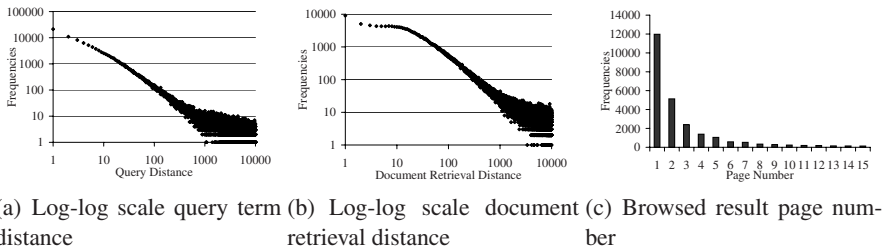


Fig. 3. Request locality distributions

We also study user behaviors in browsing paged query results. Previous web studies [3] reveal that users are reluctant to browse pages other than the first one. We conduct the analysis to find the probability for a user to view following pages. The distribution of requested page number is plotted in Figure 3(c), from which we can see most users are only interested in the top 3 pages, which contributes to 75.33% browse requests. Hence, if we consider prefetching result pages, the scope should not exceed the first 3 ones.

2.5 Correlation Studies

If the system can capture a user’s interest on-the-fly from previous operations and retrieve relevant documents beforehand, the user can acquire the documents instantaneously as he clicks the document link. Correlation studies can help us in predicting the possible relevant documents in the repository. For example, if we find users have a high probability to view or download d_i after querying Q , d_i probably is a good candidate document for the user. We study the correlation between issued queries and the documents that are browsed afterwards. We give the correlation analysis of all 334, 731 distinct logged query strings in Table 2.

Table 2. Correlation for query-document (Q - D) pairs

Correlation	Q - D Pairs	Unpopular Q Percentage
1%	221, 981	79.13%
5%	41, 718	86.61%
10%	18, 903	90.72%
50%	2, 381	95.38%

As we look into the queries that have high correlation with documents, we find most queries (90.72% for 10% correlational probability or higher, for instance) are unpopular ones, whose cumulative query frequency does not exceed 5 in the logged time interval. This result suggests that if a correlation-based prefetch is supported, it can work as a good supplementary method for ordinary cache mechanisms, which basically serve popular queries.

Correspondingly, we also analyze the correlation between document retrieval requests, which would reveal the relationship between documents. The same procedure is applied to all 393, 416 distinct document retrieval requests. The results are given in Table 3.

Table 3. Correlation for document-document (D_1 - D_2) pairs

Correlation	D_1 - D_2 Pairs	Unpopular D_1 Percentage
1%	85, 695	81.68%
5%	6, 875	84.41%
10%	2, 634	86.82%
50%	298	93.10%

Deeper analysis into the contents of involved papers shows that many highly correlated papers have inherent relationships, including the same author papers, cited-citing pairs, and semantically similar papers. For the limitation of pages, detailed analysis is not included.

2.6 Summary

In this section, an analysis has been performed on the logged user requests. In summary, we find the existence of Zipf distributions in user query terms and retrieved documents. A small portion of hot requests contributes a large portion of the entire traffic. The follow-up locality study shows that for these hot requests, usually they have a high temporal locality. Hence, the results demand a well-tuned cache for CiteSeer to improve the system's performance. Single-term study also suggests caching popular inverted lists. Supplementarily, our correlation study suggests that correlation-based prefetch can determine in advance what documents to be retrieved and reduce the response time of unpopular requests.

3 Cache Framework

Based on the conclusion in Section 2, we derive our design for the cache framework of CiteSeer^x. In this section, we first discuss what unit should be cached in Section 3.1. For each unit type, the cache is segmented and appropriate replacement policies are employed (Section 3.2). Hybrid cache framework architecture is given in Section 3.3. Finally, we discuss the implementation issues for CiteSeer^x.

3.1 Caching Unit

The first question that needs to be answered in developing a cache scheme is: what need to be cached? Look at the generic service process for CiteSeer to fulfill a user request. Basically, multiple modules and data sources in the system are involved. Based on request types, we can cache contents in multiple levels. First, our single-term analysis indicates it would be great if we can store indices in high-speed storage devices or main-memory. However, due to the cost concerns and the huge size of indices, it is unrealistic to store all indices in the cache. However, it is easy to cut out a small portion of the index (inverted lists) so that it can fit the cache size. For example, the well-known *Lucene*⁵ library supports such operations and provides in-memory index facility. We call this type of cache as **Index Cache**.

Another possibility to improve document query performance is to cache result list pages, which we call **Result Cache**. The unit for this sort of cache is the already-generated result page. Although the result cache serves the same goal as the index cache, it has its own pros and cons. The advantage is that the result cache can save more storage spaces. Also, as we find a result cache hit, the page can be returned instantaneously without looking up the index. However, the result cache can only answer precisely identical queries. One entry in the index cache can be shared by multiple queries.

The third level of cache (**Document Cache**) is to help improve document retrieval efficiency. Here, the document detail page is wrapped with other auxiliary data (PDF paper, bibtex file, etc) and stored in the cache to avoid invoking related algorithms and unnecessary I/O requests.

⁵ <http://lucene.apache.org>

3.2 Cache Replacement Policy

Previous search engine cache designs generally are tailored to improve service efficiency for popular requests, with only one cache management policy for all records. In Section 2.3, we observed the Zipf-like distributions of queried terms and retrieved documents. Roughly speaking, the Zipf-like distribution has a head for most popular (hot) requests, with a long tail (unpopular requests). The body of the distribution includes some moderately-popular requests. Observing this, we can cut the distribution into three sections with two frequency threshold parameters t_1 and t_2 , where t_1 separates the head section, while t_2 separates the tail. The value of the parameters are dependent upon the distribution and the cache setting, which will be discussed in Section 3.4.

Because request frequencies are dramatically different in each distribution segment, it is helpful to treat them differently, employing separate cache management policies for them. This approach can improve the flexibility in managing the cache so that the system performance can be tuned. For the hot requests, we use a *static cache* policy, with the cached items always resident in the cache. For the moderately-popular requests, we implement a *dynamic cache* replacement policy, in which some well-known cache replace algorithm (LRU, SLRU, etc [5]) is used. For unpopular requests, we do not reserve cache space for them. However, a prefetch buffer is included in the cache to store prefetched data based on correlation statistics. As we have seen earlier, the prefetch is very effective for unpopular requests. The one-policy approach can be viewed as an extreme case of the hybrid cache design.

For the three components of caches, three previously observed Zipf-like distributions are used. To be specific, the tokenized query term distribution is used for the index cache, the document query term distribution for the result cache, and the document retrieval identifier distribution for the document cache. Each distribution has its own frequency threshold parameters, which are represented as t_{i_1} and t_{i_2} , where $i \in \{index, result, document\}$.

Based on the above discussions, we have the conceptual view of the hybrid caching framework for CiteSeer^x, as shown in Figure 4.

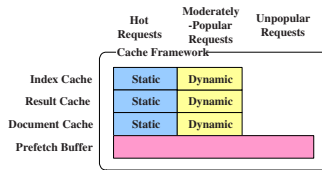


Fig. 4. Conceptual view of the hybrid cache framework

3.3 Architectural Design

As an important part of the query engine in CiteSeer^x, the cache framework captures incoming requests before they access the data repository. The whole process is briefly illustrated in Figure 5. When receiving a request, the request classifier determines the request type based on the encoded URL. Afterwards, the cache lookup module checks

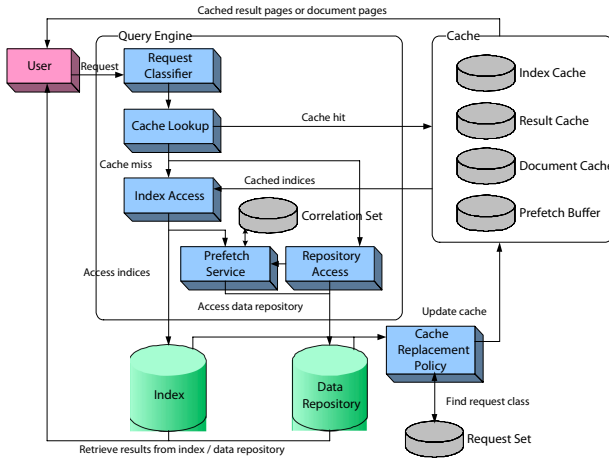


Fig. 5. The workflow in answering a user request

the cache for the results of the request. If a cache hit is found, either the result page can be returned to the user instantly or the cache contains some index entries for the query request. If it happens to be the latter case, partial resolved results are sent to the index access module for further processing. If there is a cache miss, the query engine directs the request to the index access module or the data repository access module. Prefetch can be triggered by previous requests, which access a built-in correlation set to find relevant documents, and store them in the prefetch buffer. The prefetched documents are removed from the buffer as long as the current user session terminates or another query is issued by the same user. Also, our analysis confirms that after issuing a query to CiteSeer, a user has a high possibility to access the second and third result list page. The two pages, if available, are prefetched when the user is browsing the first result list page until the session ends or the arrival of a new query. After fulfilling a user's request by index or data repository, the cache replacement policy works to check if the cache needs to be updated. A request set, resident in a high-speed local device, classifies the request into a popularity class (popular, moderately-popular, unpopular) and thus directs the request to an appropriate cache. The victim cached data entry is removed.

3.4 Implementation Issues

Suppose the available cache size is S in total, an important decision to be made during implementing the hybrid cache is to determine how to divide S for each cache component. Actually, the number of entries in the prefetch buffer is the same as the number of active users in the system, because each user has a collection of prefetched documents based on previous requests. The active user number is normally no more than 50 in CiteSeer. Hence, its size is negligible to the entire cache. We use a set of parameters, $f_x (x \in \{index, result, document\})$, to denote the size percentage of S reserved for each cache component. The value of f_x is within the range $[0, 1]$ and the sum of all f_x equals to 1.

The optimized values of f_x are influenced by request distributions. Remember in Section 3.2 two thresholds are used to separate each request distribution into three parts. Actually, parameter t_1 decides the number of popular requests for each frequency distribution. As a result, the static cache size in each component is heavily depended on t_1 . In the experimental settings, we only restrict the size reserved for each cache component. Parameter t_1 determines how many units should be cached in each static cache, and thus decides the actual static cache size. The remainder storage is used for the dynamic portion.

CiteSeer is an autonomous system that keeps obtaining new documents from Internet. System workload also varies over time. As the statistical analysis grows out-of-date, the effectiveness of the cache will be decreased. It is necessary to update the statistics in the system over a period of time. The update process mainly includes summarizing user request, updating request set, re-calculating request correlation, updating correlation set, and tuning cache sizes. These tasks can be performed offline in another server. Updated data can be switched to the public server during system maintenance period.

4 Performance Evaluation

4.1 System Setup

We develop a test query engine, which includes the proposed cache framework, for the evaluation. Actual CiteSeer data collection and real logged usage traces are used in the experimental platform. CiteSeer’s internal algorithms are also included to dynamically generate output results. Lucene is used to index documents. This testing platform needs to communicate with CiteSeer for data and services, which increases the response time. This extra overhead remains almost constant regardless the workload and query processing strategy. The dataset is stored in a SUN StorEdge 3511 FC array with SATA hard drives. We implement the cache on a workstation (CPU: 1 AMD Opteron Processor 252, 4GB Main Memory, OS: Red Hat). Index cache is an in-memory cache. Result cache and document cache are stored in local high-speed disks. We use the popular

Table 4. Default experiment implementation parameters

System parameter	Default value
Cache size (S)	2GB
Prefetch buffer size	50MB
Correlation threshold	0.1
Cache component	Size portion
Index cache	50%
Result cache	12.5%
Document cache	37.5%
Distribution	Frequency threshold
Tokenized term	$t_{index_1} = 1000, t_{index_2} = 10000$
Query term	$t_{result_1} = 300, t_{result_2} = 3000$
Retrieved document	$t_{document_1} = 1000, t_{document_2} = 10000$

cache replacement algorithm, *LRU*, as an example to manage the dynamic caches. The default parameter values we set for our experiments are given in Table 4.

Two workloads are applied in our experiments. The first workload (*actual trace*) uses another piece of system usage log (38,590 queries and 212,269 document retrievals) to mock user clients. The other workload (*uniform trace*) is randomly generated from the request corpus, without considering the popularity. The averaged response time and cache hit rate are used for performance comparison.

4.2 Effects of Caching

As the first experiment, we study CiteSeer’s performance in answering requests under different system settings. The actual trace and uniform trace are both used for comparison, whose results are given in Figure 6.

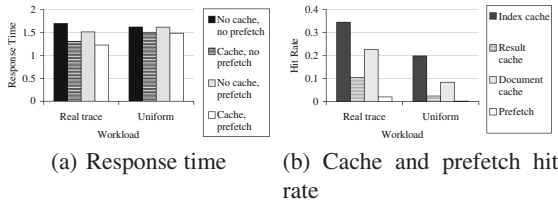


Fig. 6. Cache effect study

Figure 6(a) compares the system’s response time under various running conditions. It is obvious that using cache can improve the performance dramatically (27.65% improvement). Considering the fact that document retrieval requests contribute a dominating percentage in the actual trace, prefetch technique can further improve the response time. Also, it is shown that our caching mechanism works better for the actual workload than the uniform workload. This is because there exists a strong Zipf-like distribution in request semantics in the actual trace, which is absent in the uniform workload. In the following experiments, only actual trace is used.

Figure 6(b) gives the cache hit rates for each cache component. Index cache has a higher hit rate than any other cache component, because some queried terms are widely shared by many queries. In addition, document cache is more effective than the result cache, suggesting temporal locality in requested documents is stronger than queries, which can be observed from Figure 2 as well.

4.3 Cache Size Effects

In this section, we change the cache size parameter S . The size of the prefetch buffer and the fraction of each cache component is fixed.

It is expected that a larger cache size can bring better cache hit rate, and thus returns a shorter response time. This expectation is confirmed from the above figures. However, huge cache size is inefficient in cost and storage. As we can see from Figure 7(a), the response time does not change too much as the cache size reaches 1GB, showing a

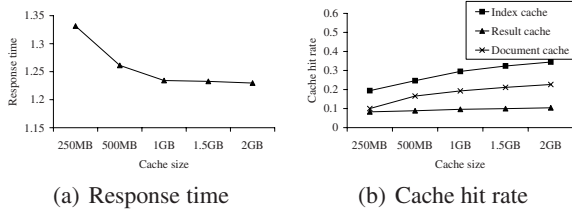


Fig. 7. Cache size effect study

reasonably good cache performance can be achieved without a larger cache. Figure 7(b) indicates when the cache size reaches 1GB or larger, all cache hit rates only increase in a slow speed with the size.

4.4 Correlation Threshold Effects

When we construct the correlation set, we use the correlation probability threshold to filter the strong-correlated request pairs. In the default setting, this value is fixed at 0.1, indicating a 10% or higher correlation probability is required. In this experiment, we change the threshold value to observe its effect.

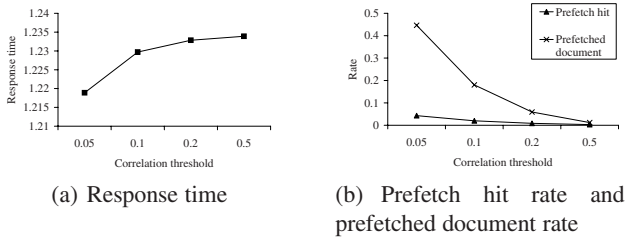


Fig. 8. Correlation threshold effect study

From Figure 8(a), we find a low threshold is very useful in improving the average response time. However, as seen from Figure 8(b), a low threshold means more correlated request pairs are qualified, and thus more documents are prefetched, most of which are not accessed by follow-up requests. When we set the correlation threshold to be 0.05, a document prefetch rate of 0.45 is reached. This means that on average each request will trigger 0.45 document to be stored in the prefetch buffer. However, the hit rate at the time is only around 0.05, meaning most of them will be replaced from the buffer without any access. The above analysis indicates us to control the correlation threshold at a reasonable level to optimize the performance.

4.5 Size Parameter Tuning

In this section, we investigate the problem of tuning each cache component size, assuming the total cache size is fixed.

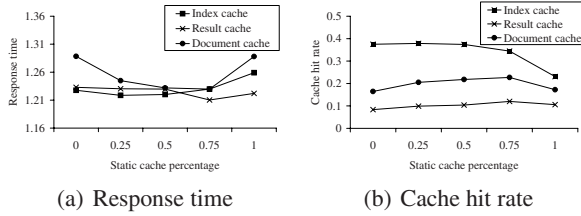


Fig. 9. Tuning static cache percentage

In the first experiment, we fix the assigned sizes for each cache component and respectively vary the static cache size in each component. Each curve in Figure 9 gives the study result for a specific cache component. Generally, the three curves are similar in their shapes in Figure 9(a) and Figure 9(b) respectively. It is suggested that the extreme cases (the entire cache uses static or dynamic policy) do not produce the optimal performance. This study also gives evidence in supporting a segmented cache management policy for search engines. It is also found that the optimal points on the curves are different for each cache component, indicating the necessity to study the distributional request feature behind each type of cache to optimize it.

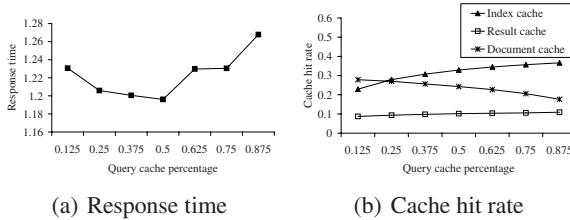


Fig. 10. Tuning query cache percentage

In the next study, we keep the static-dynamic ratio in each cache component constant and try to tune the storage reserved for the query caches (result cache and index cache) and the document cache. We perform this analysis because the two targets serve different request types. From Figure 10(b) we can clearly see how the two cache categories compete for cache space. The document cache performance is sacrificed if we increase the query cache sizes, and vice versa. Overall, the system reaches its best performance in the middle of the curve shown in Figure 10(a), meaning the cache should be divided approximately evenly for the query caches and the document cache. This value is affected by the composition of the request stream, in which we find the document retrieval requests are approximately 5 times as many as the query requests in CiteSeer.

Finally, we study the details in the query caches by modifying sizes reserved for the index cache and the result cache. The results (Figure 11) indicate the optimal size ratio is not reached in the extreme case as well. We can see that the hit rates for both caches do not change dramatically as long as their sizes are not extremely small. Thereafter, the response time is not very sensitive to the size ratio between the two cache components.

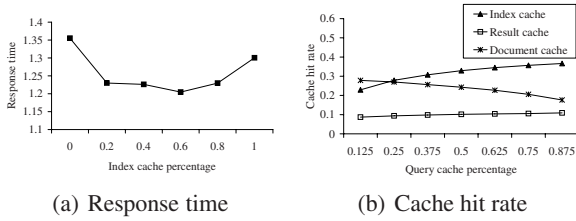


Fig. 11. Tuning index cache percentage

5 Related Work

A large number of works have been performed to study user behaviors in search engines by analyzing logged data. As one of the first few papers discussing search engine user behavior, [3] analyzed query logs of Fireball, a popular German search engine, showing a large portion of users only browse the first page of returned results. [17] studied Altavista search engine and found the majority of the users submitted only one query before the session ended. [13] shows the existence of locality in the queries for Excite search engine. The results are later confirmed by another study [16]. A late work on Altavista [10] shows over 67% of the logged queries are requested only once, while other few queries are requested frequently. Again, the Zipf distribution is observed.

Based on these observations, server-side cache management mechanism is adopted to improve system performance. Basically, two types of caches are proposed [16]. One choice is the *Result Caching* [9,13]. The previously processed query results are cached to answer future identical queries. The alternative approach is to cache inverted index other than results. The idea of inverted index caching is to extract a portion of the entire index to store in the cache [4]. Corresponding cache replacement policy that is based on access semantic is enforced. Meanwhile, the application is able to support dynamic query evaluation based on cache contents. [16] exploits the opportunity to combine the previously addressed two caching types in the search engine *TodoBR*. [12] found it is often observed that large web search engines are related to multiple data sources. Hence, a new intermediate level is added to cache intersections or projections of the frequently occurred term pairs, which contributes to the three-layer architecture. Furthermore, [18] suggests a hybrid composition of search engine cache which maintains a static set of statically locked cache entries and a dynamic set which contains replaceable cache entries. In addition, a speculative result page prefetching strategy is added to improve the hit ratio. Prefetching technique is also referred in other web applications [14,15].

Another popular cache study direction is in the infrastructure level. In summary, based on the usage patterns, it is interesting to find the best place to cache web contents [19,20]. Cache is widely used in other research domains. [8,11] studied how to maintain caches in mobile environment, with limited resources. [1] uses profiles to manage caches, which can be used for personalized services.

6 Conclusion

In this paper, we present our initial design of a cache framework for a scientific literature search engine, CiteSeer. Our workload analysis on the usage logs indicates characteristics of requests in such a system, including temporal localities and the correlation of requests. Based on our findings, the proposed system cache is comprised of multiple components, which include index cache, result cache and document cache. Correlation-based prefetch is provided to further improve the document retrieval efficiency. Experimental results show that our caching mechanism produces dramatic improvement in the system's response time.

Our future work will investigate the research interests of users to decrease the request ambiguity and improve the service precision. Another future research direction is to develop an adaptive cache management policy to tune cache parameters in real time.

Acknowledgements

We gratefully acknowledge partial support from Microsoft Research, NSF and NASA for CiteSeer project. Wang-Chien Lee was supported by NSF under Grant No. IIS-0328881, IIS-0534343 and CNS-0626709.

References

1. Cherniack, M., Galvez, E.F., Franklin, M.J., Zdonik, S.B.: Profile-driven cache management. In: ICDE, pp. 645–656 (2003)
2. Giles, C.L., Councill, I.G.: Who gets acknowledged: measuring scientific contributions through automatic acknowledgement indexing. In: Proceedings of the National Academy of Sciences. vol. 101(51), pp. 17599–17604 (2004)
3. Hölscher, C.: How internet experts search for information on the web. In: WebNet (1998)
4. Jónsson, B.T., Franklin, M.J., Srivastava, D.: Interaction of query evaluation and buffer management for information retrieval. In: SIGMOD Conference, pp. 118–129 (1998)
5. Karedla, R., Spencer Love, J., Wherry, B.G.: Caching strategies to improve disk system performance. IEEE Computer 27(3), 38–46 (1994)
6. Lawrence, S., Bollacker, K.D., Giles, C.L.: Indexing and retrieval of scientific literature. In: CIKM, pp. 139–146 (1999)
7. Lawrence, S., Giles, C.L., Bollacker, K.: Digital libraries and Autonomous Citation Indexing. IEEE Computer 32(6), 67–71 (1999)
8. Ken C., Lee, K., Lee, W.-C., Zheng, B., Xu, J.: Caching complementary space for location-based services. In: EDBT, pp. 1020–1038 (2006)
9. Lempel, R., Moran, S.: Optimizing result prefetching in web search engines with segmented indices. In: VLDB, pp. 370–381 (2002)
10. Lempel, R., Moran, S.: Predictive caching and prefetching of query results in search engines. In: WWW, pp. 19–28 (2003)
11. Lim, S., Lee, W.-C., Cao, G., Das, C.R.: A novel caching scheme for improving internet-based mobile ad hoc networks performance. Ad Hoc Networks 4(2), 225–239 (2006)
12. Long, X., Suel, T.: Three-level caching for efficient query processing in large web search engines. In: WWW, pp. 257–266 (2005)

13. Markatos, E.P.: On caching search engine query results. *Computer Communications* 24(2), 137–143 (2001)
14. Min-You, Y.J.: Web prefetching: Costs, benefits and performance.
15. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: Effective prediction of web-user accesses: a data mining approach. In: Kohavi, R., Masand, B., Spiliopoulou, M., Srivastava, J. (eds.) *WE-BKDD 2001 - Mining Web Log Data Across All Customers Touch Points*. LNCS (LNAI), vol. 2356, Springer, Heidelberg (2002)
16. Saraiva, P.C., de Moura, E.S., Fonseca, R.C., Wagner Meira Jr., Ribeiro-Neto, B.A., Ziviani, N.: Rank-preserving two-level caching for scalable search engines. In: *SIGIR*, pp. 51–58 (2001)
17. Silverstein, C., Henzinger, M.R., Marais, H., Moricz, M.: Analysis of a very large web search engine query log. *SIGIR Forum* 33(1), 6–12 (1999)
18. Silvestri, F.: High performance issues in web search engines: Algorithms and techniques. Ph.D. dissertation. Università degli Studi di Pisa—Facoltà di Informatica, Pisa, Italy.
19. Wong, T.M., Wilkes, J.: My cache or yours? making storage more exclusive. In: *USENIX Annual Technical Conference, General Track*, pp. 161–175 (2002)
20. Xie, Y., O'Hallaron, D.R.: Locality in search engine queries and its implications for caching. In: *INFOCOM* (2002)

Finalizing Dialog Models at Runtime

Stefan Betermieux and Birgit Bomsdorf

Fernuniversität in Hagen, 58095 Hagen, Germany

stefan.betermieux@fernuni-hagen.de,

birgit.bomsdorf@fernuni-hagen.de

<http://www.fernuni-hagen.de>

Abstract. This paper proposes a dialog model for web applications aiming at flexible interface generation. The basic idea is to enable the runtime system to “finalize” the dialog structure. The overall approach follows a task-oriented, user-centered development process, where models of the users’ tasks and the user-system dialog play an essential role. In our approach, these models are transferred to the run time system that allows the user to interact with the web application according to the specifications. It is based on an architecture that separates a task controller and a dialog controller, which are responsible for model execution and dialog creation. Throughout the paper, we take care of the special characteristics of web applications and show enhancements of the conceptual models and of the runtime architecture.

1 Introduction

Web sites have been developed towards highly interactive web applications. The web site visitor is no longer a mere recipient of information but a user interacting with an application, e.g., filling in data, triggering system functions, and receiving feedback from the application. Thus, web pages evolve more and more into user interfaces; besides providing content they have to support user-system interaction, also referred to as the dialog. Furthermore, up-to-date web applications, such as e-shops, include business processes. The system has to guide users through a predefined sequence of web pages through which they perform the activities of the process.

In the field of model-based design of user interfaces (e.g. [7], [20], [21], [24], [25]) well-known notations and techniques exist for developing systematically the dialog of an interactive system. They were developed, however, for modelling traditional user interfaces. Within the web modelling community ([22], [15], [13], [17], [5]), on the other hand, the interactive behavior is specified basically implicitly within the navigation model. In some approaches, similarly to the field of HCI, models describing the system from the view of the user are introduced. In WSDM [13], for example, task models are used as a high-level dialog description to guide the design of the navigation. In OOHDM [26], task descriptions are analyzed to identify the data items, which are to be exchanged between the user and the web application. In general, web modelling approaches are complemented

increasingly by the investigation and specification of user goals, tasks and activities, respectively (further examples are given by WebML [8], UWE and OO-H ([10], [17])). All in all, these aspects are used as informal input to conceptual domain, navigation and presentation design. Since interaction design and feedback specification becomes more and more important, this information should not be an add-on to an otherwise data-oriented methodology. Both a data-centric view and a task- and interaction-centric view are required. However, if a modelling approach supports mainly the data and object based specification treating dialog specification implicitly only, achievement of a usable interaction design can be very cumbersome. For example, spreading pieces of interaction specification over the navigation model makes it difficult to detect common structures and patterns, respectively. From our point of view, developers have to be supported by modelling concepts that put the dialog explicitly and coherently into play.

The work presented here proposes a flexible dialog model. The whole approach follows a task-oriented, user-centered development process. However, in this paper we focus on the conceptual task model and its associated abstract user interface model. In most existing approaches these models are taken as input to subsequent development steps, within which the sites and pages are modelled in more detail aiming at the final web pages and navigation. In contrast to this, in our approach the task and abstract user interface model are passed over to the run time system. Its control component comprises a task controller and a dialog controller responsible for “finalizing” the dialog model. The objective is, to use this technique in the adaptation of web pages to various screen sizes. Implementation of the overall framework is work in progress. The scope of this paper is to introduce the basic concepts of the *flexible* dialog.

2 Task Modelling for Web Applications

Modelling of a web application usually starts with requirement specification, similarly to traditional Software Engineering. The objective is to get a picture, as clear as possible, of information and functional as well as of usability requirements. Use case diagrams are commonly in use for a first description of them, later on refined by means of, e.g., activity diagrams or task models. Like WSDM [14], we apply task modelling for this step. While in WSDM a modified version of CTT [20] is used, we adapted the notion of VTMB [4] for the concerns of web modelling. Initially VTMB was developed to support task modelling in the context of traditional user interfaces. Focussing on interactive web applications, additional concepts are added by our current work. The concepts as relevant within this paper are introduced below, whereby we concentrate on task models as applied in conceptual design.

2.1 Task Model

Throughout this paper, a task is denoted by means of a symbol as depicted in figure 1. Defining temporal relations (top right in the task symbol) and cardinality



Fig. 1. Example of a task symbol (representing the task Buy Car)

(top left) is well-established in task modelling. In addition, we enriched it by the task lifespan (denoted bottom left).

Task Cardinality. Information can be attached to a task symbol to express the possible number of executing that task. Internally, minimum and maximum values are used to control at run time the cardinality constraints. To support readability of task models, cardinality information is attached to task symbols by means of expressive names.

Optional (opt). The label *opt* denotes that the task can be omitted.
 $minimum = 0$

Iteration (iter). A task labeled with *iter* can be performed as often as needed.
 $maximum = \infty$

Mandatory. If no label is assigned to a task symbol, the task is to be executed exactly one time, which is the default value for task performance.
 $minimum = 1, maximum = 1$

Temporal Relations. Task relations are a basic concept regarding hierarchies of tasks. In contrast to CTTs [20], we assign a temporal operator to the relation between a parent task and its children tasks and not individually between the siblings.

The universal relation is always an aggregation of the child tasks to a parent task (is-part-of relation); a parent task is fulfilled, as soon as the sub-tasks are fulfilled according to the temporal relation given. This is a very basic definition of the possible semantics of the relation, they are refined to be more useful in the context of task models.

Sequence (seq). The subtasks of a parent task have to be completed in sequence. The second subtask can only be performed, if the first subtask is finished. This process is continued for all subtasks. The parent task is fulfilled, as soon as the last subtask is finished.

Arbitrary Sequence (arb). The subtasks can be completed in an arbitrary sequence. Only one subtask can be performed at one time. The parent task is fulfilled, when all subtasks are finished.

Selection (sel). Only one of the subtasks can be selected to be performed. The parent task is fulfilled, when that subtask is finished.

Parallel (par). All of the subtasks can be performed in parallel. There can be more than one subtask performed at the same time. The parent task is fulfilled, if all subtasks are finished.

Lifespan. In the context of web applications, task models have to cope with user originated and system originated task suspensions. This holds true for traditional applications as well, but has a slightly different meaning in the context of the web.

User originated task suspension occurs, when the user navigates explicitly to a page which would not be accessible by the current task model, for example by using bookmarks or the browsers navigation buttons. These out-of-context navigations cannot be solved automatically, but have to be addressed by the task model designer.

Due to the stateless hypertext transfer protocol (HTTP), the server needs to associate user requests to a memory area where the state of the application of a user is saved. Since the memory on the server is finite and users may possibly never return, the server needs to clean up this memory area (the user session), normally by using timeouts after the last request. Hence, *system originated task suspension* occurs, when the web application server terminates a user session which contains a running task model instance.

Either way, the task controller needs to detect those exceptional events and react to them with a predictable behavior. Instead of dictating a single task model suspension strategy, it is possible to attach those strategies to tasks, where they only affect the task and all of their subtasks. It is possible to mix strategies in a task model depending on application requirements, as long as the tasks with different strategies are on separate subtrees. The example in section 2.2 will explain this behavior in more detail.

Volatile. The default strategy for tasks without a lifespan attribute. If a suspension occurs, the states of the task and all its subtasks are discarded.

Suspendable (susp). If a task is marked as suspendable and suspension occurs, the task controller will put it and all its descendants into a suspended state. The user is offered an interface option to resume the suspended tasks.

Persistent (pers). A task marked with pers is suspendable as defined by susp and additionally will be persisted to a long term storage space (i.e. database) when a session timeout occurs. This is only possible for users which can be re-identified in the next session, for example by a name/password registration, or a site cookie. During the next session, the user is offered an interface option to resume the previously suspended tasks.

2.2 Example

Figure 2 shows a simplified model of a web application, which uses most of the above mentioned concepts. The example is taken from an on-line picture gallery where users can browse existing pictures and upload new ones. Starting from the root task *On-line Picture Gallery*, three subtasks can be invoked independently (the relation to the subtasks from the root tasks is marked as parallel). The sub task *Help* provides context sensitive help and can be re-invoked if finished (task is marked as iteration). The sub task *Browse Pictures* is subdivided into two subtasks where the user has to chose one of the options (hence the selection).

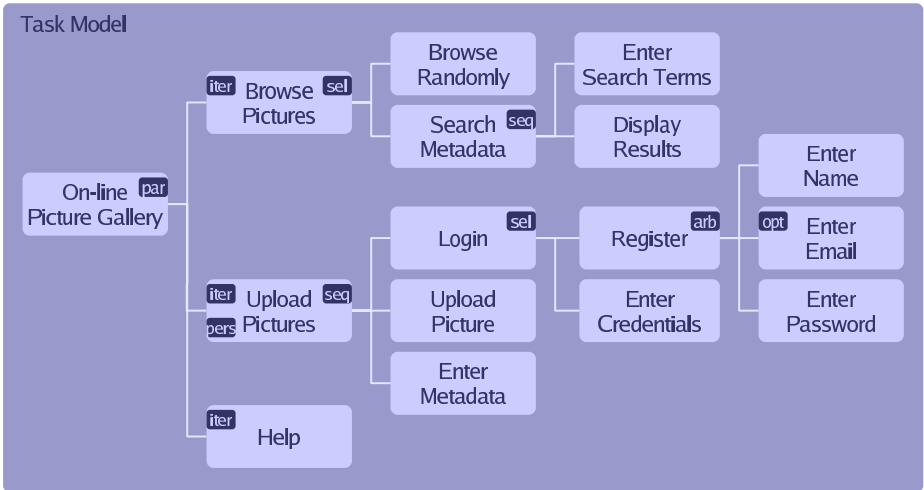


Fig. 2. Task Model

Browse Randomly is a leaf since the granularity of task refinement is adequately modelled for our purposes. *Search Metadata* is subdivided into two sequential subtasks, the first presenting a search form and the second displaying the results. The largest subtree of the application *Upload Pictures* relates to a process to register with the web application and to upload pictures. This process is marked as persistent; whenever a user leaves the web site before completing the process (subtree), it is persisted and presented again when the user returns. On the other hand, browsing pictures and invoking help are volatile, thus the states are discarded when the user leaves the web site. Uploading pictures consists of a task sequence: logging in, uploading a picture, and entering meta-data for the picture. *Login* is a selection between first time registration and authentication by using a given username/password combination. *Registration* is subdivided into three tasks, which can be performed in an arbitrary sequence: entering the user's name, entering an email address (which is optional) and choosing a password.

3 Domain Model

The domain model describes the objects of the business domain, their properties in terms of attributes, sub-object structures, and semantic relationships. Techniques adopted for defining this model correspond to well-known diagrams from Software Engineering (such as the class model in UML) or database engineering (ER diagrams). We will use UML (see figure 3). A simple entity class¹ *Account* is used to store user credentials and a service class² *AccountService* is used to create new accounts and to retrieve existing ones.

¹ See [1] for a definition of the stereotype *Entity*.

² See [1] for a definition of the stereotype *Service*.

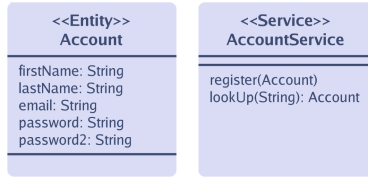


Fig. 3. Domain Model

4 Abstract Dialog Model

The task model is not yet suitable to generate the dialog between the user and the domain model. A dialog describes messages in terms of input and output interactions exchanged between the user and the system. From a users point of view, a dialog state represents the “position” within a dialog while the user is interacting with the application. The positions, i.e., the states, are changed by task execution. The tasks are associated with input and output declarations to specify what kind of data may be exchanged between the user and the system to perform a task. To enable this data exchange, appropriate interaction components have to be defined. Such information is typically described by means of a user interface model, comprising an abstract and concrete description of the presentation [20]. The abstract model describes *what* will be shown to the user at one point in time on a web page. The concrete model, also called the visual design, described *how* the web pages will be presented defining the concrete layout in terms of colors, fonts, the logo to be inserted and so on.

4.1 Basic Components and Grouping Mechanism

In our approach we introduce abstract dialog units (ADUs) for specifying the abstract user interface [2]. ADUs are associated to tasks from the task model, specifying abstract interface components for displaying output and transferring data from input elements. Thus, the abstract presentation of a web application is closely related to the dialog structure.

The abstract user interface can be constructed using predefined interaction objects, i.e., output objects (text, image) and input objects (text input, checkbox input). Since we are focusing on web applications, these components, which we refer to as generic interface components, correspond to HTML input and output elements. Figure 4 shows some of the components used in the example later on. *Label* just outputs some text, *Image* displays a picture. *Input* creates a text input and links it to a field from the domain model. The type attribute can automatically convert strings (HTTP request parameter are always strings) into domain model types and the validate attribute can perform simple constraint checks on user input (i.e. not empty, integer range, etc.). *Secret* has the same

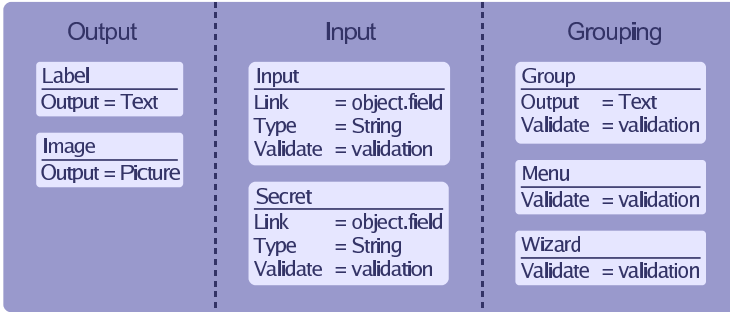


Fig. 4. Generic Interface Components

capabilities as an Input component, but the behavior of the input field displayed in the browser, which is cloaked with asterisks.

Since a task requires most of the time more than a single input or output, generic interface components can be grouped together using grouping components. A simple grouping component *Group*, specifies an ordered aggregation of its subcomponents. It outputs some text as the title of the group, displays the subcomponents as a list and can run validations on the aggregated subcomponents. *Menu* also groups subcomponents, but presents them as a link list to navigate to them individually, useful to create a menu. *Wizard* groups its subcomponents into sequential steps.

More complex components, like data grids and tree structures, can be added later on. The hierarchical structure of complex ADUs can be represented visually by means of a tree notation as shown in figure 5. The complex task *Enter Name* is composed of a set of label and input components, aggregated by the group component.

4.2 Connecting the Domain Model

The domain model can be created after the task model has been finalized, or, in case of legacy applications, can exist beforehand. In both cases, the domain model needs to be linked to the task model, and there are two steps involved.

Input components need to be linked to domain model fields, creating *value bindings*. These links are bidirectional, existing values from the domain are used to initialize the input fields. After a subsequent request arrives from the user, the user input is written back into the domain model.

Methods in the domain model can be invoked during task performance. Special fields in the ADUs called *onEntry* and *onExit* are used to link to methods in the domain model, creating *method bindings*. They are unidirectional, since method invocation always originates from the task model and targets the domain model.

While value bindings just transfer data from and to the domain model, method bindings can initiate processing on the data.

4.3 Abstract Dialog Model Example

The example (figure 5) in this section demonstrates the attachment of ADUs to tasks. First of all, the Register task is performed and the onEntry method binding is invoked, creating an empty Account object. The group component merely displays a title for all subtasks, the validation is later triggered, when the task *Register* is completed, i.e., all subtasks are completed (arbitrary sequence).

Enter Name creates two input fields with associated labels in a group and uses value bindings to link into the domain model. Validation just checks for empty strings.

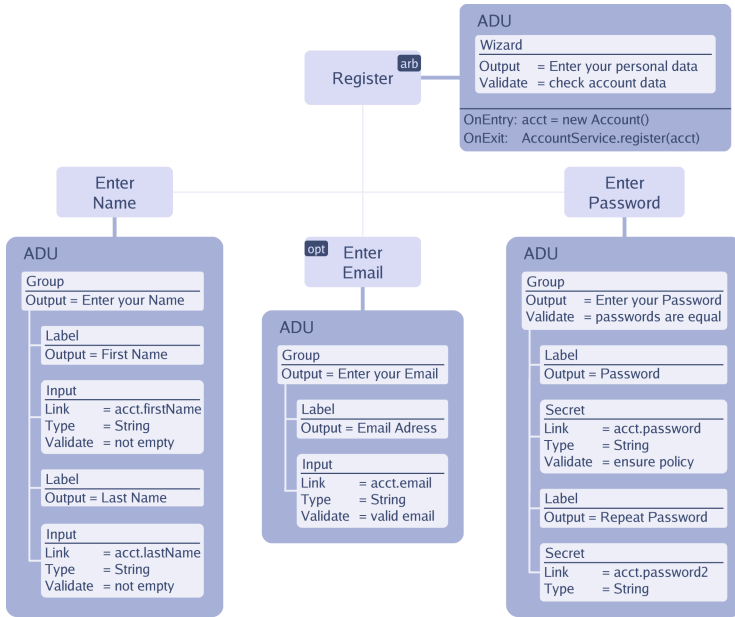


Fig. 5. Abstract Dialog Units Attached to Tasks

Enter Email creates a single input field and a label in a group. The validator checks for a syntactical correct email address and updates the value binding only if the email address string from the request qualifies.

Enter Password creates two input fields with asterisk cloaking and ensures for the first password, that it doesn't violate the password policy. The grouping component validates the entered passwords and checks if they are equal. Since the first password field doesn't know about the second password field, the check of equality can be performed at the grouping layer only, since here the data of both passwords entries are available to the validator.

After performing all three subtasks, the parent task *Register* is finished, the validation of the parent group is triggered, which checks the entered account

data for inconsistencies created by contradicting inputs in separate subtasks. Finally, the `onExit` method binding is called and the account is created.

5 Runtime Architecture

Based on the models we described in the last sections, we propose a runtime architecture to process instances of these models, i.e. a task controller which processes task model instances and a dialog controller which processes abstract dialog model instances. Figure 6 displays the overview of the architecture and the flow of a request from a user’s web client.

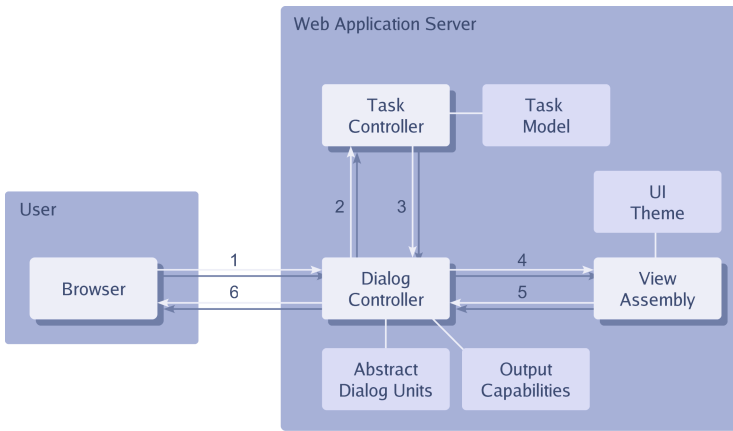


Fig. 6. Architecture View

1. All requests are handled by the *dialog controller*³. It associates requests to running task model instances and performs basic validation. 2. The *dialog controller* invokes the *task controller*, which promotes the task model instance. 3. The *task controller* returns. 4. The *dialog controller* delegates the transforming of the ADU tree into a web page to the *view assembly*. 5. The *view assembly* returns a web page. 6. The *dialog controller* returns the web page to the browser.

The architecture is based on the classic model-view-controller (MVC) pattern for web applications [23], with a special focus on controllers. The *model* matches the domain model, the *view* is created in the view assembly and the *controller* corresponds to the task- and dialog controllers in cooperation with the task- and abstract dialog model.

Task- and dialog controller are explained in more detail in the next sections. The view assembly doesn’t need a more detailed specification at the moment, because it just maps generic interface components to their HTML counterparts. Later on, with more complex generic interface components available, the mapping part will be more sophisticated.

³ Since it acts as a single point of entry, it is also known as front controller. [1]

5.1 Task Controller

The task controller is responsible for creating task model instances from a dedicated task model and managing their lifecycles. For each task it creates a finite state machine (FSM, see figure 7), which consists of the possible task's states and the transitions between the states triggered by events.

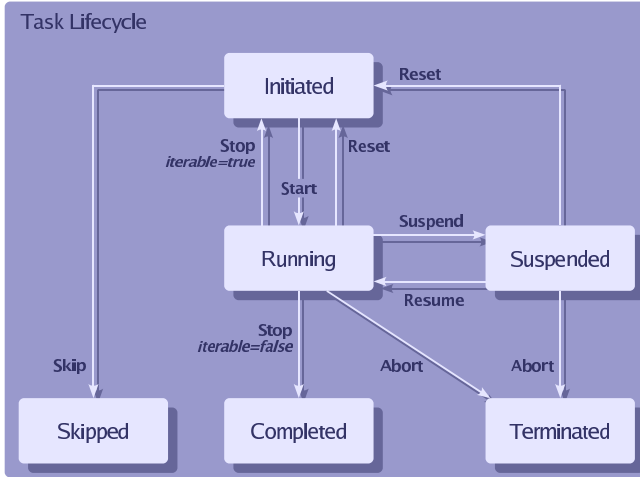


Fig. 7. Task Finite State Machine

Initially, the FSM is in the *initiated* state, and can be started or skipped. If skipped, the FSM changes into the *skipped* end state, counting the task as not completed but successfully finished. For example, if an optional task in a sequence is skipped, the subsequent task can be performed.

From *initiated*, the event *start* changes the FSM into the *running* state. By entering the *running* state, the method binding *onEntry* will be invoked. If the FSM receives a stop event, it will invoke the *onExit* method binding and change into the *completed* end state (or the *initiated* state, if the task is iterable).

Since we have to cope with suspensions, it is always possible that the task needs to be suspended while running. Depending on the lifespan of the task, it changes into the *suspended* state (if *lifespan = suspendable* or *persistent*) or in the *terminated* end state (if *lifespan = volatile*).

The hierarchy of tasks in the task model is adhered by adding conditions based on temporal relations to the transitions of the finite state machines. For example, to finalize a parent task (transition from *running* to *completed*) in an arbitrary sequence, all child tasks need to be in the state *completed* or *skipped*.

5.2 Dialog Controller

Fundamental part of the dialog controller is the *component tree*, which is used to interact with the user. ADUs are linked to tasks that need user interaction. Since we don't have generated pages yet, the information inherent to the task model hierarchy can be used to combine ADUs in order to present them on web pages. Using a task model instance from the task controller, the dialog controller creates a component tree by adding the ADUs from running tasks to a component tree, starting with the ADU from the root task and traversing the task hierarchy with a depth first search. The resulting "virtual" *generic interface component tree* was not modelled by the designer, but is created by the dialog controller to interact with the user. The component tree for the example in figure 5 is depicted in figure 8.

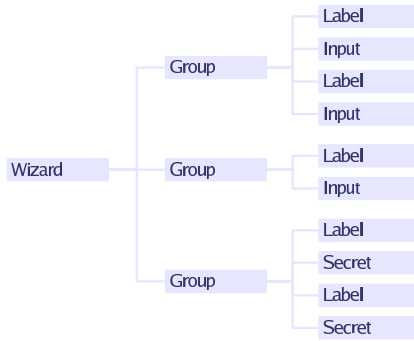


Fig. 8. Component Tree

For example, if a web site needs to be accessible using a desktop computer and a PDA, the dialog controller can decide at runtime how much screen size is used by the interface components and change the spatial appearance of grouping elements. The *Register* task in figure 5, could be presented on one page or on three subsequent pages. To combine all register tasks into a single page, the dialog controller would decide to change the appearance of the grouping element *Wizard* to display all its subcomponents at once. Since this can change the final user interface in a bad way (jumping recurring interface elements, no recognition of the web site), the change of grouping appearance should be applied to the component tree bottom-up and is still research in progress.

Figure 9 depicts the internal flow of the dialog controller, which is divided into six sequential phases⁴.

Restore Component Tree. assigns a previously stored component tree from the user session to the request, which is only possible if the request originates from a web page created by the dialog controller (i.e., a component tree has

⁴ This is inspired by the request processing of Java ServerFaces([19]).

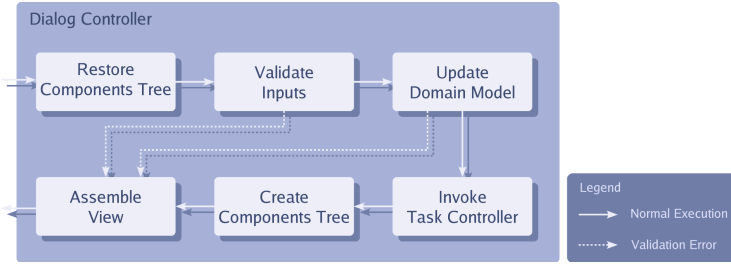


Fig. 9. Dialog Controller

been created and stored). If a component tree is found, the request attributes, which correspond to user input, populate the component tree and the next phases only use the component tree to process user interactions.

Validate Inputs. The validations of the generic interface components of the component tree are processed. If, for example, the last name field of the registration example (figure 5) has been left empty by the user, the validation would fail (mandates not empty) and the processing of the request would stop here and jump immediately to the Assemble View phase to create the same page with a validation error message, allowing the user to correct mistakes.

Update Domain Model. The values of the input components of the component tree are passed over to the domain. If the domain does not accept some of the input, the processing of the request would stop here again and jump immediately to the Assemble View phase.

Invoke Task Controller. The task controller creates a new instance of the task model if it doesn't exist and stores it in the user session. It promotes the task model instance by starting and stopping tasks depending on the users interactions. It invokes domain model methods by executing the onEntry and onExit method bindings of started and stopped tasks. The lifecycle of tasks is explained in section 5.1

Create Component Tree. Due to the changes in the task model instance in the previous phase, the set of active tasks may have changed. The dialog controller creates a new component tree based on tasks in *running* state to interact with the user. After creating the tree, it is stored in the user session.

Assemble View. Pass the component tree to the view assembly, were a HTML page is created. Insert context information into the resulting page, which allows the dialog controller on a subsequent request to associate the user to a running task model instance.

6 Related Work

WebML [11] provides mainly a visual notation for defining data-driven hypertexts. The core is made up by a continually expanded set of so-called units, e.g., Data Unit, Entry Unit, Login Unit. The units represent what is to be included on

a web page, whereby the expressiveness range from simple (e.g., a Data Unit represents information from the underlying data base) up to little processes (such as specified by the Login Unit). Application dependent sequences of task execution (called workflows) can be specified by means of operation units with additional information concerning control conditions. All in all, interaction elements, dialog control and presentation are not modelled separately.

Separation of concerns is better realized in UWE and WSDM. UWE [16] is an object-oriented approach supporting the whole life-cycle of web application development. UML is used, including some extension enabling the specification of, e.g., a navigation model. The approach has been extended for the purpose of modeling task sequencing (called business processes in UWE). Similarly, as in our approach, task modelling is distinguished within analysis and within design. At design level, a process (task) is modelled in terms of classes that are refined by UML activity diagrams. A process class specifies the information needed for performance, including state information for the purpose of handling interruptions. The integration of the process classes into the navigation model result into two separated but connected spaces: one dedicated to process performance and the other dedicated to "pure" navigation / browsing. In our approach, similar to OO-H [9], both aspects are interwoven in the navigation model.

The the objects chunks of WSDM [14] are similarly to the ADUs proposed by our approach. While in WSDM such chunks are only attached to leaf tasks, in the work presented here ADUs can be attached to a task regardless of its position within the task hierarchy. By this, we can easily denote aspects being relevant to a complete sub-task tree.

6.1 Current State of Work

Our work differs from all mentioned approaches above with respect of utilizing the "task views". In most approaches, including those developed in the field of HCI, e.g., [20], [12], [18] the task / process structure is transformed into an initial dialog model or navigation model at design time. We are postponing dialog creation at runtime to adapt the user interface to client specific requirements. The biggest drawback of this approach is poor imagination of the generated web pages by the task designer, which we try to counter with predictable behavior of grouping elements. Still, there is a trade-off between adaptability and user interface consistency.

The concepts were applied in some projects so far. For this purpose, our previous work on simulating task models ([4]) was extended to support interactive web applications at run time (an example of which was presented in ([3])). As shown in ([5], [6]) the web-MVC pattern was extended by task-related components. A task processing manager is responsible for keeping track of state information at the level of task and process execution, respectively.

We are implementing task- and dialog controllers to support our ideas of task suspension and runtime dialog generation. We have developed an XML schema of our task notation, to support easy integration of task models into the controllers. The task controller, which creates FSMs of the tasks is finished and

deployable. The dialog controller, which uses a component tree and generates web pages is also in a running state. Current work aims at the integration of both controllers into a coherent runtime system. An editor to create task models has been developed and is able to export the models in XML.

References

1. Alur, D., Crupi, J., Malks, D.: *Core J2EE Patterns: Best Practices and Design Strategies*, 1st edn. Prentice Hall, Englewood Cliffs (2001)
2. Betermieux, S., Bomsdorf, B., Langer, P.: Towards a generic model for specifying different views on the dialog of web applications. In: *Proceedings of HCI International*. Lawrence Erlbaum Associates, Mahwah, NJ (2005)
3. Biedebach, A., Bomsdorf, B., Schlageter, G.: The changing role of instructors. In: *Proceedings eLearn 2002* (2002)
4. Biere, M., Bomsdorf, B., Szwillus, G.: The visual task model builder. In: *Proceedings of CADUI 1999* (1999)
5. Bomsdorf, B.: First steps towards task-related web user interfaces. In: *Proceedings of Computer-Aided Design of User Interfaces*, pp. 349–356 (2002)
6. Bomsdorf, B.: Task modeling for customization of web applications. In: *Proceedings HCI International 2003*, pp. 33–37 (2003)
7. Bomsdorf, B., Szwillus, G.: Towards a universal modelling tool. In: Palanque, P., Paternó, F. (eds.) *DSV-IS 2000*. LNCS, vol. 1946, Springer, Heidelberg (2001)
8. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process modeling in web applications. In: *ACM Transactions on Software Engineering and Methodology* (2006)
9. Cachero, C., Gómez, J., Pastor, O.: Object-oriented conceptual modeling of web application interfaces: the OO-H method. In: *ECWEB'00*, pp. 206–215. Springer-Verlag, Heidelberg (2000)
10. Cachero, C., Koch, N.: Conceptual navigation analysis: a device and platform independent navigation specification. In: *Proceedings of 2nd International Workshop on Web-Oriented Software Technology* (2002)
11. Ceri, S., Fraternali, P., Matera, M., Maurino, A.: Designing multi-role, collaborative web sites with WebML: a conference management system case study. In: *IWWOST 2001 Workshop* (2001)
12. Clerckx, T., Luyten, K., Coninx, K.: The mapping problem back and forth: customizing dynamic models while preserving consistency. In: *TAMODIA '04*. Proceedings of the 3rd annual conference on Task models and diagrams, pp. 33–42. ACM Press, New York, NY, USA (2004)
13. de Troyer, O.: Audience-driven web design. In: *Information modelling in the new millennium*, IDEA Group Publishing (2001)
14. de Troyer, O., Casteleyn, S.: Modeling complex processes for web applications using WSDM. In: *Proceedings of the Third International Workshop on Web-Oriented Software Technologies, IWWOST* (2003)
15. Isakowitz, T., Kamis, A., Koufaris, M.: The extended RMM methodology for web publishing. Working Paper IS-98-18 (1998)
16. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of business processes in web application models. *Journal of Web Engineering* 3(1), 22–49 (2004)
17. Kraus, A., Koch, N.: A metamodel for UWE. Technical report, University of Munich (2003)

18. Luyten, K., Clerckx, T., Coninx, K., Vanderdonckt, J.: Derivation of a dialog model from a task model by activity chain extraction. In: DSV-IS, pp. 203–217 (2003)
19. McClanahan, C., Burns, E., Kitain, R.: JavaServer faces specification, v1.1, rev. 01 (2004)
20. Paternò, F., Mancini, C., Meniconi, S.: Concurtasktrees: A diagrammatic notation for specifying task models. In: INTERACT '97. Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, London, UK, pp. 362–369. Chapman & Hall, Ltd., Sydney (1997)
21. Puerta, A., Cheng, E., Ou, T., Min, J.: MOBILE: User-centered interface building. In: Proceedings of CHI 99. ACM Press, New York, NY, USA (1999)
22. Rossi, G., Schwabe, D., Lyardet, F.: Web application models are more than conceptual models. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, Springer, Heidelberg (1999)
23. Singh, I., Stearns, B., Johnson, M.: Designing Enterprise Applications with the J2EE Platform, 2nd edn. Addison-Wesley, London, UK (2002)
24. Stry, C.: Task- and model-based development of interactive software. In: Proceedings of IFIP 98 (1998)
25. van der Veer, G.C., Lenting, B.F., Bergevoet, B.A.J.: Groupware task analysis - modelling complexity. Acta Psychologica (1996)
26. Vilain, P., Schwabe, D.: Improving the web application design process with UIDs. In: Proceedings of 2nd International Workshop on Web-Oriented Software Technology (2002)

Transparent Interface Composition in Web Applications

Jeronimo Ginzburg¹, Gustavo Rossi², Matias Urbieta², and Damiano Distanto³

¹Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina
jginzbur@dc.uba.ar

²LIFIA, Facultad de Informática, UNLP, La Plata, Argentina and Conicet¹
{gustavo, matias.urbieta}@lifia.info.unlp.edu.ar

³RCOST – Research Centre on Software Technology
Department of Engineering, University of Sannio, Italy
distanto@unisannio.it

Abstract. In this paper we present an approach for oblivious composition of Web user interfaces, particularly for volatile functionality. Our approach, which is inspired on well-known techniques for advanced separation of concerns such as aspect-oriented software design, allows to clearly separate the design of the core's interface from the one corresponding to more volatile services, i.e. those that are offered for short periods of time. Both interfaces are oblivious from each other and can be seamlessly composed using a transformation language. We show that in this way we simplify the application's evolution by preventing intrusive edition of the interface code. Using some illustrative examples we focus both on design and implementation issues, presenting an extension of the OOHDm design model which supports modular design of volatile functionality.

1 Introduction

Even simple Web applications must deal with a myriad of concerns, each one of them encompassing multiple requirements. It is well known that by clearly decoupling application concerns in each development stage, and using proper composition mechanisms to weave corresponding design and implement artifacts, we can get more evolvable (Web) software [11]. However, while different techniques for advanced separation of concerns such as architectural and design patterns [9] and aspects [8] have been already introduced in the Web field (for example [2]), there are still many open problems, related with concern separation and composition, which have not been fully addressed. In this paper we focus on the design of volatile functionality, particularly on presentation issues. In a Web application there may be many different kinds of volatile requirements giving raise to volatile functionality. Some of them may arise during the application's evolution to check acceptability of the users' community (like beta functionality) and might be later considered or not core application services (e.g. forums or users' tags in Amazon.com). Others are known to be available only on short and determined periods of time such as functionality for giving donations after a catastrophe or sales for fixed periods of time (e.g. Christmas). Others are even more

¹ This paper was partially funded by Secyt Project PICT 13623.

irregular: some kinds of price discounts in e-stores (e.g. for specific left-over stock products), draws and concert tickets selling (associated with a CD), Amazon’s “fishbowl performance” for new releases of an artist, etc.

Volatile functionality may also affect irregular sub-sets of objects; for example only some CDs in a store are involved in a draw, artists’ performances in a video appear only in some novelties, etc. As an example, in Figure 1 we show the Amazon page of the last CD of Norah Jones, including a short video, which will be surely removed after some weeks. Additionally at the bottom of the page there is a link to a Valentine’s store, a volatile sub-store which was removed after St. Valentine’s day.

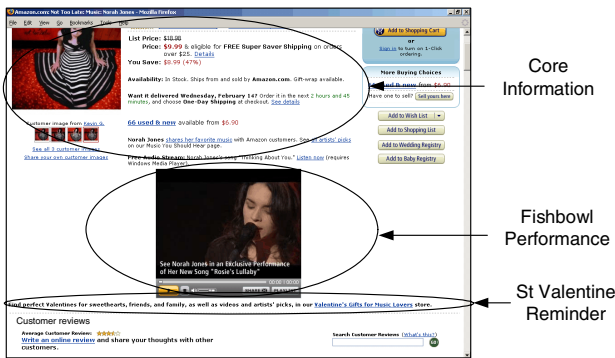


Fig. 1. Different volatile functionality in Amazon.com

Volatile functionality poses many challenges to the designer, and also to design approaches. Suppose for example that we have designed a CD class or entity type using any Web design approach. If we want that some CD objects exhibit some new behaviors or data (e.g. a video performance), we can either add an attribute to the class, create a sub-class for the new behavior, re-design the class (type) so that it is now a composite, encompassing as a component the new features, etc. Some of these solutions might be better than others depending on the context (e.g. sub-classification or class editing are not good solutions to deal with functionality attached to certain instances). However, in all cases we face a serious problem: being this functionality volatile, we might need to deactivate it after certain time. This means, once again, some kind of intrusive editing, which is cumbersome even if we are using a model driven approach and tool: the model has to be edited and this operation is certainly error prone. To make matters worse, if new requirements arose after introducing volatile modules, we might have a more complex tangle of core and volatile design components. Even using powerful configuration management environments, rolling back to the desired application’s configuration might be a nightmare.

Unfortunately, the army of existing Web design methods (object oriented ones like OOHDM [21], UWE [12] or OOWS [17], or those based on data modeling approaches like WebML [3]) lacks modeling primitives to deal with this problem. In [19] we presented an extension to the OOHDM approach to deal with volatile functionality, both at the conceptual and navigational levels. Following our approach, we model volatile features as first class entities (e.g. classes), which are completely

decoupled from core modules. Core and volatile classes are then “weaved” together at run-time, by using an integration specification. In this paper, we describe how we extended these ideas to the user interface realm, showing that it is also possible to create concern-specific interfaces which are obviously composed according to integration specifications.

The rest of the paper is structured as follows: in Section 2 we present the background of our approach; next, in Section 3 we present an extension to the Abstract Data Views (ADV) interface design notation [7] used in OOHDM with ideas coming from aspect-oriented software design; we next introduce the idea of interface composition using transformations and illustrate our approach. In Section 4 we discuss some related work. Finally in Section 5 we conclude the paper and present some further work we are pursuing.

2 Integrating Volatile Functionality into OOHDM Models

Our approach is based on the idea that even the simplest volatile functionality (e.g. a video as in Figure 1), must be considered a first-class citizen and designed accordingly. Our extension to OOHDM can be summarized with the following design decisions, which are shown schematically in Figure 2:

- We decouple volatile from core functionality by introducing a model for volatile functionality (called Volatile Layer), which comprises both a conceptual and navigational models.
- New behaviors, i.e. those which belong to the volatile functionality layer are modeled as first class objects in the volatile conceptual model; they are considered as a combination of Commands and Decorators [9] of the core classes.
- As a consequence, we use inversion of control to achieve obliviousness; i.e. instead of making core conceptual classes aware of their new features, we invert the knowledge relationship. New classes know the base classes on top of which they are built. Core classes therefore have no knowledge about the additions.
- Nodes and links belonging to the volatile navigational model may or may not have links to the core navigational model. The core navigational model is also oblivious to the volatile navigational classes, i.e. there are no links or other references from the core to the volatile layer.
- We use a separate integration specification to specify the connection between core and volatile functionality.
- We design (and implement) the interfaces corresponding to each concern separately; the interface design of the core classes (described using ADVs [7]) are oblivious with respect to the interface of volatile concerns.
- Core and volatile interfaces (at the ADV and implementation levels) are woven by executing an integration specification, which is realized using XSL transformations.

In this section we focus only on conceptual and navigational issues while Section 3 describes user interface specification. The overall conceptual model of the application comprises two coarse grained packages, one containing the core functionality and the other which is itself composed of packages describing each volatile module. Classes

in the volatile layer have a knowledge relationship to those classes which are extended with the new services. The navigational model is built analogously; there are no links between core and volatile nodes, though there might be links to the core navigational model (indicated with a dashed line in the diagram).

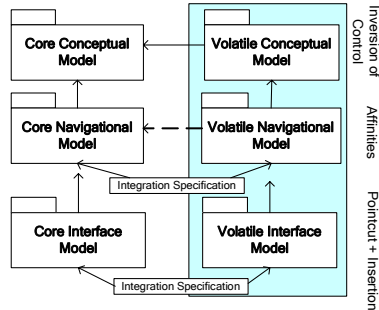


Fig. 2. Overall Schema for Volatile Functionality

Nodes comprising volatile functionality are woven onto the core model using an integration specification, which is decoupled both from volatile and core classes, thus making the nodes oblivious to the integration strategy. In this way, the same volatile functionality can be attached to different nodes at different times according to the application's needs. The specification indicates the nodes that will be enhanced with the volatile functionality, and the way in which the navigation model will be extended. For example we can add links, insert new information or components in a node, etc. The nodes which are affected by the new functionality are called the affinity of the functionality.

In Figure 3, we present an oversimplified conceptual and navigational diagram, showing a couple of outstanding classes and node classes in the CD example; the intention of Figure 3 is to serve as an anchor for further examples in Section 3. More elaborated examples of volatile functionality and integration specifications can be found in [19]. The integration specification for the video functionality is the following:

```
Affinity NorahJones (FROM FishbowlNode WHERE performer=Norah Jones)
FROM CDNode WHERE title = 'Not too Late'
      AND performer = 'Norah Jones'
Integration: Extension
```

This specification indicates that the nodes corresponding to the CD with title “Not too late” and performed by “Norah Jones” will be enriched with the Norah Jones video (as if new attributes were added). Meanwhile, the specification for the volatile St. Valentine's store is as follows:

```
Affinity Valentine
From CDNode and BookNode WHERE styleTag='Romantic'
Integration: Linkage (StValentineStore)
Additions: [Message: Text (“Find perfect Valentines for sweethearts...”)
ToStore: Anchor]
```

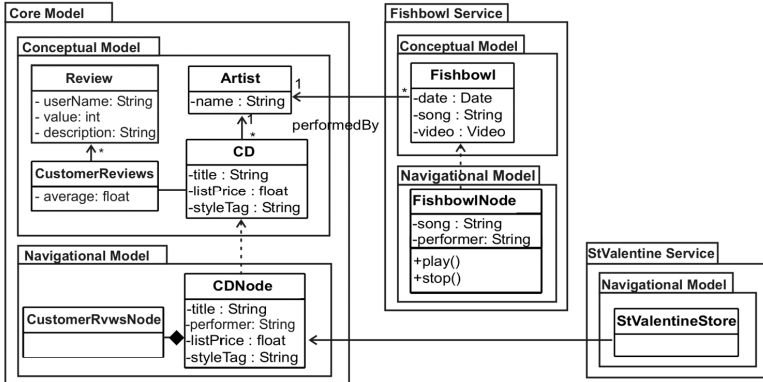


Fig. 3. Core and Volatile Conceptual and Navigational Models

In this case we wish to add a link towards the singleton StValentineStore to those products (CDs and Books) which are tagged as “romantic”. The text message is indicated in the specification and the Anchor is associated to the unnamed link created at weaving time. Notice that in both cases the interface specification (and therefore some aspects of the implementation) also should change.

Our notation which is similar to the OOHDM node definition syntax (based on object queries) allows expressing different affinities for the same volatile concern, therefore allowing great flexibility in the resulting nodes without polluting core classes. For example, once the CD is not more a novelty, we can weave the video to another node, if necessary, with a different specification; in Amazon, particularly, some of these videos are re-located (during a period of time) in the home page.

In order to support our model-driven approach and to simplify the process of weaving volatile functionality into core nodes, we have implemented a framework (CAZON), on top of Apache Struts. CAZON supports semi-automatic translation of core and volatile OOHDM models into XML specifications, and manages the instantiation of Web pages from the OOHDM navigational schema; the integration between core and volatile services is performed by executing the queries which specify the service affinities. A service manager evaluates affinity queries (stored in XML files) on the actual node which is being built, and when a query succeeds, it augments the node (in fact the corresponding XML description) with the attributes, links or aggregated nodes indicated in the specification. As a result of a request CAZON returns a node which now contains the corresponding volatile functionality and whose presentation is handled using the base Struts mechanisms and tools. A full description of CAZON can be found in [19].

In the following sections we explain how we managed to apply the ideas of oblivious composition of volatile features in the user interface, both at the design and implementation levels. Though we focus on our implementation in CAZON, most of the concepts can be easily applied in a broader context. Particularly, the idea of separation and oblivious composition of user interfaces is applicable to all kinds of design concerns though we exemplify it with volatile ones.

3 Improving Web Interface Composition

User interfaces also suffer the impact of the addition and editing of volatile functionality both at design and implementation levels. Even if we push languages like JSP to their limits regarding modularity, it is practically inevitable that code which describes the interface of core components is polluted with tags belonging to volatile functionality and therefore both concerns (core and volatile) get tangled.

As an example, the JSP page that implements the CD interface (see Figure 1) will have knowledge on both St. Valentine store and on the Fishbowl component as shown in Figure 4.

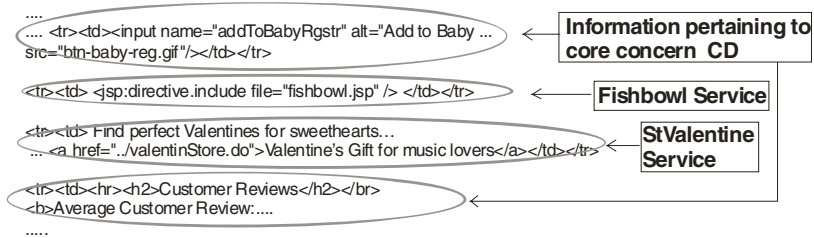


Fig. 4. Code tangling in the user interface

In Figure 4, we can see two blocks of code which refer to the Fishbowl and St Valentine concerns and which are clearly tangled with the code of the core (CD) concern therefore compromising modularity. A similar situation would arise at the interface design level regardless of the used notation. The classes corresponding to the core's interface will have explicit references (either as attributes or as aggregated classes) to the volatile ones.

It is important to state that this problem is conceptual and not technological. In common implementation technologies (such as JSP, JSF, etc.) or user interface description languages (such as usiXML [15], UIML [18], etc) we may experience this kind of tangling due to the lack of primitive constructs to implement either a solution based on polymorphism (e.g., using Decorators to make oblivious interface code insertion) or on point-cuts such as in aspect-orientation. The *include* primitive, typical of scripting languages, does not guarantee obliviousness because it yields an explicit invocation.

It is not surprising that separation of concerns is such elusive at the interface level; most modern Web engineering techniques have treated these aspects as lower-level issues (e.g. relegating it to the implementation stage). Avoiding tangling at the interface level allows better composition of existing interfaces and interface designs. Though this impact is obvious and seems more harmful at the code level (e.g. in a JSP page) it should be also addressed at design time. For the sake of understanding we describe our approach in two different sub-sections, addressing design issues first, and then showing how we realized these ideas in the implementation stage in the context of CAZON.

3.1 Composing Web Interface Designs

In OOHDM, the user interface is specified using Abstract Data Views (ADV) [7], which support an object-oriented model for interface objects. In OOHDM we define an ADV for each node class, indicating how each node’s attribute or sub-node (if it is a composite node) will be perceived. An ADV can be seen as an Observer [9] of the node, expressing its perception properties, in general as nested ADVs or primitive types (e.g. buttons). Using a configuration diagram [21] we express how these properties relate with the node’s attributes.

ADV’s are also used to indicate how interaction will proceed and which interface effects take place as the result of user-generated events. These behavioral aspects, which are specified using ADV-charts [7] (a kind of Statechart), are outside the scope of the paper; we will only focus on structural interface aspects.

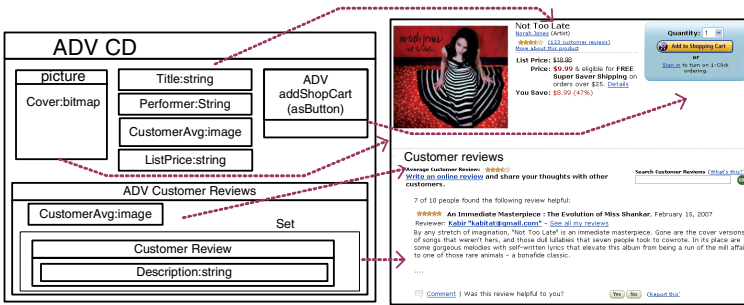


Fig. 5. The ADV corresponding to the CD Node

We have slightly modified the ADV notation in such a way that the positions of nested objects in the ADV reflect the look and feel of the interface as shown in Figure 5. This notation which is inspired in a similar one for UWE [12], allows improving discussions with different stakeholders, though it can not be processed automatically by standard ADV-based tools.

As explained in Section 2, each concern (core and volatile) will comprise ADVs for its corresponding nodes; when necessary, e.g. when a node should exhibit some volatile functionality, we weave volatile and core ADVs using an integration specification. Figure 6 shows these ideas schematically.

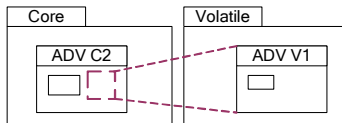


Fig. 6. ADV V1 woven into ADV C2

To express the integration, we have defined a simple specification language which allows indicating pointcuts and insertions at the abstract interface level, i.e. the position of the volatile ADV when it is inserted in the core ADV. The specification

generalizes the idea of pointcuts to the two dimensional space of Web interfaces. A pointcut and the corresponding insertion are specified using the following template:

IntegrationFor: *Concern name. affinity name*
 Target: *ADV target name*
 Add: *ADV source name | Insertion Specification*
 Relative to: *ADV name*
 Position: [*above | bottom | left | right*]

The field “IntegrationFor” refers to the navigational affinity as described in Section 2; the name of the affinity is necessary only when there is more than one affinity in the same concern. When the affinity is satisfied (at the navigational level) the interfaces must be composed according to the specification. The field “Target” indicates the name of the ADV (or ADVs) which will host the volatile interface code. Inner ADVs may be specified using a “.” notation, such as CD.Reviews to indicate that the insertion will take place in the ADV Reviews, which is a part of the ADV CD.

The “Add” field indicates which elements must be inserted in the target, either an ADV or an immediate specification, which is used when the inserted field is simple enough to avoid the specification of another (auxiliary) ADV. Finally we indicate the insertion position by using the “Relative” and “Position” fields. Notice that the specification is still “abstract”, leaving place to fine tuning during implementation.

In Figure 7 we show (on the left) the ADV for the Fishbowl volatile functionality and the integration specification which corresponds to the abstract interface of Figure 5. The result of the weaving process in the concrete interface is shown on the right of Figure 7. As shown, the ADV Fishbowl is placed between the CD’s core information and the inner ADV Customer Review.

Sometimes the integration requires additional interface objects, for example when the navigation extension is of type *Linkage*, as in the case of the St. Valentine store. The corresponding objects might be defined either as ADVs belonging to the specific volatile concern package (e.g. for reusing them in other specifications), indicated in the integration specification (e.g. when they are simply strings), or separately defined as “integrators” ADVs (e.g. when only used for this particular specification).

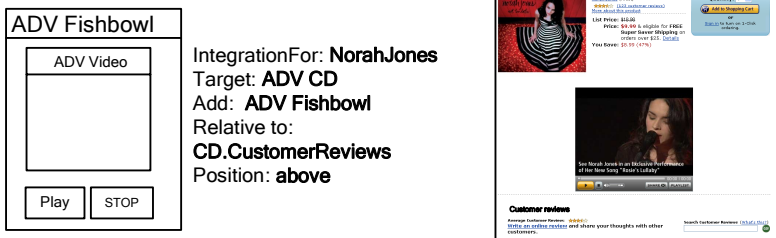


Fig. 7. Integrating Fishbowl onto the ADV CD

In Figure 8 (left), we show the ADV Reminder which is used during the integration process of the Valentine’s store. This ADV does not need to have an underlying navigational node and it provides an anchor to the St. Valentine’s store which is realized

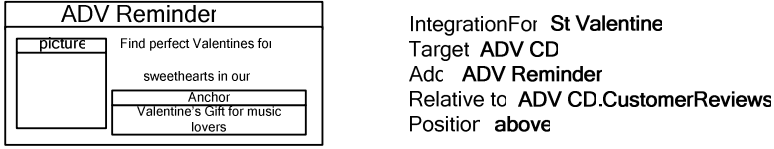


Fig. 8. ADV Reminder and integration specification

in the CD node after the navigational weaving. The result of weaving the ADV Reminder into the ADV CD according to the specification in the right of Figure 8, gives as a result the ADV in Figure 9 (left) with a concrete interface shown in Figure 9 (right).

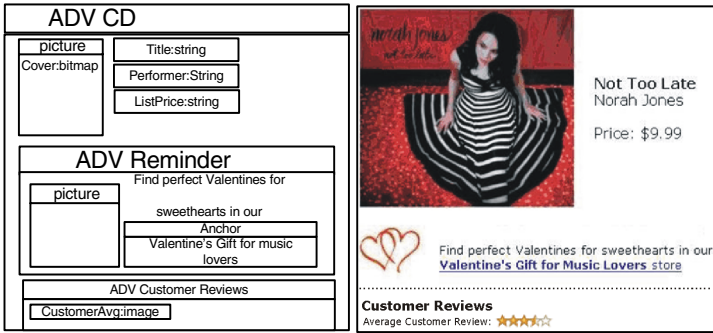


Fig. 9. ADV CD after weaving and the resulting concrete interface

A further research subject (see Section 5) is to analyze the impact of the order in which integrations are specified (in fact “executed”) in the interface look and feel, for example in the case of the volatile concerns Fishbowl and St. Valentine.

As shown in [21], ADVs can be mapped systematically into concrete interface specifications in different running environments. Next, we show our approach to achieve obliviousness of interface code in the implementation stage.

3.2 Using Transformations to Compose XML Documents

The problem of achieving obliviousness at the user interface can be expressed in terms of XML documents as follows: given two documents A and B which express the contents of a node, we need to describe how to obtain a document which integrates B into (an specific part of) A, without an explicit reference inside A. Moreover, in the case of (irregular) volatile functionality, we need that this integration is done in all documents which fulfill some conditions; this might eventually involve specific instances of different document types.

The core of our solution is to use XSL [25] transformations to compose volatile and core interfaces, and XPATH [24] to indicate the parts of the source document in which the insertions are done. The transformation acts as an aspect in aspect-orientation: the

content of a template is like an advice, while the XPATH specification which matches the template, indicates the point-cut where the advice is inserted. An XSL engine (e.g., Xalan[23], Saxon[20]) does the weaving process.

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://..." xmlns:jsp="http://..." >
  <!--Imports a transformation which copies all the elements-->
  <xsl:import href="defaultTemplate.xsl"/>
  <xsl:template match="//tr[contains(.,'Customer Reviews')]"/>
    <tr><td> <jsp:directive.include file="fishbowl.jsp" /></td></tr>
    <tr><xsl:apply-templates select="*" /></tr>
  </xsl:template>
</xsl:stylesheet>

```

Fig. 10. Transformation that inserts Fishbowl component

For example, in order to add the Fishbowl component to the CD Node, we can apply the XSL transformation shown in Figure 10 over the CD interface. The XPATH expression (point-cut) `"//tr[contains(.,'Customer Reviews')]"` refers to the row containing the text "Customer Reviews" (see Figure 4). The template (advice) leaves the existing elements of that row unchanged and inserts above a new one with the Fishbowl element.

In the case of class-based volatile functionality (e.g. functionality which applies to all instances of a class), and given that JSP pages can be written as well-formed XML documents, this kind of transformations could be applied statically to incorporate the tags with volatile functionality without polluting the source code (see Figure 11).

A problem with this simple solution is that when volatile functionality only affects some instances of a class (e.g., those products which are recommended to be St. Valentine presents), some kind of conditional structure should be included in the tags, polluting the resulting code. As applying XSLT transformations in bare JSP during run time is cumbersome, we decided to use a flexible approach using a more "pure" XML-based framework in which the publishing process is done by applying XSL style sheets to the XML content. We describe our approach in the following sub-section.

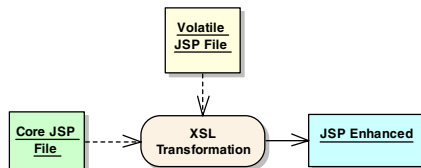


Fig. 11. Statically weaving of volatile interfaces

3.3 Our Approach in a Nutshell

In CAZON for each node type we define a style sheet which transforms its XML representation in a physical presentation object (e.g., HTML, WAP, etc). When the node contains aggregated sub-nodes, the style sheet contains calls to the style sheets

templates corresponding to the nested parts. As explained before, when the framework receives a request to perform an action, it produces as a result a node's instance, described with an XML document containing its attributes, anchors and its recursively aggregated nodes. The presentation layer gets this document and transforms it according to the corresponding style sheet. As an example, in Figure 12 (left) we show the style sheet associated to the CDNode type; when applied to the XML node representation corresponding to the Norah Jones CD (Figure 12 right) it yields a concrete interface, like the one in Figure 9, but without the reference to the St. Valentine's store.

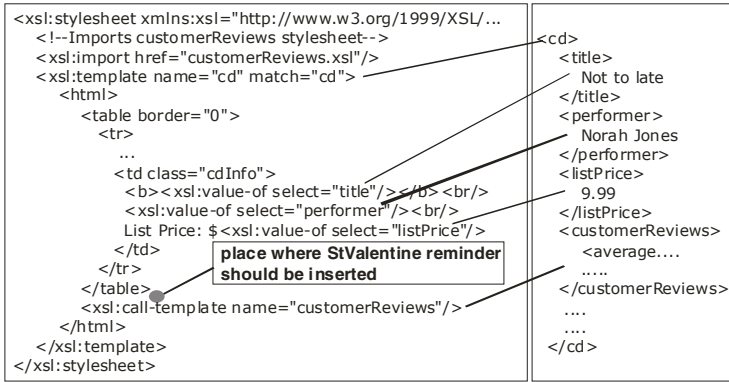


Fig. 12. CD stylesheet and CDNode instance

As explained in Section 2, when a node's instance satisfies an affinity query corresponding to a volatile service, the corresponding node (in fact its XML representation) is augmented according to the indication of the integration specification, either with a link, attributes or nested nodes. Therefore, to complete the task in the user interface and according to Section 3.2, for each integration specification we need to implement a XSL transformation which applied over the style sheet corresponding to the core node, inserts the newly added elements. The transformation associated with the St. Valentine' Reminder integration specification (Figure 8) is shown in Figure 13.

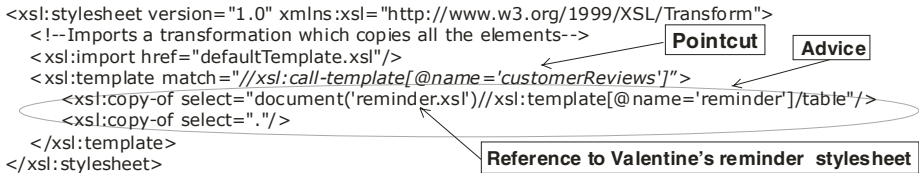


Fig. 13. Transformation that inserts Valentine's reminder

We have realized these ideas in CAZON using Stxx [22], an extension of Struts which allows action classes to return XML data documents, to be transformed using different technologies into appropriated presentation formats.

To allow dynamic weaving of style sheets during run time, we created a sub-class of the Stxx AbstractXSLTransformer such that the method *transform()* collaborates

with CAZON's ServiceManager to get the list of volatile services which had affinities with the actual node. For each of these volatile services, its associated integration transformation is obtained and applied over the base interface style sheet. Finally, the transformed style sheet is applied to its XML document (which has been previously augmented) and the final user interface is obtained, as shown in Figure 9 (right).

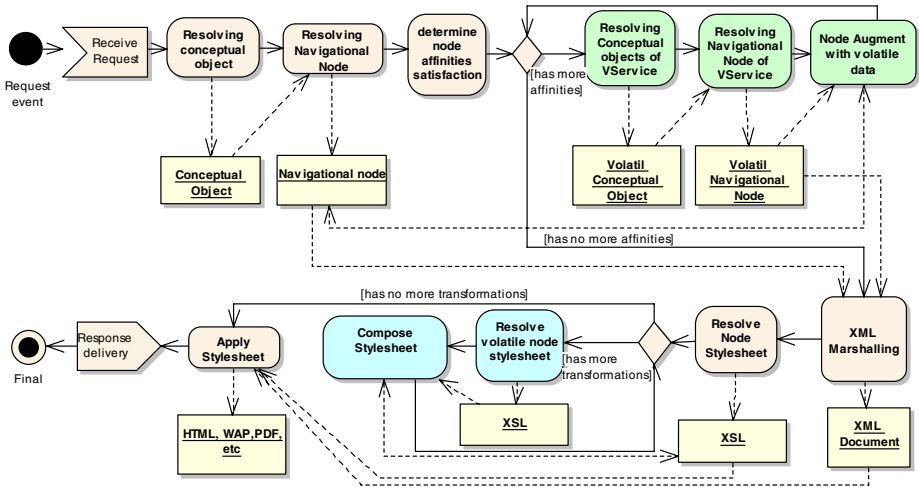


Fig. 14. Overview of request processing in CAZON

In figure 14 we show a UML activity diagram which summarizes the core steps of our implementation.

We have finally defined a set of heuristics to allow semi-automatic translation of interface integration definitions (as those presented in Section 3.1) to XSL templates and XPATH specifications, assuming that the XSL style sheets which correspond to the ADVs have been manually codified.

4 Related Work

Volatile Requirements have been a research focus in the requirement engineering community for some years. In [16], the authors present an aspect-oriented approach for representing and composing volatile requirements using composition patterns [5]. Though we deal with the problem on another level of abstraction (design and implementation), our approach complements the ideas in [16] for the kind of volatile services which are usual in the Web. We have also got inspiration from the so called symmetric separation of concerns approaches, such as Theme/Doc [4]. Both volatile and core models are modeled using the same concepts, while in asymmetric approaches crosscutting concerns are modeled with a different primitive: the aspect.

Modern Web design methods have already recognized the importance of advanced separation of concerns for solving the problem of design (and code) tangling and scattering. For example, in [2] aspect oriented concepts are used to deal with adaptivity.

Though we don't use aspects to deal with volatile functionality, the approaches are somewhat equivalent.

In [12] the authors present an XML Publication Framework based on the UWE approach; the transformation concept presented in this paper can be used to extend the framework of [12] to allow volatile concerns. One just needs to append transformations steps into the Cocoon pipeline.

Composition of interfaces has also been addressed in [14] by using operators of the tree algebra with the UsiXML description language [15]. A visual tool (ComposiXML) has been implemented with these ideas. The aim of that research is to help in the reuse of interface components. Our proposal uses affinity specifications and XSL transformations instead of tree algebra, focusing on oblivious integration of core and volatile interfaces.

So far, we are not aware of any approach supporting oblivious composition of interface design models; meanwhile, in the XML field, the AspectXML project [1] has ported some concepts of aspect-orientation to XML technology, by allowing the specification of point-cut and advices similarly to Aspect Java. The project is still in a research stage.

In [6] a J2EE framework (named AspectJ2EE) which incorporates aspects on EJB components is presented. AspectJ2EE may be used to incorporate volatile concerns on J2EE applications at the model layer, though navigational and interface aspects are not mentioned in the project. The aspect weaving is performed at the deployment stage, while we propose to perform it in runtime in order to deal with volatile functionality that only affects some instances of a class dynamically.

5 Concluding Remarks and Further Work

We have presented an original approach for seamless and oblivious composition of Web applications' interfaces. Our approach is grounded on the well-known principles of advanced separation of concerns to improve modularity and therefore to foster reuse and software evolution. We have focused on one specific kind of application's concerns: those which encompass volatile functionality, i.e. the kind of functionality that can not be guaranteed to be stable. Using the compositional approach that we described in the paper, conceptual navigation, and user interface models corresponding to core concerns can be made oblivious to the models corresponding to volatile requirements, which are designed using the same primitives (in our approach, those in the OOHDM design framework). By using a very simple syntax, we indicate the way in which interface designs are composed, and using XSL transformations we are able to weave the corresponding XML files. In this way we don't need to pollute the design model and the implementation code with references to components that may be eliminated (requiring newer code editions). We have realized these ideas in the context of CAZON, an OOHDM-based framework which automates dynamic weaving of volatile functionality into core application's modules, both at the navigation and interface levels. The ideas in this paper can be used for weaving any kind of concern into the application's core in those cases in which we want both (the concern and the core functionality) to evolve separately and obliviously.

We are now working on several research areas: first we are building tools to automate the translation of ADVs into XSL files, and point-cut specifications into XSL transformations to improve model-driven support in CAZON. We are also analyzing other kinds of concerns (either volatile or not) in which interface weaving might be crosscutting, i.e. involving further changes in the core interface. Though XSL transformations can cope with this situation, we aim to improve our specification language to support more complex crosscutting. We are also studying the problem of conflicts among volatile models, and the impact which the order of execution of integration specifications has in the interface look and feel; this problem is similar to the problem of conflicts among aspects already reported in [8]. Finally, we are researching on the process of building prototypes from requirement specifications which encompass separated concerns using early aspects approaches such as [10]. A further topic not addressed in this paper is the application of these ideas in the field of RIA (rich internet applications), particularly those built using Ajax or similar scripting languages.

References

1. AspectXML: The AspectXML home page. In www.aspectxml.org
2. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling Adaptivity with Aspects. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, Springer, Heidelberg (2005)
3. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. *Computer Networks and ISDN Systems* 33(1-6), 137–157 (2000)
4. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design. The Theme Approach. Addison-Wesley, Object Technology Series (2005)
5. Clarke, S., Walker, R.: Composition patterns: an approach to designing reusable aspects. In: Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, May 2001, pp. 5–14. ACM Press, New York (2001)
6. Cohen, T. (Yossi) Gil, J.: AspectJ2EE = AOP + J2EE Towards an Aspect Based, Programmable and Extensible Middleware Framework. In: Odersky, M. (ed.) ECOOP 2004. LNCS, vol. 3086, pp. 219–243. Springer, Heidelberg (2004)
7. Cowan, D., de Lucena Pereira, C.: Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse. *IEEE Trans. Software Eng.* 21(3), 229–243 (1995)
8. Filman, R., Elrad, T., Clarke, S., Aksit, M. (eds.): Aspect-Oriented Software Development. Addison-Wesley, London, UK (2004)
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of reusable object-oriented software. Addison Wesley, London, UK (1995)
10. Gordillo, S., Rossi, G., Moreira, A., Araujo, A., Vairetti, C., Urbietta, M.: Modeling and Composing Navigational Concerns in Web Applications. Requirements and Design Issues. LA-WEB, pp. 25–31 (2006)
11. Harrison, W., Ossher, H., Tarr, P.: General Composition of Software Artifacts. *Software Composition*, pp. 194–210 (2006)
12. Koch, N., Kraus, A., Hennicker, R.: The Authoring Process of UML-based Web Engineering Approach. In: Proceedings of the 1st International Workshop on Web-Oriented Software Construction (IWOST 02), Valencia, Spain, pp. 105–119 (2001)

13. Kraus, A., Koch, N.: Generation of Web Applications from UML Design Models using an XML Publishing Framework. In: Integrated Design and Process Technology Conference (IDPT'2002) (June 2002)
14. Lepreux, S., Vanderdonckt, J.: Towards Supporting User Interface Design by Composition Rules. In: Proceedings of CADUI'2006. Ch. 19, Springer, Berlin (2006)
15. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In: Proceedings of 9th IFIP Working Conference on EHCI-DSVIS'2004 (2004)
16. Moreira, A., Araujo, J., Whittle, J.: Modeling Volatile Concerns as Aspects. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, p. 544. Springer, Heidelberg (2006)
17. Pastor, O.: Abrahão, S., Fons, J.: An Object-Oriented Approach to Automate Web Applications Development. In: Proceedings of EC-Web, pp. 16–28 (2001)
18. Phanouriou, C.: UIML: A Device-Independent User Interface Markup Language. Ph.D. Thesis, Virginia University (2000)
19. Rossi, G., Nieto, A., Mengoni, L., Lofeudo, N., Distanto, D.: Model-Based Design of Volatile Functionality in Web Applications. Proceedings of LA-WEB 2006, Mexico 2006, pp. 179–188. IEEE Press, Orlando, Florida, USA (2006)
20. Saxon: (2007), <http://saxon.sourceforge.net/>
21. Schwabe, D., Rossi, G.: An object-oriented approach to web-based application design. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet 4, 207–225 (1998)
22. Stxx. Struts for Transforming XML with XSL (2005) <http://stxx.sourceforge.net>
23. Xalan: (2007) <http://xalan.apache.org/>
24. XPATH. XML Path Language: (2007) <http://www.w3.org/TR/xpath>
25. XSL: The Extensible Stylesheet Language Family (2007) In <http://www.w3.org/Style/XSL/>

Fine-Grained Specification and Control of Data Flows in Web-Based User Interfaces

Matthias Book, Volker Gruhn, and Jan Richter

Chair of Applied Telematics/e-Business, University of Leipzig
Klostergasse 3, 04109 Leipzig, Germany

{book,gruhn}@ebus.informatik.uni-leipzig.de, jan.richter@saxess.ag

Abstract. When building process-intensive web applications, developers typically spend considerable effort on the exchange of specific data entities between specific web pages and operations under specific conditions, as called for by business requirements. Since the WWW infrastructure provides only very coarse data exchange mechanisms, we introduce a notation for the design of fine-grained conditional data flows between user interface components. These specifications can be interpreted by a data flow controller that automatically provides the data entities to the specified receivers at run-time, relieving developers of the need to implement user interface data flows manually.

1 Introduction

Web-based user interfaces have become popular front-ends for information systems that require convenient access at any time from anywhere [1]. Especially in business-to-business and intranet applications that are designed to support elaborate business processes, these user interfaces can turn out to be quite complex. Their complexity is typically twofold: Most obviously to the user, they have intricate dialog structures that include nested dialog sequences, wizards, context-sensitive links and other navigation patterns. More transparent for the user, but all the more palpable for the developer, are the complex data flows between the interface and the business logic. No matter if these data flows mirror major business process features or serve minor technical purposes, the developer must ensure that the right data is available for the right component at the right time, all the while keeping an eye out for security issues, validity concerns, performance considerations and persistence strategies.

Despite some progress over the past years, the WWW infrastructure itself still provides only rudimentary data flow support: Originally, the only available data flow mechanism was the transmission of parameter strings from clients to servers in **HTTP requests**. Soon, web browsers became capable of receiving short **cookie** strings from a server and sending them back to the same server with every subsequent request. This in turn enabled servers to unambiguously associate multiple requests with the same client, and keep individual state information for all clients in **sessions**. Since sessions usually expire after some

time without user activity, **persistence** of selected data is typically ensured by integration of databases or other storage technologies in the back-end.

These basic data flow mechanisms are technically sufficient to build any web-based application: Requests provide a channel for data flows from client to server, sessions provide a scope for data flows among server components, and if necessary, cookies can provide an additional channel for client-server data exchange, client-side state-keeping, and even simple client-side persistence. However, these mechanisms are only convenient for a small subset of conceivable data flows: Requests are ideal for sending data from a web page to those business operations responsible for building the server's response; and sessions are ideal for making data accessible throughout the application. These alternatives mark two ends of a spectrum – in our experience, however, most data flows described by business processes lie somewhere in the middle: Often, data generated in a certain process step is intended only for a clearly defined, but not necessarily immediately following set of pages and/or operations. Thus, the respective data flows must reach beyond a single request-response cycle, but do not require session-wide data publication.

When mapping such process requirements to the technical level, developers currently need to choose the lesser of two evils: Passing data along a chain of requests toward its ultimate destination violates the principle of encapsulation, as the intermediate pages and operations need to handle data that they are not actually responsible for – an unclean and error-prone solution that requires high implementation and maintenance effort. Alternatively, storing data in the session for use at a later time relieves the intermediate steps from a lot of hassle, but bears the danger of memory leaks if the data is not removed from the session when it is no longer required. In addition, both approaches pose inherent security risks as data is exposed to pages or operations that do not need to know about it (and in the first case, even repeatedly sent over the network). Even if other application components are considered “friendly”, this unnecessary exposure multiplies the possible points of failure or attack.

Since neither the request nor the session scope provides satisfactory dialog flow support, we propose a supplemental method for realizing data flows that match the process requirements of application domains more closely, while at the same time reducing the required implementation effort. In this paper, we will first show how arbitrary point-to-point data flows can be specified in a graphical notation (Sect. 2). We then show how this specification can be interpreted at run-time by a data flow controller that provides just the specified data to each page and operation (Sect. 3). We conclude with a discussion of related work (Sect. 4) and an overview of further research opportunities (Sect. 5).

2 Data Flow Specification

In the introduction, we discussed data flows related to the user interface layer. Obviously, data also flows along the control structures within the application logic, may be exchanged with third-party, legacy or back-end systems, and can

be stored in and retrieved from persistent memory. However, since these data flows are the responsibility of the application and persistence logic, established specification methods (e.g. UML data flow diagrams, sequence diagrams etc.) and implementation techniques (e.g. SOAP, EJB etc.) can be used for these layers. Our work instead focuses on data flows between the presentation and application logic, where suitable implementation techniques (apart from the coarse request and session scopes) and specification methods have not yet been widely established.

To specify user interface data flows on a more fine-grained level than the one provided by the request or session scopes, we first need to define the concept of a data flow more clearly. Given a data source A , a data sink B and a data entity d , we define that the data flow of d from A to B is the provision to B of the d available to A (i.e. B gets to know the d that is known to A). As we will soon see, it is helpful to make data flows conditional, i.e. to execute them only if a certain constraint is fulfilled.

In designing a notation that maps this abstract definition onto a technical level, we need to answer a number of questions:

- What are concrete data sources and data sinks?
- Which constraints determine if a data flow is executed?
- How are sources and sinks related?
- What are concrete data entities?
- How is the “provision” of data entities realized?

The first three questions relate to the specification of data flows, and will be answered in the following subsections. The last two questions relate to the implementation of data flows, which will be discussed in Sect. 3.

2.1 Data Sources and Sinks

The data sources and data sinks used to model data flows should be entities that represent the structure of web-based user interfaces. In our past work, we have found it natural to distinguish between web pages and application logic operations in order to model the navigation structure of web applications: In the Dialog Flow Notation (DFN) [2], web pages are symbolized as dog-eared sheets (the so-called **masks**), while application logic operations (**actions**) are symbolized by circles. To specify all possible navigation paths, these elements are connected by arrows symbolizing **dialog events** generated by masks or actions when the user submits a request, or a business operation produces a result. In the dialog graphs specified this way, masks and actions do not have to alternate since it is conceivable for a mask to trigger a succession of several server-side operations, and also conceivable for a mask to link directly to another mask without the need for any intermediary business operations. The dialog graphs are encapsulated in **dialog modules** that can call each other at run-time to form hierarchically nested dialog structures.

We found that the DFN provides an ideal basis for specifying the data flows within a web-based user interface: The masks, actions and modules can serve as

data sources and sinks, while the dialog events can be interpreted as constraints that determine when a data flow is executed. In the following subsections, we will show how the original DFN was extended to specify different kinds of data flows. Our aim was to extend the notation in such a way that the data flows are intuitively readable and conceptually integrated with the DFN semantics, but do not add unnecessary optical clutter to the diagrams.

2.2 Data Flow Types

Depending on the types of the data sources and sinks, and their proximity within the topology of the dialog graph, we distinguish several kinds of data flows:

Parallel Data Flows. The simplest type of data flow is running in parallel with the dialog flow, i.e. a data flow from the generator of a dialog event to the receiver of the same event. It is specified in the DFN by adding the label of the data entity in square brackets to the event’s name (if the data flow shall comprise several data entities, we separate them with commas). For example, in the *login* module in Fig. 1, the *name* and *passwd* data entities entered by the user on the *login* mask shall only be provided to the *check name, passwd* action upon generation of the *submit* event. At first sight, this data flow may look equivalent to the request scope, but it is actually more finely grained: While the request scope always encompasses all intermediate actions up to the next mask, the parallel data flow is restricted to the current dialog event’s receiver only. Therefore, the *passwd* data in the example is only provided to the succeeding action, but not accessible beyond it (as it would be in the request scope).

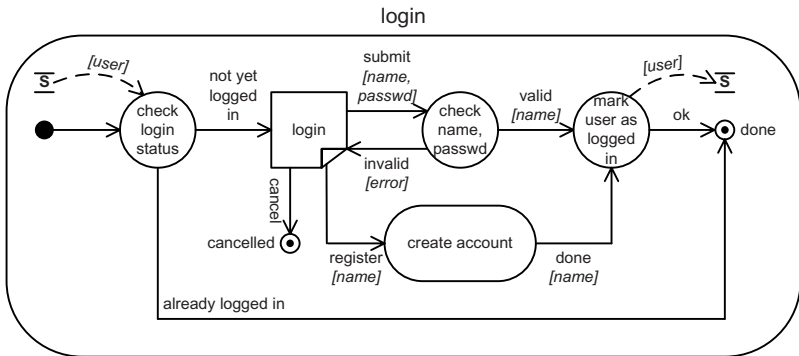


Fig. 1. Login module performing user authentication

Request scopes and parallel data flows also differ in their starting points: While a request can only be generated by a mask, a dialog event with associated parallel dialog flow can also be generated by actions or modules. For example, if the login credential check fails, the *error* data is provided by the *check name, passwd* action to the *login* mask with the *invalid* event.

Finally, parallel data flows can be used more flexibly than request scopes since developers can specify which data is provided with which events: Using only the request scope, the developer may not be able to prevent that the credentials entered by the user on the *login* mask are provided to subsequent elements both for the *submit* and (unnecessarily) the *cancel* event. Using parallel data flows, however, the developer can specify that the *name* and *passwd* data is provided to the subsequent action with the *submit* event, just the *name* data is provided to the *create account* module with the *register* event, but no data is provided with the *cancel* event.¹

Divergent Data Flows. As we have just seen, parallel data flows enable developers to specify which data entities shall be provided to the receivers of which events. Often, however, the data generated by one element should not be provided to any of its immediate successors, but rather to some more distant element in the dialog graph that is responsible for the actual data processing.

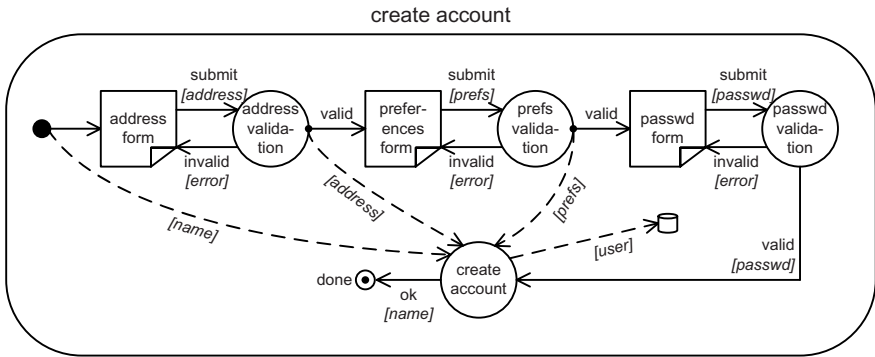


Fig. 2. *Create account* module employing a “wizard” navigation pattern for gathering user input

As an example, consider the “wizard” navigation pattern of the *create account* module in Fig. 2. Here, we prompt the user for his address, preferences and password, validate each of the inputs, and finally create an account from all the collected data. In order not to burden the various forms and validation actions with forwarding the data of their predecessors, we would like to provide the data gathered in each step directly to the final processing action. For the validation actions, the outgoing dialog and data flows therefore diverge: For example, if the *address validation* action produces a *valid* event, the dialog flow shall continue to the next wizard step, i.e. the *preferences form*, but the *address* data shall be provided directly to the *create account* action, bypassing the other steps of the

¹ Note that these restrictions cannot prevent the respective data from being transmitted across the network in the request, as this would require client-side logic. However, the data flow controller will ensure that any submitted data is only available to the specified receivers on the server side.

wizard. To specify that the data flow diverges from the dialog flow in this case, we draw a dashed line to the element that the data shall be provided to, and annotate it with the data entity's label in square brackets.

Just like a parallel data flow is always tied to a dialog event, a divergent data flow is also always associated with a particular dialog event and only executed if that event is traversed. To specify this constraint in the DFN, the data flow arrow must always begin in the same spot (marked by a black dot) as the event arrow that it shall be associated with. To maintain consistency with parallel data flows, which are always implicitly associated with an event, the DFN does not allow the specification of divergent data flows without an associated event.

Inter-module Data Flows. Of course, data does not only flow among the dialog elements within a module, but also between modules. Therefore, when a module is called, it should be able to accept data provided to it, and when a module terminates, it should be able to provide data to subsequent dialog elements. To express this behavior in the DFN, compatible data flows must be specified both in the exterior dialog graph that a module is embedded in, and in the interior dialog graph of the module itself: In the exterior dialog graph, modules can simply serve as sources and sinks of parallel and divergent dialog flows just like masks and actions. In Fig. 11, for example, the *create account* submodule is provided *name* data with the incoming *register* event and provides *name* data with its outgoing *done* event.

To specify which data entities a module can accept and provide, the initial and terminal anchors of its interior dialog graph serve as data flow interfaces: Parallel and divergent data flows can originate from the **initial anchor** (the black disk marking the starting point of the dialog graph traversal) to specify to which elements the incoming data shall be provided. Analogously, parallel and divergent data flows can lead to the module's **terminal anchors** (the circled small black dots marking the end of the dialog graph's traversal). They specify which data entities the module will provide to its exterior dialog graph, and who provides those data entities internally. In the definition of the *create account* module in Fig. 12, for example, the incoming *name* data flows directly from the initial anchor to the *create account* action, which will process it later together with the other data collected by the wizard. The *name* data will then flow from the *create account* action to the *done* terminal anchor, so it can be provided to the module's successor in the exterior dialog graph upon its termination.

Through these incoming and outgoing interfaces, data can be flexibly passed from several sources outside a module to several sinks inside a module and vice versa. Developers need to ensure that the data flows specified in the interior and exterior dialog graphs match, as unmatched entities will otherwise be unavailable to their sinks.

2.3 Shared Scope Access

Parallel and divergent data flows enable developers to define the propagation of data entities on a very fine-grained level. However, their specification may

become cumbersome if the same data entities shall be provided to many elements, as a profusion of explicit data flow arrows would be required. For this reason, the DFN provides constructs to symbolize data exchange through various shared scopes.

Module Scope. Often, the same data entities should be provided to virtually all elements within the same module. As we mentioned earlier, storing such data entities in the session scope is not an optimum solution for this problem since the developer would be responsible for removing them from the session before the module is terminated. Instead, it would be preferable if all elements within a module had a shared scope that is cleared out automatically once the module terminates. Our Dialog Control Framework (DCF) described in Sect. 3 provides the **module scope** for this purpose: It is comparable to the session scope in that it is associated with the current user, but exists only during the traversal of the current module. An empty module scope is instantiated whenever a module is entered, and discarded again once a terminal anchor has been reached.

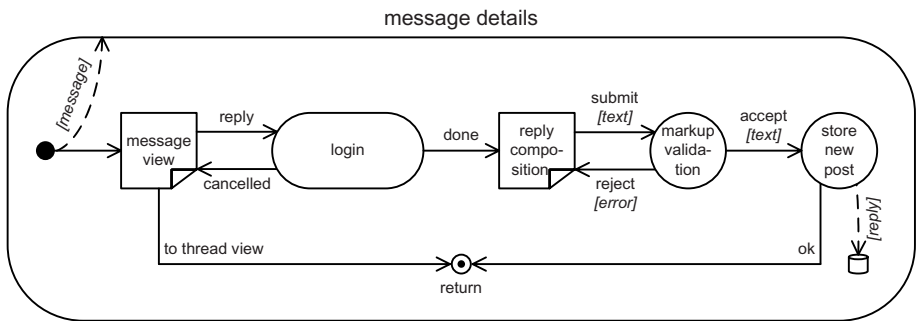


Fig. 3. *Message details* module for viewing and replying to messages in a discussion forum

To specify in the DFN that a data entity should be provided through the module scope, developers can simply draw a divergent data flow with the respective entity label in square brackets from any dialog event to the module’s contour. In the *message details* module in Fig. 3, for example, the incoming *message* data is provided to all other elements through the module scope right away.

Of course, data flows to the contour can also originate from any other source, indicating that the respective data will be available throughout the module as soon as the associated event is traversed. Since all data in the module scope is implicitly available to all dialog elements, no explicit notation construct is required to indicate that an element accesses data from the module scope. In the *message details* module, for example, all masks and actions can access the original *message* data in the module scope to display it, quote it in the text of the reply, and reference it with the new post.

In the DCF implementation, module scopes are stacked to reflect the nested module call structure: When a module calls a sub-module, a fresh module scope for the sub-module is pushed onto the stack, rendering the calling module's scope temporarily inaccessible. When the sub-module terminates, its module scope is removed from the stack, and the calling module's scope becomes available again.

Session, Application and Cookie Scope; Storage Access. The mechanisms described so far are helpful in many situations where the data scopes provided by the application server are too coarse for the data flow requirements at hand. However, there are obviously also a number of situations where those more generous scopes are the perfect choice – information about the user's login status, for example, should be available throughout the application and is therefore ideally stored in the session scope. Other data may best be globally provided through the application or cookie scope, and business objects often need to be stored in or retrieved from persistence storage by the application logic.

The data flows for these scenarios usually concern only the application (and possibly persistence) layer, so the dialog control logic that couples the presentation and application layer should not be involved in them. Consequently, the DCF does not handle data flows through these larger scopes, but lets the business logic interact directly with the application server or persistence layer API. This restriction of responsibilities allows for a clean separation of concerns on an architectural level, and allows developers to choose whichever scope and persistence framework is suited best to their needs, instead of being bound to the DCF's data flow mechanisms throughout the application.

Due to the framework's limitation to user interface management, the DFN technically would not need to provide additional constructs related to the larger scopes. Intuitively, however, developers will expect to be able to specify not only fine-grained data flows through the user interface, but also data shared through the more coarse scopes.

The DFN solves this dilemma by providing a compromise – developers cannot specify the complete application logic's internal data flows within the dialog flow, but the DFN enables them to specify where the dialog flow interfaces with the larger scopes: Data flow arrows leading to or from the letters **S**, **A** or **C** enclosed by horizontal bars indicate that the respective data entities are stored in or retrieved from the session, cookie or application scope, respectively. For example, in Fig. 11 the *check login status* action retrieves *user* data from the session scope, and the *mark user as logged in* action stores *user* data in the session scope. In addition, data flow arrows leading to or from a “can” symbol indicate that the respective data entity is stored in or retrieved from persistent storage – for example, the *user* data constructed by the *create account* action in Fig. 12 is sent to the back-end for storage. Note that in contrast to divergent data flows, the data flows leading to those larger scopes cannot share their starting point with an event, since the respective data handling operations are beyond the control of the dialog controller that is only aware of dialog events.

Since the DCF does not provide mechanisms for handling these scopes, the corresponding notation constructs have only illustrative character – the actual

data handling will have to be implemented manually by the developer. In contrast, the module scope and parallel, divergent and inter-module data flows are executable specifications: The developer only needs to specify the desired behavior, but does not need to implement it since the framework handles the respective data flows automatically.

2.4 Summary

With the traditional scopes, the module scope and the new finely-grained data flows, developers now have a full spectrum of data flow mechanisms at their disposal, as Fig. 4 illustrates: While the application and session scopes are suitable for publishing application-global and user-global data, parallel and divergent data flows are ideally suited for data propagation between arbitrary dialog elements. Since multiple parallel and divergent data flows can be associated with any dialog event, developers can control which data is propagated where under which conditions very flexibly.

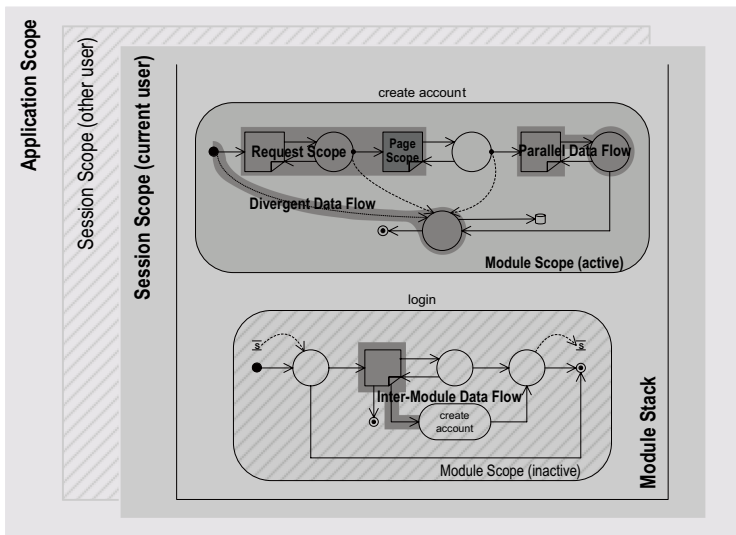


Fig. 4. Scopes and data flows supported by the Dialog Flow Notation and Dialog Control Framework

The page scope remains an important tool for data exchange between the components involved in constructing a response (e.g. a JavaServer Page (JSP) and its tag libraries). The request scope, however, has been rendered virtually redundant since its visibility is determined by purely technical criteria that only seldomly map exactly to business requirements for data propagation. This allows us to retrofit it for conveniently accessing the new data flow mechanisms, as we will show in the following section.

3 Data Flow Controller

Having introduced the new data flow constructs of the extended Dialog Flow Notation, we will now show how their semantics are supported by the Dialog Control Framework (DCF). Revisiting our initial questions from Sect. 2, we first need to address the technical representation of data entities, and discuss how the provision of those entities from sources to sinks is interpreted. After this, we will give a brief overview of the technical implementation of the data flow control mechanism within the DCF.

3.1 Data Entity Representation and Provision

In the previous sections, we identified data entities only by their label (e.g. “*user data*”). In concrete applications, a data entity can be any object – in Java-based applications, it will typically be an instance of a `JavaBean` holding various attributes, which is stored in or retrieved from a data scope using its label as a look-up key. In Java Enterprise Edition-compatible application servers, for example, `HttpSession` instances provide the `Object getAttribute(String name)` and `void setAttribute(String name, Object value)` methods for this purpose. Just like the DFN relies on existing notations to specify the layout of dialog masks (e.g. visual design sketches or XForms) or the control flow within actions (e.g. UML activity or state diagrams), it does not provide its own constructs for the specification of data entities’ internal structure. Instead, we recommend using existing notations such as entity-relationship or UML class diagrams for this purpose.

Since we assume that the data entities in our data flows are objects (i.e. reference types), we have two alternatives for interpreting the flow of data from a source to a sink: “providing a data entity” could mean

- forwarding the data entity itself from the source to the sink, or
- extending the data entity’s scope so it is not only available to the source, but also to the sink.

While these alternatives may at first sight look like equivalent implementation variants, they exhibit different behavior if a source provides the same data entity to several sinks, as illustrated in Fig. 5. In a situation like this, the first approach (forwarding the data entity) intuitively implies that both *B* and *C* receive identical copies of *d* from *A*, and that any changes *B* makes to *d* will not affect *C*’s copy of *d*. In contrast, the second approach (extending the entity’s scope) implies that *B* and *C* can now both access the same instance of *d* that is already known to *A*, so any changes that *B* makes to *d* will also affect the *d* available to *C*. In short, the first approach requires a copy-by-value mechanism, while the second can be implemented as copy-by-reference.

The copy-by-value implementation has the advantage that side effects are avoided. However, since we are dealing with objects that may have arbitrary complexity, copying data entities is not trivial. We could simply require that all data entities are cloneable and thereby put the responsibility on application

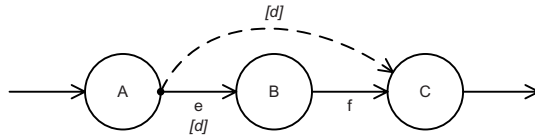


Fig. 5. Data flow with different copy-by-value and copy-by-reference behavior

developers, however, this is hardly a satisfactory solution for a framework that strives to make developers' jobs easier. Furthermore, cloning comes at a high performance and memory cost (especially since some clones may turn out to be unnecessary if the respective data sinks happen never to be visited in the user's subsequent traversal of the dialog graph), and some data structures may not be cleanly cloneable with justifiable effort. A restriction to flat copies or primitive types is not a realistic option either, since it would limit application developers' design freedom severely.

For these reasons, we prefer a copy-by-reference implementation of data flows. While it may involve a bit more subtle semantics, its basic concept and the situations in which side effects may occur are well known to experienced developers. This approach is much easier to implement for framework and application developers, and does not cause the performance and memory overhead of cloning.

The above considerations were confirmed by our initial prototype of a data flow control extension for the DCF: Here, clones of JavaBeans were only flat copies, so any nested references were not cloned, and the isolation between the data entities available to the dialog elements was not perfect. We are therefore currently switching to a data flow controller implementation that copies only object references instead of whole instances, and thus realizes the scope extension.

3.2 Data Flow Controller Design

Having determined the operational semantics of data flows, we still need a way to actually provide the specified data to the respective dialog elements in a web application. At first sight, an obvious solution would be to equip every mask and action instance with a look-up table that holds references to all data entities available to that element. This individual "element scope" could then be accessed by the application and presentation logic just like the session or application scope, using `setAttribute` and `getAttribute` methods. A data flow controller would ensure that object references are copied from one element scope to another according to the data flow specifications.

While this approach seems straightforward, it proves cumbersome in practice since the element scope itself cannot be easily made available to the presentation logic: Servlets, JSPs and other web resources provide convenient mechanisms for accessing the standard scopes (e.g. through the implicit `application`, `session` and `request` objects of the Java EE Expression Language), but do not provide as convenient means for accessing custom-built scopes.

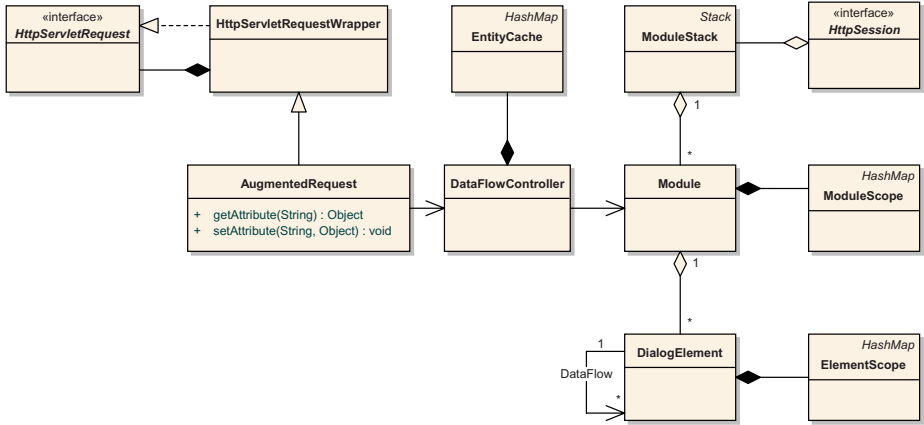


Fig. 6. Data flow control logic of the Dialog Control Framework

In the data flow extension to the DCF, we therefore took a slightly different approach: The data flow controller still manages look-up tables that contain the object references available to each dialog element. However, these tables are not directly accessible to the dialog elements. Rather, we project their contents into the existing request scope that is already conveniently available to all masks and actions.

As Fig. 6 shows, this can be achieved by wrapping the original HTTP request into a wrapper object (here, `AugmentedRequest`) that looks and behaves just like an HTTP request, since it passes most method calls directly to their counterparts in the original `HttpServletRequest`. The only exceptions are the `setAttribute` and `getAttribute` methods that are normally used to access the request scope, but now rerouted to the `DataFlowController`. This central controller has (via the current `Module` on top of the `ModuleStack`) access to the data flow model (a graph of `DialogElements` linked with `DataFlow` edges), the current `ModuleScope` and the current element’s `ElementScope` (two `HashMap`s containing the object references associated with the data entity labels).

Whenever a mask or action calls the request’s `getAttribute` method with some data entity label, the `DataFlowController` looks up the respective object reference in the current `ModuleScope` and `ElementScope` (the element scope takes precedence over the module scope in case both contain a reference with the same label), and returns the found reference.

Populating the look-up tables requires slightly more effort: Whenever a dialog element calls the request’s `setAttribute` method, the `DataFlowController` stores the respective data entity in the current element’s scope. It also looks up any departing data flow edges in the module’s data flow model and caches the entity’s object reference if it is associated with any outgoing event. Once an event is generated by the current dialog element, the associated data references

from the `EntityCache` are stored in the element scopes of the respective data flow receivers, and the cache is cleared.² This way, the data flow controller ensures that every element can access through the request context all data entities available to it according to the specification.

Besides giving the presentation and application logic convenient access the element and module scopes, projecting the element scope into the request scope has the additional benefit that the mechanism is transparent to other web application frameworks: In our prototype implementation, we integrated the DCF with JavaServer Faces (JSF) to make use of its UI component model and validation logic. Since JSF pages exchange data through the request scope, they can work with the new module scope and data flows seamlessly.

4 Related Work

Many modeling languages for web-based applications have traditionally had a strong focus on data-intensive web applications [3], allowing developers to specify how users navigate through complex data schemas. More recently, a number of languages such as OO-HDM [4], OO-H and UWE [5], and WebML [6] have also incorporated aspects of business process modeling, thereby narrowing the gap between process and navigation specifications that developers need to bridge. The data focus of these modeling languages is reflected in the variety of constructs they provide for specifying relations between and manipulations of data entities, most of which reside in the back-end of a web application.

However, apart from the transport links in WebML, which have similar semantics as our divergent data flows, the above modeling languages do not seem to provide explicit notation constructs for the fine-grained specification of inter-element data scopes and provision: The navigational model and process flow model in UWE, for example, specify how to navigate across and manipulate a data space provided by the back-end, but does not show which data instances are provided from one interface component to another. In OO-H, activity and navigation access diagrams enable developers to specify which navigation nodes will invoke which data-manipulation methods on data objects, but any scoping of these instances is not explicitly modeled.

The extension to the Dialog Flow Notation we introduced here focuses on the fine-grained specification of data scopes and data provisioning (i.e. which data instances are made available to which dialog elements). These data flow specifications do not have to be implemented manually, but are automatically enforced by the Dialog Control Framework at run-time.

In contrast to the above notations, the DFN does not provide constructs for specifying how the data that is provided to the various dialog elements is

² The actual data flow controller is a bit more complex than the diagram and this brief description suggest – among other things, it also moves incoming request parameters into request attributes, and ensures that data references are also copied to other scopes if they have not been set by calls to `setAttribute`, but received through data flows from other elements.

manipulated. This is in keeping with our philosophy that the dialog flow is what distinguishes web applications most from traditional applications – we therefore focus the DFN on the typical and unique challenges of navigation and data flow in web applications, and encourage developers to use other established notations for modeling those aspects that go beyond this layer (e.g. by using activity or state diagrams to specify how data entities are manipulated in actions).

Regarding run-time support for complex dialog flows, popular web application frameworks have recently also adopted the notion of encapsulating dialog sequences in so-called “flows” (in Spring Web Flow [7]) or “conversations” (in the Shale Framework [8]), which have associated data scopes. However, these frameworks do not provide mechanisms for the realization of parallel or divergent data flows, and are lacking a corresponding notation that would provide executable specifications for such data flows.

5 Conclusion

In this paper, we presented a data flow extension to the Dialog Flow Notation (DFN) that enables web application developers to specify data flows between arbitrary elements in a dialog graph, as well as between nested dialog modules. Since all data flows are associated with events whose traversal is the condition for data propagation, and the receiver of a data flow may be different from the receiver of the associated dialog event, developers can build on the existing DFN semantics to specify fine-grained conditional data propagation within a web application’s user interface. A new module-wide data scope complements the existing coarse scopes provided by common application servers.

To relieve developers of the tedious and error-prone effort of manually implementing secure and correct data flows throughout a web-based user interface, the data flow specifications created with this notation are executable: Using a graphical editor, they can be transformed into machine-readable specifications that are interpreted by our Dialog Control Framework (DCF), which manages the dialog and data flow automatically. At run-time, the framework ensures that data entities are always provided to their specified receivers, and that dialog elements can only access those data entities that they are entitled to. Since the new data flow mechanisms are transparently added to the existing request scope, other web application frameworks relying on this scope can also benefit from the specified data flows.

Our work aims to support the development process for web-based applications in both the design and the development phase: We expect it to be easier for designers to communicate with clients and other non-technical stakeholders in the software development process, since data flow information that was previously contained only in business process models can now be mapped to dialog flow diagrams, thereby providing a smoother transition from requirements to implementation. Given the specifications created in this way, we are confident that developers can reduce implementation, testing and maintenance efforts since they do not need to be concerned with the actual data flow implementation.

A prototype version of the enhanced Dialog Control Framework has already demonstrated that the described concepts are technically feasible. In our ongoing work, a first application for the data flow mechanism will be the DiaGen extension to the DCF, which automatically breaks dialog masks into wizard-style interaction sequences at run-time to cater to different device capabilities [9] – the use of divergent data flows will greatly simplify the auto-generated data propagation code in this context. Apart from this, we are striving to test our hypotheses regarding the positive impact on the software development process in real-life projects. We are also considering further refinements of the data flow notation to provide interfaces to other modeling languages for the specification of the application logic’s internal data handling. Another major upcoming research effort based on the data flow engine will be the development of algorithms for recognizing and handling backtracking, cloned browser windows and other unexpected user activities in web applications.

References

1. Gaedke, M., Beigl, M., Gellersen, H., Segor, C.: Web content delivery to heterogeneous mobile platforms. In: Kambayashi, Y., Lee, D.-L., Lim, E.-p., Mohania, M.K., Masunaga, Y. (eds.) *Advances in Database Technologies*. LNCS, vol. 1552, Springer, Heidelberg (1999)
2. Book, M., Gruhn, V.: Modeling web-based dialog flows for automatic dialog control. In: *ASE 2004. 19th IEEE International Conference on Automated Software Engineering*, pp. 100–109. IEEE Computer Society Press, Los Alamitos (2004)
3. Fraternali, P.: Tools and approaches for developing data-intensive web applications: A survey. *ACM Computing Surveys* 31(3), 227–263 (1999)
4. Rossi, G., Schmid, H., Lyardet, F.: Engineering business processes in web applications: Modeling and navigation issues. In: *Third International Workshop on Web-Oriented Software Technology*, pp. 81–89 (2006)
5. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of business processes in web application models. *Journal of Web Engineering* 3(1), 22–49 (2004)
6. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process modeling in web applications. *ACM Transactions on Software Engineering and Methodology* 15(4), 360–409 (2006)
7. Vervaet, E., et al.: Spring web flow. <http://www.springframework.org/go-webflow>
8. Apache Software Foundation: Shale framework. <http://shale.apache.org>
9. Book, M., Gruhn, V., Lehmann, M.: Automatic dialog mask generation for device-independent web applications. In: Book, M., Gruhn, V., Lehmann, M. (eds.) *ICWE 2006. Proceedings of the 6th International Conference on Web Engineering*, pp. 209–216. ACM Press, New York (2006)

Authoring Multi-device Web Applications with Database Access

Giulio Mori, Fabio Paternò, and Carmen Santoro

ISTI-CNR, Via Moruzzi 1,
56126 Pisa, Italy

{Giulio.Mori, Fabio.Paterno, Carmen.Santoro}@isti.cnr.it

Abstract. In this paper we present an environment for authoring Web sites through a model-based approach for user interface design. In particular, we focus on how it supports the access to remote databases and the dynamic generation of the Web pages presenting the corresponding query results. The environment is able to support development of applications implemented in many Web mark-up languages (XHTML, XHTML MP, X+V, VoiceXML) adapted to various interaction platforms (vocal, mobile, desktop, ...).

Keywords: Dynamic Web Applications, Multi-device Environments, Interfaces Adapting to the Platform.

1 Introduction

The increasing availability of interaction devices poses a number of challenges to Web designers and developers to obtain usable interactive applications. There is an increasing number of Web applications that would benefit from the possibility of accessing them through many interaction platforms that can even vary in terms of the modalities supported (graphical, vocal, gestural, ...). To address such issues the W3C consortium started various activities in the area of Ubiquitous Applications [3] and Multimodal Interfaces [5] and a number of mark-up languages have been developed, including XHTML, XHTML Mobile Profile (MP), VoiceXML, X+V.

In this context, model-based approaches have shown to be useful because they provide logical descriptions independent of the implementation languages and allow designers to better manage such increasing complexity. Many types of model-based approaches have been proposed: in the software engineering area there are several tools based on UML; for data-intensive Web applications there are approaches such as WebML [2]; examples of model-based approaches in the user interface area are UsiXML [7] and TERESA XML[4]. In the latter community, there is a general consensus regarding the useful logical descriptions [1][6] supporting user interface design: the *task and object level*, which reflects the user view of the interactive system in terms of logical activities and objects manipulated; the *abstract user interface*, which provides a modality-independent description of the user interface; the *concrete user interface*, which provides a modality-dependent but implementation language-independent description of the user interface; The *final implementation*, in an

implementation language for UIs. In particular, when designing multi-device user interfaces, this framework has the additional advantage that the task and the abstract level can be described through device-independent languages: it means that it is possible to use the same languages for whatever platform we aim to address.

In this paper, we present how an approach for model-based user interface design is able to support access to remote databases and the dynamic generation of the Web pages presenting the corresponding query. We discuss the solution proposed at both conceptual and implementation level. An example application is also presented, followed by empirical feedback collected from some Web designers and developers.

2 Design of Multi-device Applications with Database Access

Our approach is based on a tool, TERESA [4], which applies transformations and logical descriptions to generate interactive applications for different devices starting with different abstract models. Here we present an extension of the approach able to overcome the two main limitations of the original environment: generation of only static Web pages, and need for providing designers with greater control over the UI generated. In order to enhance the environment with features able to support access to remote databases, we provided support through different abstraction levels and suitable transformations among the different levels indicated in the introduction, including the generation of the appropriate code of the final user interfaces for various target platforms starting from specifications at the concrete level.

As for the final implementation language considered, for this type of support we focus on JSP, a well-known language for generating dynamic web pages. Therefore, if the application built by the designer contains at the concrete level objects supporting access to a remote database, the tool automatically produces dynamic JSP pages to provide support to this feature, which then will generate the implementation languages depending on the target platform. In the case of generation of applications for database access, the tool also generates some servlets that have to be installed in the server side in order to make the environment working properly. On the server side, the servlet is in charge of receiving the query specification, connect to the concerned database and execute the query to the database, sending the result of such query to a presentation. The environment for the generation of pages accessing to remote databases can provide different implementation languages (such as XHTML, XHTML Mobile Profile, VXML...) for the modalities supported by the platforms. We can analyse the new support by considering the possible abstraction languages.

2.1 The Task Level

The task models are specified in the ConcurTaskTrees notation [6]. At the task level, the allocation of tasks (namely, if they are performed by the application, or the user, or an interaction between the user and the system) and the objects manipulated by the tasks are important information for modelling and supporting the access to a database. Indeed, if we consider the specification of a task requiring access to a remote database, we should include an interaction task through which the user defines the attribute values; then, another (application) task is supposed to be sequentially connected to

the first one (from which it receives input values) and able to send such values to the back-end module of the server accessing the database. The last task is another application task, providing the user with the presentation of the query result. As for the specification of the objects manipulated by the tasks, they are classified in *perceivable* and *application* objects. The former have a direct impact on the UI as they are associated with concrete interface elements (menus, buttons, labels, etc.), whereas the “application” objects refer to logical objects corresponding to elements of the underlying application. When the task model is transformed into an abstract interface specification, tasks corresponding to database access are transformed into abstract interactors (called *activators*), which are associated with the functionality of the core that should be accessed and other attributes indicating the request parameters. The tasks presenting information to the user can be automatically detected because they are application tasks manipulating both perceivable and application objects. We will see that they will be supported by a new type of object at the abstract user interface level, the *table*, mapped at the concrete user interface level onto a *database_table* object.

2.2 The Abstract and the Concrete User Interface Description level

In TERESA XML, the logical interfaces are structured into presentations. To support access to remote database, at least two presentations are needed: one for allowing the user to specify the values of the query parameters and to send them to the associated database; another one for showing the query results. In terms of Abstract Interface, the first presentation is composed of a group of interactors allowing users to edit the query, which are related to an object of *activator* type (supporting the triggering of the functional module sending the values to the database). The second presentation includes the elements that will contain the query result. This type of element at the abstract level is a *description*, mapped at the concrete level onto a new type of element, a *database_table*. For the first presentation, the input elements, such as the *textedit* objects are mapped into the concrete elements of type *textfield*, whereas the *activator* is mapped onto the *activate_database* concrete element, introduced to support this new feature. As we said, the object *activate_database*, which has been introduced in TERESA XML as a new type of *activator* interactor, enables the connection with the concerned database. It has a number of attributes for specifying the parameters necessary to build the database query, e.g. *label* (for naming the concrete object), *database_properties* (for specifying the properties of the considered database, e.g. the server on which the database resides, the name of the database, user account), *attributes_names* (the set of attributes on which queries will be possible), *presentation* (the name of the presentation that will visualise the query result). Figure 1 shows how the *activate_database* element is supported within the TERESA authoring environment: in the left part there is the list of edited presentations, in the right top part there is the abstract specification of the currently selected presentation (the presentation titled “Ford News”), in the right bottom part there is the concrete description of the currently selected element of the abstract part (*Activator_7_0*). In the concrete user interface language for graphical interfaces we have also introduced the *table* element for managing the data corresponding to the query result. In particular, two types of tables were introduced: *database_table* and *normal_table*. The first one is used when the content of the table is not known at design time, as it is filled at runtime with the

query result. Thus, the designers do not know the number of its rows, but they should know the number of columns (database attributes). The *normal_table* is a table whose content (both textual and graphical content) is statically decided by the designer.

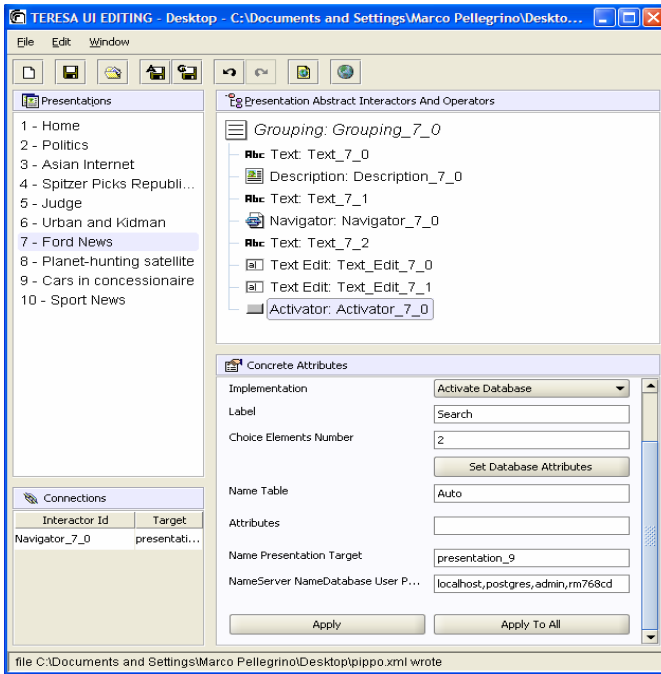


Fig. 1. The specification of the activate_database element with the TERESA tool

For both tables it is possible, with the tool, to customise their visualisation (modifying parameters such as colour and size of the table border, background colour, etc.).

3 An Example Application and a First Usability Evaluation

In this section we analyse an example application, working for both the desktop and the mobile platform, which provides the user with a list of up-to-date news. The user can query a database to get more information about a specific topic (eg: cars). Fig.1 shows the TERESA environment while editing the presentation allowing users to specify the query to the concerned database about cars. In this case, the attributes include brand name, type of fuel, number of car doors, etc. As the desktop is supposed to have good capabilities in display size, the designer chooses to visualise all the attributes: indeed, no attribute is specified in the related panel field labelled "Attributes". For the mobile platform, the number of attributes to be visualised should be more limited, and listed by the designer in the field "Attributes".

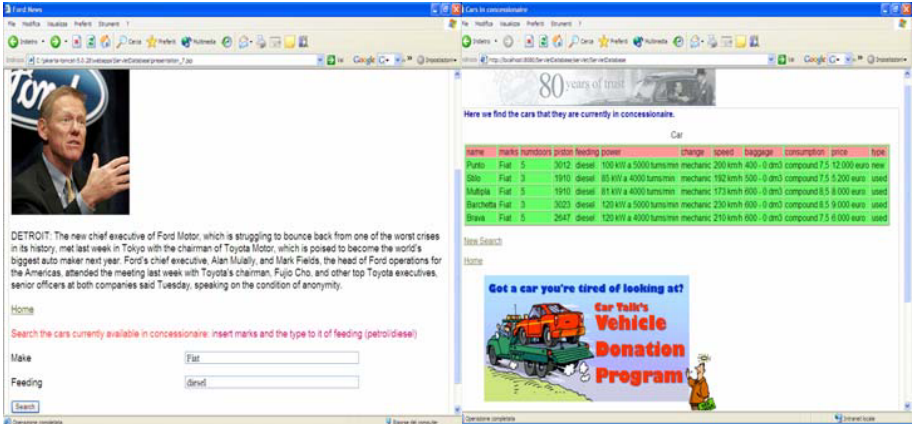


Fig. 2. The final user interface for the considered example (desktop platform)

In Figure 2 shows the user interface to specify the query (left) and the corresponding information result regarding all 12 attributes. For the mobile platform the case is different. Not only the designer has to select a more limited number of attributes to be visualised in the result (namely, the columns of the table), but, we have also to limit the information to be shown in each presentation (namely, the number of rows that can be reasonably presented in each table), because of the limited size of the mobile platform screen. In the example we decided to present only two attributes (car model name, and power type), and to enable the visualisation of only five rows in each presentation. The result of such settings produces the presentations shown in Fig. 3. As you can see the tool automatically adds the links required to navigate through the various pages generated for presenting the query result.



Fig. 3. The final user interface for the considered example (mobile platform)

A first evaluation session was performed to assess whether the new version provides designers of dynamic user interfaces with useful support and more control on the generated UIs, and its usability. A first test was carried out involving 5 developers recruited from the institute community, ageing between 25 and 38, and with laurea degree in Informatics. Before the exercise, users read a short introduction text about the tool and then instructed about the task that they were expected to carry out: building an application able to access a database with the support of the tool. Differences initially noticed between people having some knowledge of the tool and people who had not soon disappeared as soon as users gained familiarity with it. The intuitiveness of the tool was rated good (the average value was 3.5 in a (min) 1-to-5 (max) scale), although improvable. The pages built with the tool were judged usable (average rating: 4); testers reported that the final UI reflected their objectives, showing that the tool provides a good control on the UI produced (average rating: 4). Almost all users judged extremely valuable the help provided by the tool to the designers during the building of UIs accessing to remote databases (average rating: 4). Especially useful was considered the flexibility given by the tool in combining such dynamic objects with more static parts of the user interface.

4 Conclusions and Future Work

In this paper we present a new tool for supporting generation of interactive Web applications for various types of devices and able to access remote databases. The solution developed is able to support authoring of applications for desktop and mobile platforms, and generate page implementations in XHTML, XHTML MP, VoiceXML (only vocal interaction) and X+V (vocal and graphical). Future work will be dedicated to further testing it in order to receive empirical feedback regarding its usability and suggestions for further improvements.

Acknowledgments. We thank Marco Pellegrino for the help in the implementation.

References

1. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.A: A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers* 15(3), 289–308 (2003)
2. Ceri, S., Fraternali, P., Matera, M.: *IEEE Internet Computing*, 6, 4, July/August, pp.20–30 (2002)
3. Ubiquitous Web Application Activity: <http://www.w3.org/2007/uwa/>
4. Mori, G., Paternò, F., Santoro, C.: Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering* 30(8), 507–520 (2004)
5. Multimodal Interaction Activity, W3C, <http://www.w3.org/2002/mmi/>
6. Paterno, F.: *Model-Based Design and Evaluation of Interactive Applications*. Springer, Berlin (1999)
7. 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)

Enriching Hypermedia Application Interfaces

André T. S. Fialho and Daniel Schwabe

Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro
(PUC-Rio) – Caixa Postal 38.097 – 22.453-900 – Rio de Janeiro – RJ – Brazil
atfialho@inf.puc-rio.br, dschwabe@inf.puc-rio.br

Abstract. This paper presents a systematic approach for the authoring of animated multimedia transitions in Web applications, following the current trend of rich interfaces. The transitions are defined based on an abstract interface specification, over which a rhetorical structure is overlaid. This structure is then rendered over concrete interfaces by applying rhetorical style sheets, which define concrete animation schemes. The resulting applications has different transition animations defined according the type of navigation being carried out, always emphasizing the semantically important information. Preliminary evaluation indicates better user experience in using these interfaces.

1 Introduction

Current web applications have become increasingly more complex, and their interfaces correspondingly more sophisticated. A noticeable tendency is the introduction of animation as an integral part of Web application interfaces, after the advent of AJAX technologies (see, for instance, the Yahoo Design Patterns Library for transitions [7]). In hypermedia applications a more complex kind of animation is involved when considering entire interface changes that occur during navigation, where there is a transition as a result of a navigational operation. As such, this type of interface change is a prime candidate for the application of animation techniques.

This paper presents an approach for systematically enriching hypermedia applications by extending the SHDM approach [3]. In particular, attention is paid on how to relate animations to the application semantics expressed in the SHDM models.

The remainder of this paper is organized as follows. Section 2 gives some background on the use of animation, and on the representation of interfaces in SHDM. Section 3 presents the proposed approach, and Section 4 presents a discussion about the results and conclusions.

2 Background

The common definition of animation is the result of several static images that, when exhibited in sequence, creates an illusion of continuity and movement. Nowadays there are several systems that use animation with the purpose of enriching the interaction process and the ser experience or even provide smooth transitions the prime examples being the MacOS, and more recently Windows Vista. Some experiments

indicate that animations can be of help [1]. Furthermore, film editing knowledge has been applied to the design of human-computer interfaces [5].

Since animations will be expressed in terms of interface elements, we first summarize how the interface is specified in SHDM through Abstract and Concrete Interface Models [6]. The abstract interface model is built by defining the perceptible interface widgets. Interface widgets are defined as aggregations of primitive widget (such as text fields and buttons) and recursively of interface widgets. Navigational objects are mapped onto abstract interface widgets. This mapping gives them a perceptive appearance and also defines which objects will activate navigation.

The Abstract Interface is specified using the Abstract Widget Ontology, which establishes the vocabulary. According to it, an abstract interface widget is can be a Simple Activator, which is capable of reacting to external events, such as mouse clicks; an Element Exhibitor, which is able to exhibit some type of content, such as text or images; or a Capturer, which allows input of data, including widgets like input text fields, and selection widgets such as pull-down menus and checkboxes, etc... Finally, it can also be a CompositeInterfaceElement, which is a composition of any of the above.

Once the Abstract Interface has been defined, each element must be mapped onto both a navigation element, which will provide its contents, and a concrete interface widget, which will actually implement it in a given runtime environment.

3 Introducing Animations in Hypermedia Applications

The process for the insertion of animations during the design of the application is composed of four stages illustrated in Fig. 1, in which each of the stages produces a specific output used by the next stage.

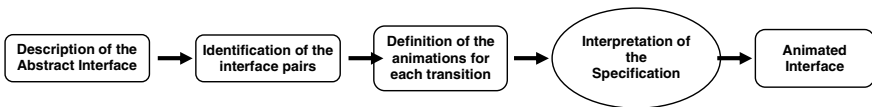


Fig. 1. Steps to produce an animated interface

The animations proposed in this work are displayed to the user during interactions that define a change in the navigational state. Each change is relative to a pair of distinct source/destination interfaces that represent these corresponding navigation states. We call this process a transition, which can be represented as an intermediate animated interface between two interfaces. This intermediate interface is only representational, and is described as a list of animations.

Each interface is a composition of widgets. A transition animation between interfaces is the process of visual transformation that transforms the source interface into the destination interface. Therefore, the transformation to be applied to each widget has to be specified.

To set up an interface animation it is necessary to identify the abstract widgets that compose each interface, and then specify which pairs of interfaces will define the source and destination of the transition. This is determined by the navigational

structure of the application and associated abstract interfaces, as specified in the SHDM model of the application.

For each defined transition we need to identify widgets (in the source and destination interfaces) that are mapped to the same or related element in the model. As a result, we identify which widgets remain unchanged, which disappear, which appear, and define which widgets are related. The first three behaviors are straightforward; the last one will depend on which relationship the designer wishes to expose. A common example of this last widget relation is when widgets in the source and destination interfaces are mapped to different attributes of the same element in the model (e.g., a name and a picture).

After pairing the widgets, we must provide the transition specification for the navigational change. The transition specification is made considering a pre-defined set of animation functions, identified considering that only three basic actions can be applied to a widget: a removal, an insertion or a transformation, which can be a

- Match – For widgets that are identified as remaining in the destination interface (i.e. they are present in both the source and the destination interfaces), it is necessary to match their appearance parameters such as position, size and color. This transformation animation responsible for matching these parameters.
- Trade – A transformation animation responsible for exposing the relation between two distinct related widgets during the transition, for example as a morph.
- Replace – When the same widget exists in the source and destination interfaces, but the associated elements in the model are different, this transformation animation replaces the information contained within a widget.
- Emphasize – A transformation animation that alters certain parameters such as size or color of a widget to emphasize an element.

Each of these functions will also have properties that describe the point in time it should occur within the transition, and which effect should be used (fade, push, grow, etc). These properties are specified according their role in the transition, described next.

3.1 Rhetorical Animation Structure

When we define the transition specification we must describe not only the list of the animation actions that will occur, but in which order in the timeline they will be executed, animation effect and the duration of each action. This sequence in which the animations are presented has great importance since it influences how the transition will be interpreted by the user.

In order to determine the best sequence and which effects should be used in each animation we propose the use of a rhetorical animation structure. This approach is inspired by the use of Rhetorical Structure Theory (RST) [4], as it has been used for generating animation plans ([2]). With this structure we can define the communicative role of each animation during the transition, and so identify which animations are more important and how they should be presented to better inform the user of the transformations that occur.

The rhetorical structure is specified in terms of rhetorical categories, which classify the various possible animation structures, as follows:

- Removal – Set of all animations that achieve an element removal (widgets that disappears). Rhetorically, these animations clean up the screen to set the stage for the upcoming transition;
- Widget Feedback – Any kind of transformation that represents an immediate feedback of the triggered widget. Rhetorically, these animations emphasize that the request made has been received, and the application is about to act on it.
- Layout animations – Set of animations that change (insert or transform) interface widgets that are independent of the contents being exhibited. These widgets are typically labels, borders, background images, etc...
- Secondary animations – Set of animations that transform interface widgets associated to secondary (satellite in terms of RST) elements
- Main animations – Set of animations that transform interface widgets associated to main (nucleus in terms of RST) elements.

Once the structures are chosen the designer must categorize the animation functions that have been identified in the previous step. We can partially aid this classification by observing the navigational model, identifying which relations are more important to describe. For example, transitions between objects of different classes should help identify the relation and the contexts associated with the navigation step being carried out in the transition.

The next step after the functions have been allocated to the rhetorical categories is to determine a rhetorical structure in which the animations will be presented. Different sequences can be arranged for each type of navigation. For example, Fig. 2 shows one possible sequence using these rhetorical categories.

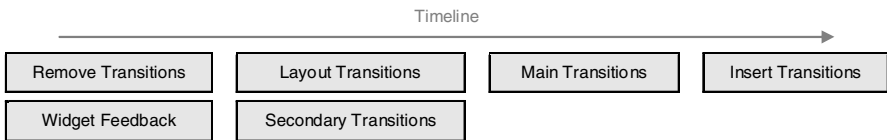


Fig. 2. Rhetorical animation structure

This sequence follows the rationale that first the screen should be cleared of elements that will disappear, simultaneously with a feedback of the activated widget. Next, the screen layout is changed to reflect the destination interface, in parallel with the secondary transitions (i.e., those that are judged as accessory to the main transition) are made. Then the main transition is carried out, as the most important part, followed by the insertion of new element.

After defining the rhetorical animation structure we need to map the categories into concrete transitions that describe which are the effects, duration and the sequence of the actions within the structure. The specification is done through a set of styles defined as a Rhetorical Style Sheet which reflects the designer preferences, and can be guided by the use of specific patterns that gather solutions to common transition problems within a specific context.

3.2 Implementation

The next step once the specification is done is to interpret this specification so the animations are presented to the user during the interaction. In this work we use an environment for supporting animation on web documents, in which HTML web pages represent the different types of interfaces, and JavaScript technology using dynamic HTML for the animations. Fig. 3 shows a diagram with the sequence of events in the implemented environment.

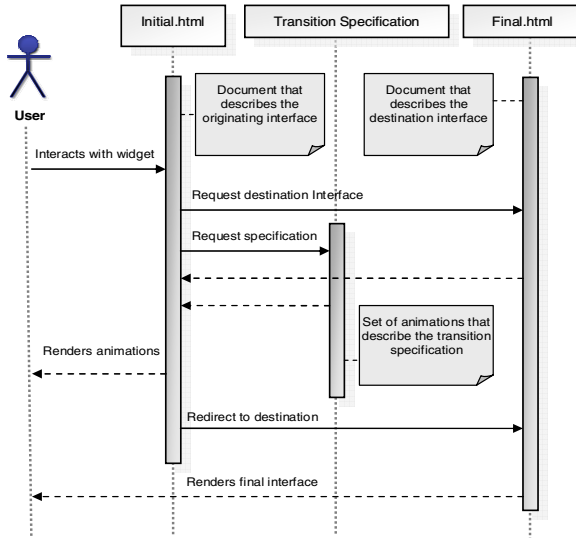


Fig. 3. Diagram representing the environment

Given the reduced space available, and the nature of this work, we have developed a demo flash application of an hypermedia movie database application using the approach described in this document, with step-through functionality to help understand the rhetorical animation structure being followed.. This example can be accessed at <http://www.inf.puc-rio.br/~atfialho/hmdb/hmdb.html> (requires a flash plug-in to execute).

4 Conclusions

This paper presented an approach for adding animation to hypermedia applications, enriching a set of existing models in SHDM. Although several initiatives exist to add animation to web pages, we are not aware of any published description of approaches dealing with entire web page transitions.

We have so far made only informal evaluations of the resulting interfaces obtained through this approach. Users have given positive feedback about the so-called “user experience”, and seem to prefer animated interfaces over equivalent non-animated interfaces. However, a more systematic evaluation will still be carried out.

While based on models and being more structured, the present approach still poses authoring difficulties, since they require manual insertion and choice of animation effects for each interface widget. We are currently investigating the use of wizards and the construction of a Rhetorical Style Sheet library to aid designers for the more common tasks routinely encountered in designing hypermedia applications.

Acknowledgement. Daniel Schwabe was partially supported by a grant from CNPq.

References

1. Bederson, B.B., Boltman, A.: Does Animation Help Users Build Mental Maps of Spatial Information? In: *InfoVis '99. Proceedings of IEEE Symposium on Information Visualization '99*, pp. 28–35. IEEE Computer Society Press, Los Alamitos (1999)
2. Kennedy, K., Mercer, R.E.: Using Communicative Acts to Plan the Cinematographic Structure of Animations. In: Cohen, R., Spencer, B. (eds.) *LNCS (LNAI)*, vol. 2338, pp. 132–146. Springer, Heidelberg (2002)
3. Lima, F., Schwabe, D.: Application Modeling for the Semantic Web. In: *Proceedings of LA-Web*, Santiago, Chile, Nov. 2003, pp. 93–102, IEEE Press, ISBN (2003), available at <http://www.la-web.org>
4. Mann, W.S., Thompson, S.: Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(13), 243–281 (1988)
5. May, J., Dean, M.P., e Barnard, P.J.: Using Film Cutting Techniques in Interface Design. In: *Human-Computer Interaction*, vol. 18, pp. 325–372. Lawrence Erlbaum Associates, Inc., Mahwah, NJ (2003)
6. Moura, S.S., Schwabe, D.: Interface Development for Hypermedia Applications in the Semantic Web. *Proc. of LA Web*, Ribeirão Preto, Brasil, pp. 106–113. IEEE CS Press, Los Alamito (2004)
7. Yahoo Design Patterns Library (transitions), available at <http://developer.yahoo.com/ypatterns/parent.php?pattern=transition>

Functional Web Applications

Torsten Gipp and Jürgen Ebert

University of Koblenz-Landau
{tgi,ebert}@uni-koblenz.de

Abstract. Web applications are complex software artefacts whose creation and maintenance is not feasible without abstractions, or models. Many special-purpose languages are used today as notations for these models. We show that *functional programming languages* can be used as modelling languages, offering substantial benefits. The precision and expressive power of functional languages helps in developing concise and maintainable specifications. We demonstrate our approach with the help of a simple example web site, using Haskell as the implementation language.

Keywords: Web Application Modelling, Specification, Functional Languages, Haskell.

1 Introduction

A *web application* (or *web site*) is an application that is delivered through the web. Creating such a web site is a complex task. Aside from the most trivial web sites that can be simply written down in one go, ‘real’ web sites require the application of a sound and consistent engineering approach. The end product must be expandable, reliable, error-free, and, of course, adhere to the given ‘specification’ perfectly. However, the trade-off between the required development time and the aspired quality is much too often solved by sacrificing the latter.

The Web Engineering discipline suggests using *models* to build abstract descriptions of a web site and to *derive* the end product from these models (e.g., [16]). This becomes particularly useful if the derivation can be done automatically (e.g., [23]), at least to a significant degree, and if the modelling does not impose too much overhead. Our idea is to apply modelling as well, but to do it using a *functional language*.

Example. As an example application we consider a travel booking system that offers its services over the web. The system is called the Travel Agency System (TAS). A customer can search for trips by supplying the origin and destination city together with the desired timeframe for a trip and the system will respond with a list of possible alternatives. Picking one of these provides further details about the selected trip, including the calculated prospective costs, and the customer can choose to book this trip, which will trigger the steps necessary for the financial transaction.

Every *page* of such a web site must be modelled, stating its inherent compositional *structure* and, most importantly, define which type of *content*, or data, it shall display. This must be done in a precise and abstract way. Here especially, functional languages are an almost natural choice because of their conciseness and power. Section 4.4 will introduce an abstract data type for page descriptions that is used as the backbone for the page models. We chose Haskell [25] as our implementation language, because it is in widespread use and well supported. In principle, however, the implementation of our approach does not depend on any particular language.

For this paper, we assume the reader to have some knowledge of functional programming languages. Due to space limitations, we cannot provide an extensive introduction into functional programming or Haskell. The abstract concepts, however, should be understandable nonetheless. Using Haskell, page specifications look like in this example:

```

tasTripDescription :: PageInfo
tasTripDescription = PageInfo "TAStripDescription" $
  λ params →
    do tasMainTemplate
      (Element "slots" []
       [ Element "heading" [] [Text "Trip_Description"]
       , Element "navigation" [] [ tasNavigation params ]
       , Element "body" []
         [ (tasTripDescriptionForm [])
         , (tasPreferencesList params)
         ]
       , Element "footer" [] [ Text "Footer" ]
       ]
    )

```

(1)

This constructs a page that fills the four slots heading, navigation, body and footer of a page template. A page template defines the common structural layout of all pages. The composition of a page is defined by assembling smaller parts, like the function `tasTripDescriptionForm` (see mark (1)) that describes a web form. Section 4.4 will give the necessary background information in full detail.

Figure 1 shows the hyperlinks between the single pages that constitute the example web site. The dashed arrows suggest the steps of the buying ‘workflow’ just described. A diagram like this is used as a depiction of the *navigation structure*. This structure models another one of six distinguished *aspects* of the web site. Section 4, the main part of this paper, will give an overview over this and the other important aspects, and section 4.3, in particular, will detail on diagrams that represent the navigation structure. But first, the following section 2 briefly lists the problems that our approach actually solves, and how this is done. We will reference related work as we go, but a coherent look into the state of the art is deferred until section 5. The TAS, our example web site, will be used throughout the remainder of this text to demonstrate our ideas. A preliminary result using the same example was described in a workshop paper [11].

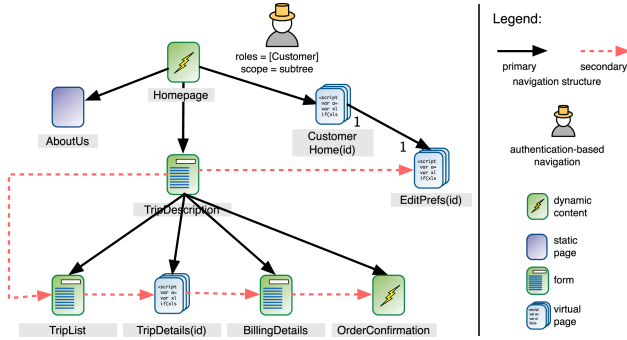


Fig. 1. Navigation structure

2 Benefits

Our approach delivers the following benefits:

- *Declarative description in combination with models allows for executable specifications.*

We employ a functional programming language to write down specifications of web sites, suggesting the combination of model-driven approaches with purely functional ones. Many web sites today are not specified in any way, or if they are, then the specifications are often not precise enough. A declarative description of a web site, however, written in a functional programming language, can serve as both: as a precise description and as an *executable* specification of the end product. The functional language is the perfect vehicle to produce *consistent* specifications.

- *Formalisation of the requirements as early as possible.*

We suggest that the formalisation of the requirements be tried as extensively as possible, which results in the specification being sufficiently close to the requirements. This ensures the consistency of the web site and its coherence with the specification. With functional languages, requirements that are given in a declarative way, like “In a trip description, the date of return must not be earlier than the date of departure.”, can be written down directly, in this case by giving a boolean function that checks the constraint on the two date values (see the code in section 4.7).

- *Abstractions can be introduced wherever needed, to master complexity.*

The emerging complexity of a web site and its model is almost necessarily handled by using abstractions. Functional languages provide very powerful ways to introduce new abstractions, which makes them an almost ideal vehicle for these kinds of specifications. Higher-order functions can be used to implement new control structures, for example, and combinator libraries (e.g., as in [13]) or domain-specific embedded languages (DSEL) like Peter Thiemann’s WASH/CGI [29] help tremendously by hiding implementation details.

- *The separation of concerns leads to the identification of six core aspects.*

We provide a backbone structure for the modelling of web sites by the separation of content, navigation, pages, queries and updates, presentation, and dynamics.

- *Support for testing and simulating.*

Functional programs also lend themselves very well to testing. Thus, it is possible to construct test cases directly from the requirements documents. The simulation of a web site visitor is possible as well, as a visit of the web site is nothing more than a sequence of function calls with concrete parameters.

3 Functional Web Sites

The core idea of our approach is the consistent use of a functional programming language to specify a web site. Following a strict *separation of concerns*, we partition the task of describing a web site into a set of separate aspects that can be tackled individually. The identified aspects are

- the content,
- the navigation structure (site map),
- the pages (navigation objects),
- queries and updates,
- the presentation, and
- the dynamics.

Section 4 will provide details for each of them. Each aspect is captured in a model, and the combination of these models provides the overall picture of the entire web site. Since the concerns are tackled separately, each can be treated according to the particular requirements for the respective aspect. The content, for example, is modelled using a conceptual model, which gives an abstract view on the content in form of *concepts* (or classes). The content itself is stored in a graph data structure, which in turn adheres to a (type) schema that is defined by the conceptual model. The navigation structure is captured using a visual language. The pages are modelled using functions that yield the actual page upon evaluation, using concrete parameter values. The dynamics, the queries and updates as well as the presentation are also specified in a functional way.

Thus, the functional language is used extensively. Any specification can benefit from this fact, because the full power of the functional programming language can be used at any level. The introduction of new abstractions is possible at any time, which is very important to master the complexity. As an example, regard the need for some sort of ‘templating’ when describing the single pages of a web site, as many pages will share common parts like design elements or a visualisation of the site map. We simply use a function that fills given fragments of a web page into a page template that contains everything that does *not* change from page to page. Additionally, the template itself is not static. It can contain conditional expressions that allow variations of the template to be handled smoothly. All this

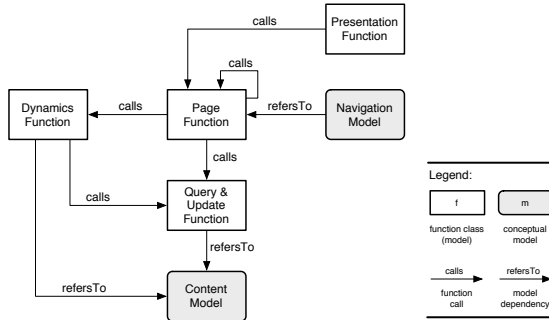


Fig. 2. Model overview

can be embedded into the functional language without syntactic clutter, due to the extensibility of the language via higher-order functions or the definition of new operators. Thus, new abstractions can be defined *in the specification itself*, without the need for any other language.

4 Functional Specifications

According to the list of aspects given in the previous section, we will now provide the respective details. Starting with a bird’s eye view (section 4.1), the subsequent sections introduce one aspect each.

4.1 Overview

Figure 2 shows the dependencies between the aspects. The aspects are depicted by rectangles. Rounded rectangles represent a conceptual model, normal rectangles represent a set of functions. The *content model* (at the bottom of the diagram) is a conceptual abstraction of the application domain. Relevant terms, or concepts, like, in our example, *customer* or *trip*, are identified and related to one another. We can therefore abstract from the actual content, and we can operate on the ‘class’ level rather than on the ‘instance’ level, to use object-oriented terminology. This level of abstraction is exploited in the definition of *query and update functions*. These functions specify the content access by using the terminology provided by the content model. The same abstraction step is done for the *dynamics functions*. They capture the application behaviour or ‘business logic’. Both kinds of functions are used in the definition of the single web pages through the *page functions*. They are at the heart of our approach. There is one function for each (kind of) page. The overall navigation structure of the web site is modelled separately in the *navigation model*, and the conversion of the abstract page descriptions into a concrete output language is specified by the *presentation functions*.

We will now give the details, starting with the *content* aspect.

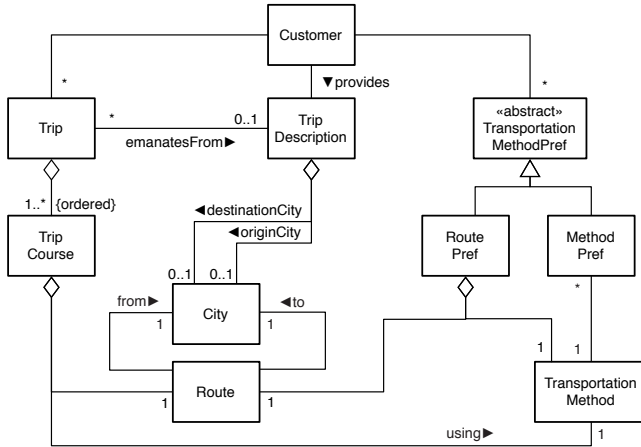


Fig. 3. Content model for the TAS example (attributes omitted)

4.2 Content

As far as the content is concerned, we consider the model (schema) level and the instance level.

Content Model. On the content model level, we capture the application domain by defining a conceptualisation, that is, the model identifies concepts and their relationships. The creation of the content model is an important step towards the understanding and mental structuring of the application domain. It captures what the application is all *about*. For the travel agency system, this includes trips, cities, customers and their preferences, flight schedules, etc.

It is almost consensus to use UML class diagrams as a notation for those conceptual models. This way, concepts (represented by classes), their attributes, and their relationships to other concepts can be visualised conveniently. The class diagram serves as a means of communication between the project participants. The model usually evolves over time, until, after a number of iterations, a sufficiently stable version has emerged.

The content model defines a *terminology* for talking about the application domain. The terms are referenced in other models, thus serving as a building block for the overall set of models. We will define query functions, for example, that access the content by using the abstractions defined in the content model.

Content Instance. The actual content itself is an *instance* of the content model. This means that the type of an individual content object corresponds to a class, and that its relationships adhere to the structure given by the content model. The content itself must be stored, retrieved, and changed. To this end, we employ *graphs*.

Graphs are a powerful mathematical structure that make an almost ideal data repository. We rely on typed and attributed graphs to store the web site's content. Nodes and edges of the graph have a type, and each type directly corresponds to a class or a relationship, respectively, in the content model. Instance attributes are attached to nodes. Node and edge types constitute a graph's *schema*, and the schema defines a *class* of graphs. The graph repository is implemented with the Functional Graph Library (FGL, [5]).

The job of retrieving and changing graph data is performed by the query and update functions, which will be dealt with in section 4.5.

4.3 Navigation Structure

Figure 1 shows the navigation structure, or *site map*, of our example web site. It is a visual representation of the hyperlink structure that connects all pages constituting the 'web'. In our opinion, a visual language is suited very well for showing, to a human, what the structure of the web site actually looks like. Especially during the design phase, moving icons around is easier and more 'intuitive' than writing formulas. This language has been successfully applied in some of our web engineering projects (e.g., [7]). The remainder of this section briefly introduces the language; cf. [10] for further details.

The *syntax* of the navigation structure diagrams is quite simple. The *primary* navigation structure, given by the solid arrows, defines a tree of page nodes. This tree assigns a unique path to every page. It is also easy to communicate to a web site visitor, who can create a mental image of the site map fairly quickly, which in turn is a very important ergonomic feature.

In the page functions, which will be explained shortly, the primary navigation structure is accessible through a simple, regular term structure.

The *secondary* navigation structure is visualised by dashed arrows. They represent arbitrary links between pages, without heeding the tree structure.

Types of Pages. There are four different types of pages, visually differentiated by four different page icons (cf. the legend in fig. 1). A lightning bolt marks a page as being *dynamic*, i.e., as a page whose relevant content is calculated (and thus potentially varies) at the time of access. In contrast, the content of *static* pages does not change at run-time. The classification of a page as being either static or dynamic is not necessarily unambiguous, because the definition of 'relevant content' is subject to interpretation. The distinction merely serves communicative purposes during modelling. There are no consequences on the implementation level. In our example, the home page is dynamic, and the 'about us' page is declared static.

Pages providing a form to let a web site visitor enter some data can be distinguished by a corresponding *form* icon. In our example, the user can enter and submit a trip description on the TripDescription page.

A small piece of script code on a stacked page icon signifies a *virtual* page that is computed by a script. In contrast to the 'lightning bolt' pages with dynamic content, the script-generated pages are *entirely* calculated by a set of

parameters, where one (the first) parameter defines the name of the page. This is inspired by the skolem functions from Strudel [6]. As an example, consider the `CustomerHome(id)` page, which represents a set of personalised pages, one page for each customer. The virtual pages do not exist under a pre-defined identifier, like the pages with dynamic content do. They are rather created on-the-fly, every time the page is called with a concrete identifier. We will use the term *instance* to talk about concrete virtual pages. There is one instance for each possible identifier.

These four basic web page flavours can also be mixed on one page. A virtual, a static, or a dynamic page can contain a form (or more than one). Since non-dynamic virtual pages do not make much sense – because this would mean that every instance looked the same and did not make use of the identifying parameter – the lightning bolt adornment will not be applied to virtual page icons, and virtual pages will count as *always* being dynamic.

Technically, pages of all four page types are defined by a page function of the *same* signature. Therefore, the page type chosen in the site map diagram is of no relevance implementation-wise, it is only important conceptually. Section 4.4 will deal with the page functions.

Additional Information about Links. The navigation structure diagram may also contain information on *authorisation-dependent navigation*. In the example, the primary link to `CustomerHome(id)` is annotated with a role-icon. It states that a web site visitor must possess the role `Customer` in order to access the page. The `scope=subtree` declaration expands this constraint to the whole subtree rooted at this page. The alternative value `thisPage` for `scope` would prohibit this expansion. The actual mechanism for checking the authorisation of a given user, a given action and a given object is intentionally left open in our approach. We can encompass any matrix-based scheme that maps permissions to roles.

The diagram can also capture the *multiplicity* of links to or from virtual pages. This is useful because virtual pages are like classes in that they represent a set of instances. Thus, we adopted a subset of the UML's multiplicity symbols to define how many instances may be connected. In figure 1, each `CustomerHome(id)` instance is connected to exactly one instance of `EditPref(id)`.

The actual checking of the constraints and of the authorisation is contained in the associated page functions in form of expressions. They also contain the definition of the links for the secondary navigation structure. Thus, we can employ the full power of the underlying functional language to provide conditional links, whose behaviour or mere existence depends on the system state and other context information.

4.4 Pages

Basic Definitions. Each page that is part of the web site is specified by a *page function*. A first example for a page function was given in the introduction.

Page functions return a value of the abstract type APD, short for *abstract page description*. This type allows for defining a page on an abstract level in terms of hierarchically nested, labelled, and attributed elements (comparable to XML). Here is the definition of this data type (in Haskell):

```
data APD
  = Text String
  | Element Name Attrs APDs
  | Link Name Attrs APDs PageInfo Params
  | Form Name Attrs APDs PageInfo Params
  | Field Name Attrs FieldType APDs String
  | Empty
```

There are six constructors for the APD type. An APD term can be a simple text node (**Text**); an element (**Element**) with a name, a list of attributes, and a list of child terms; a link or a form (**Link**, **Form**) with a name, a list of attributes, a list of child terms, information about the destination page, and a list of parameters that should be passed to this page; a field in a form (**Field**) with a name, a list of attributes, the type, a list of child terms, and a default field content; or it can simply be empty (**Empty**).

Some auxiliary declarations are used: The name of an element (**Name**) is a string. Attributes and parameters (**Attrs**, **Params**) are modelled as lists of key/value-pairs. Elements as well as forms and fields can contain child nodes, so they use APDs as a container for a list of arbitrary APD terms. A **PageInfo** term contains information about a single page, comprising an identifier, and a *page function*, which is the function that returns the APD for that page. Since functions are first-class objects in a functional language, the term is able to contain the proper function itself.

A page function basically maps a set of parameters to an APD. The current system state, including the session information, is passed along as an implicit parameter with the help of the Haskell **StateT** monad transformer (cf. [10]).

Links and form destinations are defined in terms of **PageInfos**. This implies that links are represented by terms in an APD structure, attached with a reference to the actual page function they link to. This guarantees link consistency.

Example. The following code for the function `tasTripDescriptionForm` represents a form for entering a trip description. The form itself is not a complete page, but rather just a building block. It is used inside another page function, `tasTripDescription`, that was already shown in the introduction. Figure 4 shows a rendered version of the form after a transformation to HTML (cf. section 4.6 for information about how this transformation is specified).

```
tasTripDescriptionForm :: StatefulPageFunc
tasTripDescriptionForm _ = do
  (graph, session) ← get
  return $
    Form "TripDescriptionForm" []
      [ Text "From:"
```

(1)

Fig. 4. Example page fragment

```

, Field "originCity" [] (OptionListField $ getOriginCities graph) [] ""      (2)
, Text "To:"
, Field "destCity" [] (OptionListField $ getDestCities graph) [] ""
, Text "Departure:"
, Field "dateOfDeparture" [] SimpleField [] ""
-- some similar fields omitted
, Text "Sorting_Order:"
, Field "sortingPreference" []
  (OptionListField $ map show possibleTripSortingPreferences) [] ""
, Field "submit" [] SubmitField [] ""
]
tasTripList []

```

The form is defined using the `Form` constructor (see mark (1)). The form's content is a list of `Text` and `Field` elements, which stand for a simple text label or a corresponding input field, respectively. The example code unveils two demonstrations of *re-use*:

1. The possibility for a page function to also define a *fragment* of a page, rather than a complete one, seems trivial and minor. In fact, however, this implies that pages can be assembled from smaller building blocks, which is a very important feature for encouraging re-use of page fragments.
2. The definition of helper functions like `getOriginCities` (see mark (2)) helps cleaning up the specification. Here, this function encapsulates the access of instance data that is stored in a graph.

Next to the list of form fields we can see the link to `tasTripList`. This is the `PageInfo` function that gets called when the form is actually submitted (the action handler).

The same principle applies to ordinary links between pages defined by the secondary navigation structure, as following a link is only a special case of submitting a form. One can regard a link as a form without any fields. Hyperlinks are defined with the `Link` constructor, in the same manner as forms.

Templates. The introduction already mentioned page templates as a useful abstraction element for pages with recurring content. We use a transformation-based approach to implement templating: A template function transforms a given input APD into an output APD in a filter-like manner, provided that the input conforms to some simple constraints. It must provide a "slots" element that contains a list of named slots. These slots are then merged with the template. It is the template function's job to define how the slots are actually rearranged, and thus it defines the general structure of all pages that use this template.

A template function traverses the given APD term and processes the slots it knows about. Typically, the slots' content is copied into a new APD term that represents the output page. Generally, arbitrary transformations on the input are possible.

Thus, templates are an example of an abstraction that is introduced in order to reduce complexity. This easy abstraction is possible due to the functional language.

4.5 Queries and Updates

We rely on graphs to store the content data. This has many advantages compared to other data structures or even to storing the data in an external database system. Since the content model is given in terms of classes and associations, it is possible to use almost any kind of representation for the underlying content repository.

One strong point for graphs is the possibility to employ powerful graph *querying* to retrieve values from it. In our implementation, we defined a simple querying interface to graphs.

Consider, as an example, the following query that retrieves the list of all available cities:

```
queryAllCities :: AttributedGraph → [String]
queryAllCities g =
  nodesToValues
    g
    (λ lbl → getValue lbl "id")
    (query g (nodes g) [ constrainByType "City" ])
```

Without diving into the implementation details, you can see from the signature that this function returns a list of strings, given a concrete Graph *g*. It does so by first selecting all nodes that are of type *City*, and then mapping a function that extracts the value of the *id* attribute over this list of nodes, resulting in the desired list of city names.

The function `queryAllCities` is used in the definition of `getOriginCities` and `getDestCities`.

Updating the graph is done analogously, by defining a function that returns the changed graph as its result. The calling function then puts this new graph into the session context, replacing the old one.

4.6 Presentation

The presentation model is given by defining one or more mappings (presentation functions) from an APD to the corresponding presentation level language. In the case of a web application that is to be rendered by a user agent that understands XHTML, a simple transformation of the regular APD into XHTML was implemented as a Haskell function. Alternatively, the APD could be converted to any other XML dialect first, and subsequent transformations may be done with technologies like XSLT [3]. All conceivable possibilities are open at this point, and the approach can be easily adapted to a great number of run-time systems. Note that the actual transformations can be selected at run-time, even on a page-to-page basis, or according to context information. This opens the path to adaptive web applications, encompassing customisation, personalisation, and multi-mediality.

Our implementation uses a straight-forward transformation of an APD into XHTML (cf. the example in figure 4). Links and form actions are coded into simple URL query strings. As a proof-of-concept, this is sufficient, but we would like to enhance this by integrating a proper web server (see the outlook in section 6).

4.7 Dynamics

The ‘business logic’ or *dynamics* of a web site is captured in the requirements documents. Use cases, for example, are employed to describe the behaviour of the site and which interaction steps are possible.

Using the terminology defined in the content model, many statements concerning the behaviour can be formalised. We suggest to do this with functional specifications. This way, the dynamics is broken down into well-specified functions that can be glued together in the page function definitions.

As a simple example, consider that, for the TAS, the requirements state that a trip description must always be well-formed, meaning that “(a) The cities denoted by `originCity` and `destinationCity` are not equal; and (b) `dateOfReturn` is later than `dateOfDeparture`.” (that is, the travel agency is not happy if you order a trip of length zero, and they don’t offer time travels either). This statement is captured by a function:

```

checkWellformedness :: TripDescriptionRecord → Bool
checkWellformedness td =
  (originCity td ≠ destinationCity td)                                (1)
  && (
    if (isJust (dateOfReturn td)) -- is the return date provided at all?
      then (fromJust (dateOfReturn td) > (dateOfDeparture td))      (2)
      else True
  )

```

Line (1) tests statement (a), and line (2) lets the function return `True` if, and only if, statement (b) holds as well.

The function is used in the page that shows the trip list (see line (1)):

```

tasTripList :: PageInfo
tasTripList = PageInfo "TASTripList" $
  λ params →
    do (graph, session) ← get
       navigation ← tasNavigation params
       tripDescr ← return $ validateTripDescription params (graph, session)
       if (isJust tripDescr && checkWellformedness (fromJust tripDescr))
         then do
           trips ← return $ prepareTrips (fromJust tripDescr) (graph, session)
           -- remainder omitted

```

(1)

5 Related Work

Relying on models for describing and specifying web sites has quite a long tradition. Overviews and comparisons of the most prominent approaches are given e.g. in [18], [8], and [16]. The approaches can be very coarsely classified by their ‘foundations’: Some focus on object-oriented models, others rely on entity-relationship models, and again others put documents into the center of interest. The most influential ‘schools’ are the graph-based Strudel approach [6], the TSIMMIS project [2], the ER-based RMM [14], Araneus [21], HDM [9] and OOHDm [26], WebML [1], and UWE [17]. The integration of the access to models into a programming language by using a domain specific language is reported in [24].

Significant effort has been put into developing and describing diverse methodologies for web site generation, of which none, to our knowledge, relies as much on functional specifications as we do. We envision a synergetic potential for the integration of our findings into existing approaches, or, vice versa, the integration of selected parts of the aforementioned approaches into ours. This vision was the reason for our approach being as abstract and as extensible as possible. The idea of integrating the models by making them functions, which is unique to our approach, clearly works best when *all* models are specified as functions.

It is interesting to compare the various notations used in the respective approaches. Some approaches rely on proprietary notations for some of the diagram types, especially for the hypertext models. A majority of the current approaches employs the UML (and its extension mechanisms) for the notation of diagrams. The main reasons stated for using UML are the availability of tools ([12, p. 2]), the fact that the UML is well-documented ([19, p. 2]), and the coherence gained by using UML for a web application that is connected to other systems that are already modelled using UML ([4, p. 64]). As of today, one can state that using UML class diagrams for the notation of entity-relationship views simply is standard practice.

Our approach is based on functional specifications. We aim at integrating the advantages of this ‘way of thinking’ into existing web engineering practice. To the best of our knowledge, only very little effort has been put into this direction. Producing HTML and XML with a functional language in a type-safe way is, e.g., investigated in [27], [28]. This is expanded by work directed towards the

specification of XHTML-based, interactive web applications (esp. [30], [13]). A very inspiring solution for Scheme is described in [20].

6 Summary and Conclusion

We described an approach to web site modelling by using functional languages. It is very important to use models as a basis for the development of web sites. We practise a separation of concerns and identify six core aspects that have to be considered for modelling, namely the content, the navigation structure, the pages, the queries and updates, the presentation, and the dynamics. The content model and the navigation structure are captured using ‘traditional’ object-oriented, conceptual models and a simple, graphical language, respectively. The other four aspects, however, are formalised using a functional language. In our examples, we used the functional programming language Haskell to write down these functional specifications.

The functional programming language can unleash its full power for the benefit of concise, easily maintainable, and re-usable specifications. Furthermore, the specifications are also executable, which is an advantage over the potential ‘gap’ between an abstract model, given in one language, and a manually crafted implementation written in another. The inherent features of a functional language allows for powerful and new abstractions wherever they are needed, which is almost a necessity to master the complexities of real-world applications. Relying on a wide-spread implementation language like Haskell facilitates the specification even further, because a great array of data structures and function libraries are already available.

Our future work will be directed towards the extension and streamlining of our approach. The specifications could benefit from improving the usage of type information for the content that is stored in the graph. Currently, we rely on simple, string-based labels for the types, while a real type-system, possibly built using the specification language, would be desirable. The same idea applies to the typing of the output documents; here the integration of a domain-specific embedded languages (DSEL) like WASH/HTML (for XHTML documents) might be tried, possibly sacrificing some of our approach’s generality.

We would also like to integrate our current implementation with either HSP [22] or WASH/CGI [30], two very powerful web server solutions written in Haskell. Both approaches offer substantial benefits, as they correctly deal with the bookkeeping of states and sessions, and also with the user jumping backwards in the browser history, or cloning the browser window. This integration should also provide an opportunity to test the scalability of our approach.

The template functions we use are an example for an ad-hoc extension that becomes possible because the functional programming language allows to do it. The notion of templates and slots could be sharpened by using a separate data structure for templates.

An end-user who wants to specify a complete web site needs better tool support than a text editor to write Haskell programs with. Libraries with commonly

needed auxiliary functions is not enough. A *visual language* for specifying the pages, for example, could be used by a graphical tool to *generate* the functional specifications. The visual language might be less powerful than the functional one, but it might suffice for the majority of the cases. An interesting compromise between user-friendliness and expressive power is sketched in [15], proposing a more user-friendly way of working with functions in a spread-sheet software. We would also like to investigate the integration of existing graphical modelling languages, so to avoid inventing yet another new visual language.

References

1. Ceri, S.: Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks* (Amsterdam, Netherlands: 1999), 33(1–6), pp. 137–157 (2000)
2. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: 16th Meeting of the Information Processing Society of Japan, pp. 7–18, Tokyo, Japan (1994)
3. Clark, J.: XSL Transformations (XSLT) Version 1.0. W3C Recommendation (1999) <http://www.w3.org/TR/xslt>
4. Conallen, J.: Modeling Web application architectures with UML. *Communications of the ACM* 42(10), 63–70 (1999)
5. Erwig, M.: Inductive graphs and functional graph algorithms. *Journal of Functional Programming* 11(5), 467–492 (2001)
6. Fernández, M., Florescu, D., Levy, A.Y., Suciu, D.: Declarative specification of Web sites with Strudel. *VLDB Journal* 9(1), 38–55 (2000)
7. Fleer, J.: Entwurf und Implementierung eines erweiterbaren Web-Portals. Studienarbeit, University of Koblenz-Landau, Koblenz (2005)
8. Fraternali, P.: Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys* 31(3), 227–263 (1999)
9. Garzotto, F., Paolini, P., Schwabe, D.: HDM – a model-based approach to hypertext application design. *ACM Transactions on Information Systems* 11(1), 1–26 (1993)
10. Gipp, T.: *Functional Web Site Specification*. Logos Verlag Berlin, Berlin (2006)
11. Gipp, T., Ebert, J.: Web engineering does profit from a functional approach. In: Koch, N., Vallecillo, A., Rossi, G. (eds.) *Workshop on Model-driven Web Engineering (MDWE 2005)*. Proceedings, pp. 40–49. University of Wollongong (July 2005)
12. Gorshkova, E., Novikov, B.: Exploiting UML extensibility in the design of web information systems. In: *Proc. Fifth International Baltic Conference on Databases and Information Systems*, pp. 49–64, Tallinn, Estonia (June 2002)
13. Hanus, M.: Type-oriented construction of web user interfaces. In: *PPDP’06. Proc. of the 8th International ACM SIGPLAN Conference on Principle and Practice of Declarative Programming*, pp. 27–38. ACM Press, NewYork (2006)
14. Isakowitz, T., Stohr, E.A., Balasubramanian, P.: RMM: A methodology for structured hypermedia design. *Communications of the ACM* 38(8), 34–44 (1995)
15. Jones, S.P., Blackwell, A., Burnett, M.: A user-centred approach to functions in Excel. *SIGPLAN Not.* 38(9), 165–176 (2003)
16. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W(eds.): *Web Engineering: Systematische Entwicklung von Web-Anwendungen*. dpunkt.verlag, Heidelberg (2004)

17. Knapp, A., Koch, N., Zhang, G., Hassler, H.-M.: Modeling business processes in web applications with ArgoUWE. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) UML 2004 - The Unified Modeling Language. Model Languages and Applications. LNCS, vol. 3273, pp. 69–83. Springer, Heidelberg (2004)
18. Koch, N.: A comparative study of methods for hypermedia development. Technical Report 9905, Ludwig Maximilians-Universität München (November 1999)
19. Koch, N., Kraus, A., Hennicker, R.: The authoring process of the UML-based Web engineering approach (june 2001) (on-line) <http://www.dsic.upv.es/west/iwmost01/files/contributions/NoraKoch/Uwe.pdf>
20. Krishnamurthi, S., Hopkins, P.W., McCarthy, J., Graunke, P.T., Pettyjohn, G., Felleisen, M.: Implementation and use of the plt scheme web server. Higher-Order and Symbolic Computation (2007)
21. Mecca, G., Merialdo, P., Atzeni, P.: Araneus in the era of XML. IEEE Data Engineering Bulletin 22(3), 19–26 (1999)
22. Meijer, E., van Velzen, D.: Haskell server pages - functional programming and the battle for the middle tier. Electronic Notes in Theoretical Computer Science, 41(1) (2001)
23. Meliá, S., Kraus, A., Koch, N.: Mda transformations applied to web application development. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 465–471. Springer, Heidelberg (2005)
24. Nunes, D.A., Schwabe, D.: Rapid prototyping of web applications combining domain specific languages and model driven design. In: ICWE '06. Proceedings of the 6th international conference on Web engineering, New York, NY, USA, pp. 153–160. ACM Press, NewYork (2006)
25. Peterson, J., Chitil, O.: The Haskell Home Page. (December 2004) <http://www.haskell.org/>
26. Schwabe, D., Rossi, G.: The object-oriented hypermedia design model. Communications of the ACM 38(8), 45–46 (1995)
27. Thiemann, P.: Modeling HTML in Haskell. In: Pontelli, E., Santos Costa, V. (eds.) PADL 2000. LNCS, vol. 1753, p. 263. Springer, Heidelberg (2000)
28. Thiemann, P.: A typed representation for HTML and XML documents in Haskell. Journal of Functional Programming 12(4 and 5), 435–468 (2002)
29. Thiemann, P.: An Embedded Domain-Specific Language for Type-Safe Server-Side Web-Scripting. ACM Transactions on Internet Technology 5(1), 1–46 (2005)
30. Thiemann, P.: Web Authoring System Haskell (WASH) (February 2007) <http://www.informatik.uni-freiburg.de/~thiemann/haskell/WASH/>

Integrating Databases, Search Engines and Web Applications: A Model-Driven Approach

Alessandro Bozzon¹, Tereza Iofciu², Wolfgang Nejdl², and Sascha Tönnies²

¹ Politecnico di Milano , P.zza L. da Vinci 32, I-20133 Milano, Italy

² Forschungszentrum L3S, Appelstr. 9a, 30167 Hannover, Germany
bozzon@elet.polimi.it, {iofcIU,nejdl,toennies}@l3s.de

Abstract. This paper addresses conceptual modeling and automatic code generation for search engine integration with data intensive Web applications. We have analyzed the similarities (and differences) between IR and database systems to extend an existing domain specific language for data-driven Web applications. The extended Web modeling language specifies the search engine's index schemas based on the data schema of the Web application and uniquely designs the interaction between the database, the Web application, the search engine and users. We also provide an implementation of a CASE tool extension for visual modeling and code generation. Experimentation of the proposed approach has been successfully applied in the context of the COOPER project.

Keywords: Web Engineering, Web Site Design, Search Engine Design, Index Modeling.

1 Introduction and Motivation

In data intensive Web applications, traditional searching functionalities are based on site navigation and database-driven search interfaces with exact query matching and no results ranking. There are many applications, though, where the content relies not only on structured data, but also on unstructured, textual data; such data, from small descriptions or abstracts to publications, manuals or complete books, may reside in the database or in separate repositories. Examples of these classes of applications are digital libraries, project document repositories or even simple on-line book stores. The search capabilities provided by database-driven search interfaces are less effective w.r.t search engines (using information retrieval, IR, techniques), especially when dealing with large quantities of text: users, nowadays, are accustomed to interact with search engines to satisfy their information needs and the simple yet effective Web search functionalities offered, for example, by Google are by all means standard. Nevertheless, using external search engines may not suffice: they usually crawl only documents openly provided by the application on the Web, losing possible contextual information (unless the Web application provides an exporting tool for its metadata) and rank resources taking into account all the data existing on the Web. Sensitive information and private data are not published freely on the Web and authorized

users would not be able to make use of them in searching. Integrating IR functionalities within the Web applications allows a finer control over the indexed data, possibly by making use of the published information schema and by tailoring the retrieval performances to the specific needs of different categories of users. Traditional solutions integrate IR and database systems by writing a central layer of software acting as a bridge between the sub-systems to properly distribute indexing and querying functionalities [1]. The key advantage is that the two systems are independent commercial products continuously improved by vendors and can be combined independently from the respective technologies. In addition, existing databases and applications might co-exists while new IR features are introduced to enhance searching performances. On the other hand, such solutions introduce problems of data integrity, portability and run-time performance which can be solved only by means of integration software designed for the specific adopted technologies, databases schema and applications.

This paper presents a method for the conceptual modeling and automatic generation of data intensive Web applications with search engine capabilities by extending an existing Web modeling language. The aim is to overcome the problems of ad-hoc integration between database and IR systems by leveraging on a model-driven approach which allows a declarative specification of the index structure and content composition. Our model-driven approach leads to a homogenous specification as the conceptual modeling and the automatic code generation for indexing and searching are done through the same techniques as for the database system. Thus, the synchronization between the search engine indexes and the data back-end is also modeled. The starting point is the Web Modeling Language (WebML), a visual modeling language for the high-level specification of data-intensive Web applications and the automatic generation of their implementation code. Nonetheless, the examples and results are independent from the WebML notations and could be applied to other modeling languages [2] and tools.

The contribution of the paper is twofold: 1) an extension of the concepts of a Web model to allow (i) the specification of search engine index schemas based on the data schema of the Web application and (ii) the design of the interaction between the database, the Web application, the search engine and users; the core problem is how to provide (in a model-driven fashion) access and synchronization features to (and from) the search engine. 2) A validation-by implementation of the proposed model extensions. We have extended a commercial CASE tool (WebRatio) with novel runtime components and code generation rules. These two contributions go beyond the state of the practice in the field and allow a seamless integration and synchronization between database, Web Applications and search engine technologies.

2 Case Study

To better motivate the need for the integration between databases and search engine technologies in the domain of data intensive Web Applications, we introduce

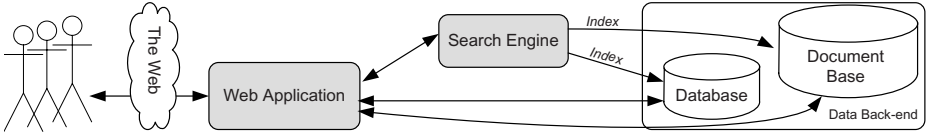


Fig. 1. Architecture of the case study

as an example the COOPER platform [3], a model-driven solution for project oriented environments, where users (students, professors or company employees) from different locations form teams to solve assigned tasks. The COOPER platform is currently used by three corporate users, two universities and one company, each one having different data and searching requirements. It is therefore important for the core platform to be easily adaptable to clients' specifications, in order to respond quickly to changes in the platform requirements.

The COOPER architecture, presented in Figure 1, is built upon a data back-end mainly composed of (i) a document base, containing the knowledge resources managed by the platform, and (ii) a database, containing meta-data about such resources and references to the files in the document base. Users work in teams and thus participate in one or more projects at a given time. On top of the data back-end there are (i) the Web Application, connected both to the database and to the document base and acting as a bridge between the two with the aid of (ii) a search engine module, responsible for indexing the content of the whole data back-end. Figure 2 shows a simplified data schema for the COOPER core database. Users can perform two types of search operations over the data back-end: by operating only over database data, via SQL queries, or by leveraging on the search engine that queries both the information in the database and in the document base, ordering the results according to their relevance to the user's query. Both querying strategies should take into account the whole data structure of the COOPER database. Users can search for resources, users or related projects. Search operations have to take into account also the information related to the searched item (e.g., when searching for a resource, users can specify information about its authors and related projects). After submitting the search request, users can navigate through the results, refine their search query or select one result from the list for further navigation. For example when a user wants to find resources about "indexing" written by persons working in the COOPER project, she could post the following query: 'indexing' and 'author works COOPER'. Such type of requests can also be defined for database queries when the text is stored in the database, but only by hard-wiring the query predicate into the Web application's code, restricting the user's search horizon. When the user is creating an IR query she doesn't need to know much of the schema itself: the queries are more flexible, allowing users to express their information needs without the constraints imposed by the underlying query structure. The application also allows users to modify the content of the repository. Users may add new

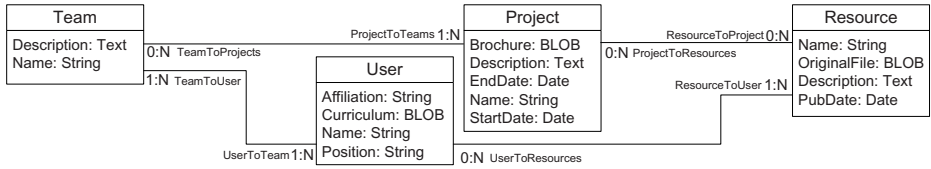


Fig. 2. WebML Data model of the case study application

resources, whose content has to be indexed by the search engine. Data about a resource (or the resource itself) may be modified by the user, forcing the system to update both the database and the search engine index. Finally, a resource can also be deleted from the repository, which means that both the database entry and the indexed information have to be deleted.

3 Conceptual Modeling for Search Engine in Web Applications

In this section we present how we have extended a conceptual model for data-intensive Web applications to support integration with a search engine, in order to reuse well-proven code generation tools and implementation architectures. We outline a comparison between IR and database systems in order to provide a new modeling layer for the definition of index schemas; afterward we present how we enriched the hypertext model with a set of primitives enabling the specification of the interaction between the database, the Web application, the search engine and users.

3.1 Modeling the Index Structure

In Web applications design, the goal of the *data model* is to enable the specification of the data used by applications in a formal way; likewise, when creating a search engine, designers should also be able to specify the model for the indexed content. According to [4], an IR model is a quadruple $\langle D, Q, F, R(q_i, d_j) \rangle$ where D is a set composed of logical views for the document in the collection, Q is a set composed of logical views for the user's information needs (queries), F is a framework for modeling the document representation, the queries and their relationships; $R(q_i, d_j)$ is a ranking function which associates a real number with a query $q_i \in Q$ and a document representation $d_j \in D$. In our work we refer as *index data model* to the set of document representation contained in D . In the classic information retrieval, such model is simple and considers that an *index* is composed of a set of *documents* (where by document we mean the representation of any indexable item) and each *document* is described by a set of representative words called *index terms* (that are the document words or part of words). This unstructured model relies on indexed contents coming either from structured (e.g. database), semi-structured (e.g. XML documents)

and unstructured data (e.g. Web pages). Different works (like [5] or [6]) show the usefulness (and the potential) of a model combining information on text content with information about the document structure. These models (also known as *structured text retrieval models*) bring structure into the index data model and allow users to specify queries combining the text with the specification of structural components of the document.

Analysis of the index data model from a databases perspective. The logical representation of indexes is an abstraction for their actual physical implementation (e.g. inverted indexes, suffix trees, suffix arrays or signature files). This abstraction resembles the data independence principle exploited by databases and, by further investigation, it appears clear how databases and search engine indexes have some similarities in the nature of their data structures: in the relational model we refer to a *table* as a collection of rows having a uniform structure and intended meaning; a *table* is composed by a set of columns, called *attributes*, having values taken from a set of domains (like integers, string or boolean values). Likewise, in the index data model, we refer to an *index* as a collection of *documents* of a given (possibly generic) type having uniform structure and intended meaning where a *document* is composed of a (possibly unitary) set of *fields* having values also belonging to different domains (string, date, integer...).

Differently from the databases, though, search engine indexes do not have *functional dependencies* nor *inclusion dependencies* defined for their fields, except for an implied *key dependency* used to uniquely identify documents into an index. Moreover, it is not possible to define *join dependencies* between fields belonging to different indexes. Another difference enlarging the gap between the database data model and the index data model is the lack of standard *data definition* and *data manipulation* languages. For example both in literature and in industry there is no standard query language convention (such as SQL for databases) for search engines; this heterogeneity is mainly due to a high dependency of the adopted query convention to the structure and to the nature of the items in the indexed collection.

In its simplest form, for a collection of items with textual representation, a query is composed of keywords and the items retrieved contain these keywords. An extension of this simple querying mechanism is the case of a collection of structured text documents, where the use of index fields allows users to search not only in the whole document but also in its specific attributes. From a database model perspective, though, just *selection* and *projection* operators are available: users can specify keyword-based queries over fields belonging to the document structure and, possibly, only a subset of all the fields in the document can be shown as result. Queries over indexes are also set-oriented, which means that traditional ordering clauses can be defined: a standard ordering clause is based on the $R(q_i, d_j)$ ranking function. Also grouping clauses over fields can be defined but, usually, just to support advanced result navigation techniques like, for example, faceted search [7].

The index data model. Web modeling languages usually make use of Entity-Relationship (E-R) or UML class diagrams to model applications' data. We propose the usage of a subset of the E-R model to define a complete representation of indexes and their structure. With respect to the E-R model, we define the following modeling primitives: the *entity* concept is converted into the (i) *index* concept, which represents a description of the common features of a set of resources to index; an index *document* is the indexed representation of a resource. Index *fields*(ii) represent the properties of the index data that are relevant for the application's purposes. Since all index documents must be uniquely distinguishable, we define a single special purpose field named (iii) *object identifier*, or *OID*, whose purpose is to assign a distinct identifier to each document in the index. Fields can be *typed* and we assume that a field belongs to one of these domains: *String*, *Integer*, *Date*, *BLOB* and *URL*. While the meaning of *String*, *Integer* and *Date* data types are self-explanatory, *BLOB* and *URL* fields assume a different flavor in index modeling and, hence, they deserve some further explanations: *BLOB* and *URL* fields contain information taken from resources which, because of their size and format (MIME Type), must be indexed (and perhaps queried) differently from plain text. Example of *BLOB* resources are PDF/Word/Excel documents, or audio/video files while *URL* resources consist of indexable documents identifiable on the Web by their Uniform Resource Locator. Finally, the *relationship* concept is not used by the index data model since, as stated in [3.1](#), there are no semantic connections between indexes. Likewise, no *generalization hierarchies* are allowed.

Index data model for the case study application. Figure [3](#) depicts the index data model for the case study application; we modeled an index for resources (*ResourceIndex*), for projects (*ProjectIndex*) and for users (*UserIndex*). Since the goal of our search engine is to allow the retrieval of database instances, each index has a set of fields composed by a subset of the associated data model entity's attributes: the *ResourceIndex*, for example, contains the *Name*, *Description* and *OriginalFile* fields. In addition, such a set is completed with further fields specifically aimed to enrich the information contained in the index in order to improve its retrieval performances. For example, the *ResourceIndex* has a field named *BelongingTo* which, as the name suggests, will contain indexed information about the projects for which the resources have been produced.

ResourceIndex	ProjectIndex	UserIndex
BelongingTo: String	Brochure: BLOB	Affiliation: String
Description: String	Description: String	Curriculum: BLOB
IndexingDate: Date	InvolvedUsers: String	Name: String
OriginalFile: BLOB	Name: String	Position: String
Name: String	RelatedDocuments: String	PublishedDocuments: String
WrittenBy: String		WorkingProjects: String

Fig. 3. Index data model for the case study application

Mapping the index model over the data model. Once the index model is defined, it is necessary to instruct the search engine about how to extract information from its data source(s). In our approach this is accomplished in a model-driven fashion by specifying a mapping between an *index* in the index schema and an *entity* in the data schema, making indexes a partially materialized view over the application data. Figure 4 depicts a graphical representation of the case study data model mapping for the *ResourceIndex* index: the $\{/Resource\}$ label aside the index name represents the identifier for the entity on which the index is mapped [4]. An entity can have more than one index associated with it, and each index could assume different forms depending on the design goals. Since search engines and databases are different systems, unique identifiers for

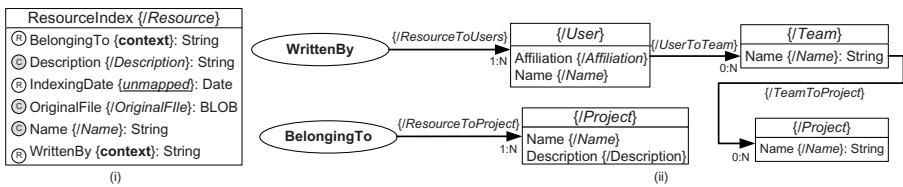


Fig. 4. Example Of Mapping

tuples and documents need to be paired manually by keeping an unambiguous reference to the indexed tuple inside the index documents; the index will contain two implicit fields, not specifiable by designers but nonetheless fundamental for the indexing process: (i) a unique identifier for the document in the index (named *OID*) and a (ii) unique identifier, (named *entityOID* or *EOID*) for the indexed tuple in the database. Other *fields* composing an index can be classified according to two orthogonal mapping dimensions, summarized in Table 1. The first one, called *storing policy*, specifies how the index should manage the indexed content for a given field; an index may cache a (possibly condensed) portion of the application data in order to improve the searching activity. An example for this situation is the case of resources like textual PDF/WORD/TXT files: caching snippets of their data allow one to present to users a summarized but significant snapshot of their contents, improving the visual representation of the search results without the need to download the original file; also caching database content might be useful, especially for attributes containing a significant quantity of text. Caching also enables the use of advanced search engine features like query keywords highlighting in the shown snippets. Nevertheless, caching introduces space (or time) consumption issues which are highly dependent on specific application needs but, as side effect, it might help to improve the overall performances of the application by minimizing the number of additional requests addressed to databases or to the repository. Fields can be therefore

¹ The notation using a / character in front of the entity identifier is borrowed from UML, where it is used for specify derived attributes.

Table 1. Mapping dimensions for the specification of an index content

Mapping Dimension	Allowed Values	Description
Storing Policy	Reference, Cached	Management policy for an index field content
Field Association	Mapped, Document, Un-mapped, Context	Source of an index field content

defined as *reference* or *cached*. *Reference* fields contain a representation of the original data useful just for index purposes. Conversely, in *cached* fields, original data are also stored (possibly after some processing) to allow their publication without further access to original data sources. Figure 4 reports an example of the notation used to express the *storing policy*: the field named *BelongingTo*, associated with a white circle, is defined as *reference* while the *OriginalFile* field, marked with a filled circle, is *cached*.

The second mapping dimension, named *field association*, addresses the specification of the source of a field data. *Mapped* fields contain information directly extracted from one attribute of a data schema entity. In Figure 4(i) mapped fields (like *Description*) are defined by the indexed attribute in the data schema, specified, for example, by the $\{/Description\}$ label aside the field name. Such information can be indexed (or cached) using the same domain type as the original attribute or, when allowed, translated into a different domain. A particular mention should be addressed to *BLOB* attributes: since an entity instance contains only a reference to the original file, a translation into a different domain (e.i. different from *BLOB*) will imply the indexing of the raw information contained into the attribute (which, in our case, is the file path). *Document* fields, conversely, address the cases where such a change in the domain type is not performed by containing an indexed version of the files stored into the data repository. *Unmapped* fields, instead, contain values not directly derived from the database nor from indexed documents; their content is specified at run-time and might differ from document to document. Such fields can be used, for example, to index time-dependent information, like the *Indexing Date* field in Figure 4(i), which stores the date of an index document creation; in general, the content of *unmapped* fields can be freely specified, allowing the definition of custom content composition tailored to specific needs. Finally, *Context* fields contain information originating from the context of the indexed entity. Here by *context* we mean information semantically associated to a single concept split across different related entities because of the data schema design process: for example, the *WrittenBy* field in the *ResourceIndex* from Figure 4(ii) will contain indexed information about the users who wrote the indexed document, like their names or their affiliations or even the teams they work in. The content of a context field, hence, is composed by data indexed from attributes belonging to entities related with the one defining the index. We leverage on the data derivation language of WebML (which makes use of an OCL-like syntax) to compose such content. In Figure 4(ii), for example, the content of the *BelongingTo* field is obtained by

navigating the *ResourceToProjects* relationship role and by indexing the value of the *Description* and *Name* attributes of the related entity instances.

3.2 Modeling Search Engine Interaction in the Hypertext Model

In WebML the hypertext model specifies the organization of the front-end interface of a Web Application by providing a set of modeling constructs for the specification of its publication (*pages* and *content units*), operation (*content units*) and navigation (*links*) aspects [8]. We extended the WebML primitives to allow the specification of the interaction between the Web application, the user and the search engine. Similarly to how WebML units work on database data, the new modeling primitives cover all the spectrum of querying and updating operations specifiable for a search engine, taking into account the boundaries defined by the index data model as described in Section 3.1.

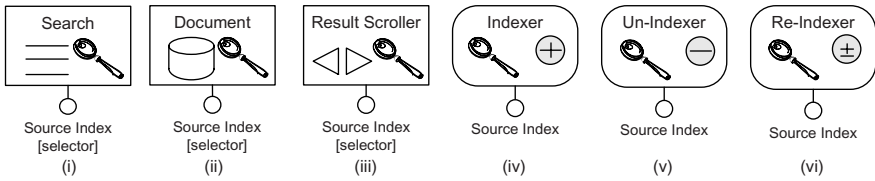


Fig. 5. Modeling primitives for the specification of search engine-aware Web interfaces

Modeling search engines-aware Web interfaces. The first type of user interaction with search engines is the possibility to query and navigate content extracted from the indexes. We extend the WebML notation with three content units, depicted in Figure 5: the (i)*Search* unit, the (ii)*Document* unit and the (iii)*Search Scroller* unit. Table 2 gives an overview of their common properties. These units model the *publication*, in the hypertext, of documents extracted from an index defined in the index schema. To specify where the content of such units comes from, we use two concepts: (i)the *source index*, which is the name of the index from which the documents are extracted, and (ii)the *selector*, which is a predicate used to define queries over the search engine. In IR systems a basic query is a set of keywords and boolean operators (OR, AND and BUT) having relaxed constraints (w.r.t. boolean algebra) for their matching properties or *phrase* and *proximity* operator [4]. For document collections with fixed structure, sub-queries over specific fields may also be addressed; the set of documents delivered by each sub-query is combined using boolean operators to compose the final collection of retrieved documents. Our approach focuses on the specification of queries leveraging on the structure of a given index, using boolean operators to join sub-queries addressed to the index fields. The *selector* of the new units will be therefore composed by the conjunction of a (possibly empty) set of conditions defined over the fields of their *source* index; in addition, every unit disposes of an implicit condition over the *OID* field, to retrieve one or more documents given

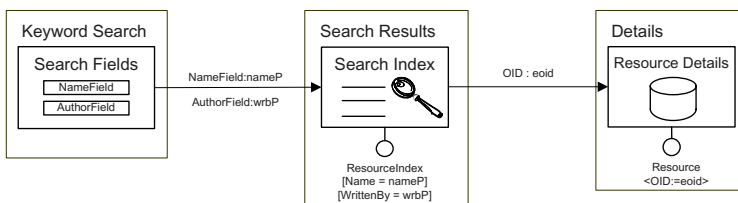
Table 2. Properties and Parameters for the Data Primitives

Property	Description
Source Index	the index providing the documents to the unit
Selector	a set of conditions defined over the source index
Input Parameter	<i>Search String</i> : set of terms to search for; <i>OID</i> : identifier for the index documents to publish
Output Parameter	<i>OID</i> : a set of identifiers for the retrieved documents; <i>EOID</i> : a set of identifiers of the database tuple associated with the retrieved documents

their unique identifier. Matching values for such conditions are retrieved from the parameters associated with the input links of the unit. When no conditions are defined, only the default *searchString* input parameter is available and the query is performed by matching its content over all the fields in the index.

The basic navigational pattern for defining the interaction with the search engine is when the user poses a query to it and then selects one result from the retrieved list. The *Search* unit models such process by providing as output the document *OID* as well as the associated *EOID* of the selected result. For a *Search unit* the following additional properties can be defined: (i) a list of *displayed fields* (selectable from the fields mapped as *stored*) and an optional (ii) *result number* value, which specifies the top-*n* results to show; if such value is not defined, the unit will display all the result collection from the search engine. Figure 6 depicts an example of usage for the *Search* unit in our case study application. From the *Keyword Search* page, the parameters provided on the *Search Fields*'s outgoing links are used by the *Search Index* unit to perform a query on the index and to display its results. When the user selects a result from the *Search Result* page, the *OID* of the selected *Resource* entity is transported to the *Resource Detail* unit, allowing the user to visualize all the database information about the chosen resource. In this case the link passes information from the index to the database.

Another way for users to navigate through search results is by browsing them in a paged fashion, using commands for accessing the first, last, previous, next or *n*th element of a sequence. To model such interaction we borrowed from WebML the *Scroller* unit, adapting its behavior to cope with the new information sources offered by the index model: our *Search Scroller* operates on a given index, providing as output a set of *OIDs* for the document contained by the current displayed page. Figure 7 depicts an example of usage for the *Search Scroller* unit in our case study application: when the user issues a query, the *Result Scroller* provides

**Fig. 6.** Example of user to search engine interaction: result selection

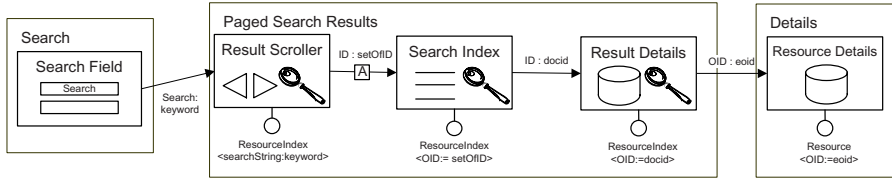


Fig. 7. Example of paged navigation through search results

to the *Search Index* unit the set of OIDs for the first page in the result collection. The user is able then to move through the retrieved results and, eventually, select one of them to display further details about it before continuing and show the actual data instance. This is accomplished by the *Document* unit, which, given a list of *displayed field* and an optional set of selector conditions, enables the visualization of a single document of an index without accessing the database.

Modeling operations on search engines. The second typical integration involving a search engine is its interaction with the Web application and its underlying data back-end. Therefore, there is the need to specify how the Web application acts as a broker between the search engine and the database in order to keep the content of the latter synchronized with the content of the former. We extend the WebML notation with three operation units, depicted in Figure 5: the (iv) *Indexer* unit, the (v) *Un-Indexer* unit and the (vi) *Re-Indexer* unit.

It has to be noticed how the proposed approach does not grant transactional features, which means that, for example, if the creation of a document in the index fails, the database and the index are out of synchronization and any further reconciliation activity must be modeled separately or performed manually.

The *Indexer* operation unit models the process of adding new content into an index to make it available for searching. The unit specifies as *source object* the desired index and accepts as input parameter the set of object identifiers for the entity instances to index. In addition, if the current index specifies some fields mapped as *unmapped*, the input parameters set is extended to allow the definition of the values for such fields. In our case study for example the *IndexingDate* is an unmapped field. For having the possibility to provide the current date during runtime, the set of input parameters is extended by one. The value of all the other fields is filled automatically according to the mapping rules, as specified in the index model mapping. Output parameters depend on the creation status; the OK and KO links will provide all the successfully created, respectively not created, document OIDs. Referring to the case study application, Figure 8 depicts a typical example of usage for the *Indexer* unit: a user, adding new resources into the data repository, has to provide the data to store in the database and to upload the physical file(s). This is modeled in WebML with an *entry unit* specifying a field for each entity attribute to fill. By navigating its outgoing link, the user activates the *Create* unit which creates a new entity instance in the database and eventually stores the provided files into the document base. If the

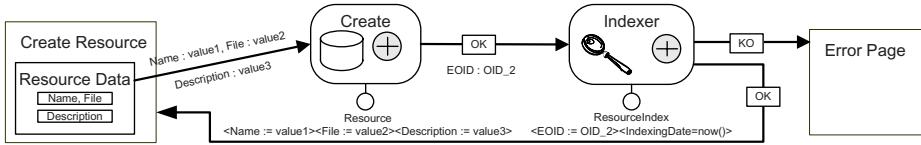


Fig. 8. Example of index document creation

creation succeeds, the OID of the created instance is provided as input to the *Indexer Unit* which takes care of the creation of the new index document. The success of the operation leads the user back to the *Create Resource* page, while its failure leads the user to an error page.

The process of removing content from an index and making it unavailable for search is modeled by means of the *Un-Indexer* operation unit. Like for the *Indexer* unit, this unit specifies as *source object* the required index. As input parameters the unit accepts a set of object identifiers for the entity instances associated to the index documents to remove from the index. Alternatively a set of document identifiers for the index documents can be used as input. In the first case the *Un-Indexer* unit searches inside the index and removes every entry referring to the entity instances' EOIDs. In the second case no search is performed and exactly the documents having the provided OIDs will be removed from the index. The *Un-Indexer* unit provides as output of its OK link the OIDs of all the documents successfully deleted while, on the KO link, the OIDs of the document for which a deletion was not possible. A typical example of usage of the *Un-Indexer* unit (shown in Figure 9) is the synchronization of the index after the deletion of a resource from the database. This scenario is modeled in WebML with an *index unit* showing the list of resources currently stored in the database. The index unit is linked to a *delete unit*, which is activated by navigating the outgoing link; after activation, the unit removes the entry from the database and also all the linked files from the repository. If the deletion succeeds, the OID of the deleted object is delivered as an input parameter to the *Un-indexer* unit, which, in turn, deletes all the documents referring to the provided EOID. If the deletion is successful the operation leads the user back to the *Delete Resource* page, otherwise it leads to an error page.

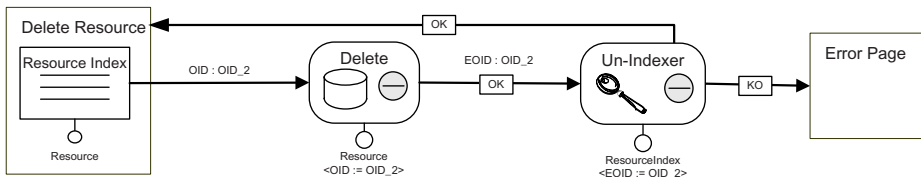


Fig. 9. Example of an existing index document deletion

Finally, the *Re-Indexer* unit allows to model the process of updating the content of index documents by re-indexing again their source data. This unit, given a source index to work on, accepts as input parameters the set of OIDs for the objects needing to be re-indexed. Because the re-indexing process is actually the same as the indexing one, this unit borrows from the *Indexer* unit the same input and output parameter specification. For space reasons we don't report an example of usage for the *Re-Indexer* unit which is anyway straightforward: this unit usually works paired either to one or more *Modify* or *Delete* units operating on the entities indexed directly (with mapped field) or indirectly (with context fields) by the search engine.

4 Implementation and Validation

Model extension, code generation, and run-time experiments were conducted in the context of the COOPER project, allowing us to successfully cover and validate all aspects of the outlined modeling approach. The modeling primitives discussed in the paper have been implemented in WebRatio [9], a CASE tool for the visual specification and the automatic code generation for Web applications. WebRatio's architecture consists of (i) a design layer for the visual specifications of Web applications and (ii) a runtime layer, implementing a MVC2 Web application framework. A code generation core maps the visual specifications (stored as XML) into application code exploiting XSL transformations. In our prototype, index model extensions have been realized by enriching the data model primitives offered by WebRatio with custom properties for the definition of indexes, fields and their respective mappings. Hypertext model primitives have been plugged into the existing architecture. We also created extensions to the code generator core of WebRatio in order to generate XML descriptors for the modeled indexes. Such descriptors have been used to instruct the search engine about mapping with the database and, hence, about how to retrieve and process the indexed data. Figure 10 depicts the new WebRatio run-time architecture. Search engine functionalities has been provided by exploiting Apache Lucene [10] as implementation technology. Lucene is an object oriented text search engine library written entirely in Java. We created custom classes for the automatic creation, management and maintenance of indexes based on the XML index descriptors produced by the code generator. In order to optimize the re-indexing processes, which may be time consuming, we have exploited advanced indexing techniques by creating ad-hoc parallel indexes for fields mapped as *Document* and *Context*. When modifications do not involve such fields, their associated information are not re-indexed, reducing the overall indexing time. The interaction between the WebRatio run-time and the index files has been developed inside the boxes of the MVC2 *Struts* framework, while concurrency and synchronization features over indexes have been left under the control of Lucene. Hypertext modeling primitives interact directly with the Lucene libraries in order to perform query and update operation over the content of indexes.

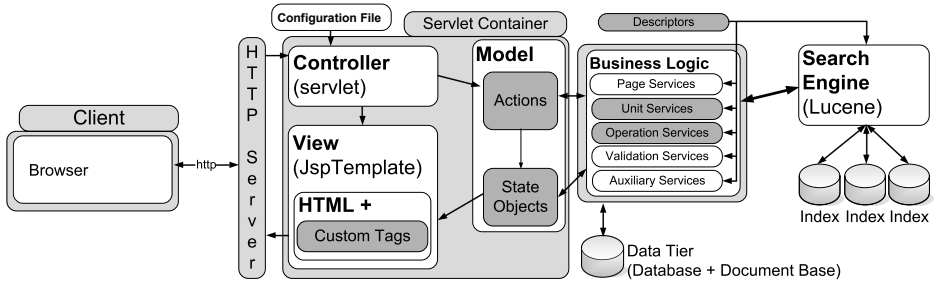


Fig. 10. Index data model for the case study application

5 Related Work

There are several approaches to overcome the limitations of not integrating IR techniques in database Web applications. Developers usually use back-end systems' built-in data retrieval features (based on standards query languages like SQL or XQuery) to enable content search into their Web sites. However, it is generally known how such approaches lack in flexibility due to exact-matching retrieval of data objects, which are often insufficient to satisfy users' information needs. The presence of additional resources along the structured data - which are only referenced in the database - exacerbates this problem and leads to the need for alternative solutions to cope with the limitations.

One possible solution is to integrate IR features into the data management systems by extending the existing models (relational, OO) to include traditional IR and user-defined operators. Several extensions for standard [11] [12] and extended [13] relational query languages have been proposed both in literature and by database producers [14]. This approach allows a seamless integration of IR features inside the robust environment offered by an RDBM system, which results in instantaneous and transitionally-aware updates on the internal indexes. However, such solutions do not implement all the functionalities offered by IR systems, and, since they do not rely on widespread adopted standards, they are usually implemented only by few vendors; moreover, pre-existing applications leveraging on traditional RDBMS products can not benefit on such extensions, leaving to developers the task of integrating external IR technologies. Other approaches deal with allowing keyword search on databases or on semi-structured data [15] [16] by annotating the database keywords with schema information and creating different query schemas. The problem is that this approaches work best for situations where there is not much text in the database and they provide un-ranked results sets. In our situation we consider both having textual information in the database and having external repositories.

Traditional solutions [17] integrate IR and database systems by exploiting a mediation layer to provide a unified interface to the integrated systems. According to [1], such solutions belong to the category known as loosely coupled (or middleware integration) systems. Our proposal also fits into such category but

we introduce a model-driven approach for the specification of a mediation layer actualized as a Web application by leveraging on WebML [8], one of a family [2] of proposals for the model-driven development of Web applications. For context fields mapping we referred to [6], which shows how we can index materialized views of semi-structured data. In this approach the terms are indexed along with the paths leading to them from the root node. The (term,context) pairs become the axes of the vector space. The resulting index is not merely a flat index, it also contains structure information because of the association between tags and their paths. The queries undergo the same operations. The method is closer to our needs, but still there is the problem of creating too many axes in the vector model, which we try to avoid by artificially recording the path information and index related information under a single axe as presented in [18].

6 Conclusions and Future Work

In this paper we have addressed the problem of the integration between search engine technologies and Web applications in a model driven scenario. We have proposed an extension for a specific modeling notation (WebML) and a development tool suite (WebRatio) to support the specification and automatic generation of search engine-aware Web applications. Such extensions involve (i) the definition of a novel conceptual schema (the index model), (ii) a set of rules to specify in a model-driven fashion the content of search engine indexes and (iii) novel modeling primitives for the specification of the integration between the database, the Web application and the search engine. Our future work will proceed along three directions: more complex hypertext modeling features will be considered, to widen the spectrum of interaction scenarios usually available for users (e.g. faceted search); extended index model primitives to support the definition of user policies for the access of indexed content; finally, further investigation will be addressed to apply the proposed approach to wider search scenarios, possibly involving multimedia content querying and retrieval.

References

1. Raghavan, S., Garcia-molina, H.: Integrating diverse information management systems: A brief survey. *IEEE Data Engineering Bulletin* (2001)
2. Gu, A., Henderson-Sellers, B., Lowe, D.: Web modelling languages: the gap between requirements and current exemplars. In: *AUSWEB* (2002)
3. Bongio, A., van Bruggen, J., Ceri, S., Matera, M., Taddeo, A., Zhou, X., et al.: COPPER: Towards A Collaborative Open Environment of Project-centred Learning. In: Nejd, W., Tochtermann, K. (eds.) *EC-TEL 2006*. LNCS, vol. 4227, pp. 1–4. Springer, Heidelberg (2006)
4. Baeza-Yates, R., Ribeiro-Neto, B.: *Modern Information Retrieval*. Addison-Wesley, London, UK (1999)
5. Navarro, G., Baeza-Yates, R.: Proximal nodes: A model to query document databases by content and structure. *ACM TOIS* 15, 401–435 (1997)

6. Carmel, D., Maarek, Y., Mandelbrod, M., Mass, Y., Soffer, A.: Searching xml documents via xml fragments. In: SIGIR (2003)
7. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for rdf data. In: ISWC (2006)
8. Ceri, S., Fraternali, P., Brambilla, M., Bongio, A., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, Seattle, Washington, USA (2002)
9. WebRatio: <http://www.webratio.com>
10. Apache Lucene: [http://lucene.apache.org/.](http://lucene.apache.org/)
11. Crawford, R.: The relational model in information retrieval. JASIST, pp. 51–64 (1981)
12. Vasanthakumar, S.R., Callan, J.P., Bruce Croft, W.: Integrating inquiry with an rdbms to support text retrieval. Data Engineering Bulletin (1996)
13. Ozkarahan, E.: Multimedia document retrieval. Information Processing and Management 31(1), 113–131 (1995)
14. Oracle: Oracle technical white paper (May 2001)
15. Hristidis, V., Papakonstantinou, Y.: DISCOVER: Keyword search in relational databases. In: Procs. VLDB, August 2002 (2002)
16. Weigel, F., Meuss, H., Bry, F., Schulz, K.U.: Content-Aware DataGuides: Interleaving IR and DB Indexing Techniques for Efficient Retrieval of Textual XML Data. In: McDonald, S., Tait, J. (eds.) ECIR 2004. LNCS, vol. 2997, pp. 378–393. Springer, Heidelberg (2004)
17. Grabs, T., Bm, K., Schek, H.J.: PowerDB-IR: information retrieval on top of a database cluster. In: IKE'01. Atlanta, Georgia, USA (2001)
18. Iofciu, T., Kohlschütter, C., Nejdil, W., Paiu, R.: Keywords and rdf fragments: Integrating metadata and full-text search in beagle++. In: Workshop on The Semantic Desktop at ISWC, Galway, Ireland (November 6, 2005)

A Method for Model Based Design of Rich Internet Application Interactive User Interfaces

M. Linaje, Juan C. Preciado, and F. Sánchez-Figueroa

Quercus Software Engineering. Universidad de Extremadura (10071 – Cáceres, Spain)
mlinaje@unex.es; jcpreciado@unex.es; fernando@unex.es

Abstract. During the last years, Web Models have demonstrated their utility facilitating the development of Web Applications. Nowadays, Web Applications have grown in functionality and new necessities have arisen. Rich Internet Applications (RIAs) have been recently proposed as the response to these necessities. However, present Web Models seem to be incomplete for modelling the new features appearing in RIAs (high interactivity, multimedia synchronization, etc). In this paper we propose a Model Driven Method, validated by implementation, called RUX-Model that gives support to multi-level interface specifications for multi-device RIAs.

Keywords: Rich Internet Applications, Models, Web Engineering.

1 Introduction

Web distribution architecture has inherited benefits such as low maintenance costs, decentralization and resource sharing among others. During the last few years, the growth of Web projects has brought about the development of models and techniques coming from the Web Engineering community. Nevertheless, the complexity of activities performed via Web interfaces keeps increasing, coming close to that of desktop applications. In this context, HTML-based Web Applications are showing their limits when considering high levels of interaction and multimedia support [5].

One solution to this limitation resides in Rich Internet Applications (RIAs) which combine the benefits of the Web distribution model with the interface interactivity of desktop applications. However, there is still a lack of complete development models and methodologies related to the new capacities offered by RIA [11]. As a result of RIAs technological characteristics and requirements, the current methodologies can not be directly applied to model and generate them. Most Web methodologies do not support multimedia properly, their focus being on data intensive Web applications (e.g., WebML [3], UWE [7] and OO-H [6]). In addition, most of the multimedia methodologies are not data intensive oriented and they focus on temporal specifications to support multimedia/animations and final-user interaction (e.g., HMT [2] and OMMMA [4]). According to previous studies, these methodologies do not cover RIA composition parameters fully at all [11].

To be more precise, Web modelling approaches can be extended towards RIA modelling in two different directions: 1) sharing business logic and establishing data

persistence between client and server sides; 2) extending Web User Interface (UI) capacities to specify the richness of presentation and interaction in RIA.

The first issue is beyond the scope of this paper and has been treated in [5]. For the second issue, lessons learned from Multimedia and Hypermedia fields are necessary.

The contribution of this paper is a Model Driven Method for the definition of rich UIs with high levels of user interaction and multimedia temporal relationships called RUX-Model (Rich User eXperience Model). This proposal is validated by implementation on RUX-Tool (the RUX-Model CASE Tool available at <http://www.ruxproject.org/>).

The rest of the paper is organized as follows: in section 2 the main design decisions for RUX-Model are shown and argued, after which in section 3 we describe RUX-Model in detail. Finally, conclusions and future work are outlined in section 4.

2 Design Decisions in RUX-Model

Due to RUX-Model being a multidisciplinary proposal and in order to decrease cross-cutting concepts, the interface definition is divided into levels. According to [14] an interface can be broken down into four levels, *Concepts and Tasks* (which has no bearing in RUX-Model, it correspond to the hypertext model level and is the starting point for RUX-Model), *Abstract Interface*, *Concrete Interface* and *Final Interface*. In RUX-Model each Interface level is composed by Interface Components.

For the ***Abstract Interface*** we are interested in models as independent as possible. RUX-Model Abstract Interface is partially based on Object-oriented Modelling of Multimedia Applications (OMMMA) [4] in order to define the set of media components. OMMMA is independent enough, but quite limited, so it must be improved by adding RUX-Model views and connector elements. The ***Concrete Interface*** in RUX-Model consists of spatial, temporal and interaction UI presentations.

Spatial Presentation.- There are many proposals to build a generic interface common to multiple devices and to customize it for specific devices [12]. Among them, our interest is focused on those languages being scalable and flexible enough to be extended with new features and, if possible, being standard. We have selected User Interface Modelling Language (UiML) [1]. UiML might be the most well-known and widely spread general XML-based UI description language. However, UiML is not enough to deal with some RIA necessities [10], so several extensions to UiML are proposed in RUX-Model.

Temporal Presentation.- The main interest is on the definition of temporal logic relations among Interface Components and the validation of the set of temporal relations allowed. In this sense, the Synchronized Multimedia Integration Language (SMIL) [15] is used to express temporal logic relations. Graphical commonly-adopted representation of SMIL is based on Petri Nets. RUX-Model uses Petri nets' structure as a bipartite directed multigraph. Timed Petri Nets (TOCPNs) are introduced for multimedia presentations in [8] and extended in DMPS [9], but they have mainly been used to give temporal functionality to static multimedia presentations of desktop systems. So, in RUX-Model graphical representation of Temporal Presentation definition is lightly inspired by sequence diagrams defined in OMMMA. The set of possible temporal relations in RUX-Model has been validated by correspondences of the set of temporal relations defined in HMT [2].

Interaction Presentation.- The proposal that covers part of our goal is XML Events [16]. The XML Events is an events syntax for XML that provides XML-based languages with the ability to uniformly integrate event listeners and associated event handlers with Document Object Model event interfaces. Handlers in RUX-Model are attached to the Temporal or Interaction Presentations. An event handler in RIA must be able to define the alterations of the presentation caused by the user interaction and the connections with the underlying business logic. UiML has a nice proposal to define handlers but it is not powerful enough to express RIA capabilities according to [10], so RUX-Model extends it and proposes a graphical representation over it to specify handlers in a visual way based on Flow Charts. In addition, OCL is used when low level detail operations are to be performed.

RUX-Model *Final Interface* describes the final UI according to RIA selected rendering technology. At the moment, the technologies considered in RUX-Model are Flex [17], Lazslo [18], and Ajax together with DHTML, but the model should be able to describe similar RIA rendering technologies (e.g., XUL or XAML).

Table 1. Summary of RUX-Model related technologies and their extensions

		Abstract Interface	Concrete Interface			Final Interface
			Spatial	Temporal	Interaction	
MODELS	Flow Charts	-	-	<i>Extended to represent graphically condition-action trees</i>		-
	HMT	-	-	<i>To validate the set of temporal relations</i>	-	-
	OMMMA	<i>Extended to define and group Media</i>	-	<i>Extended to perform sequence diagrams</i>	-	-
	TOCPN DMPS	-	-	<i>To express Temporal Logic Relations</i>	-	-
LANGUAGES	OCL	-	-	<i>Very low level specification of handlers</i>		-
	SMIL	-	-	<i>To express temporal logic relations</i>	-	-
	UiML	-	<i>Subset for Structure and Style of the UI</i>	<i>Extended to specify handlers</i>		-
	XICL	<i>Extended to define Interface Components</i>	<i>Extended to define and map Interface Components</i>			<i>Extended to define and map Interface Components</i>
	XML Events	-	-	-	<i>Listener definitions</i>	-
	AJAX,FLEX,LASZLO		<i>Used to define the set of Concrete Interface Components</i>			<i>Rich Render Technology</i>

Finally, glue between levels is needed. XICL (eXtensible user Interface Components Language) [13] is an extensible XML-based mark-up language for the development of UI components in Web systems and browser-based software applications. XICL allows not only the specification of components with their properties, methods and so on, but also the definition of target platforms and the mapping among Interface Components. RUX-Model extends XICL to define Interface Components and to establish mapping options between two components of adjacent Interfaces.

Due to the amount of RUX-Model related technologies, table 1 summarizes the models, methodologies and languages selected to be used or extended in order to conform parts of RUX-Model. We must clarify that RUX-Model does not depend on any of these models and languages and just takes advantage of these previous works.

3 RUX-Model Core

RUX-Model is an intuitive visual method that allows designing rich UIs for RIAs. Main RUX-Model features are:

- It defines reusable interfaces: RUX-Model Abstract Interface is reusable because it is common to all the RIA platforms, so all the devices that can run this kind of application may use the same Abstract Interface.
- It allows the spatial arrangement of the UI to be specified, as well as the look&feel of the interface elements based on a flexible set of RIA components.
- It models the temporal aspects, allowing the specification of those behaviours which require a temporal synchronization.
- It allows the user’s interactions with the RIA UI to be modelled based on the active Interface Components definition.

RUX-Model allows communication with the underlying hypertext application, hence data-content and functionality is offered by the HTML-based Web application. The final application’s functionality is thus dependent on the chosen Web model capacities.

Thus, RUX-Model is formed by the definition of a set of interfaces with distinctive responsibilities and abstraction levels (Figure 1).

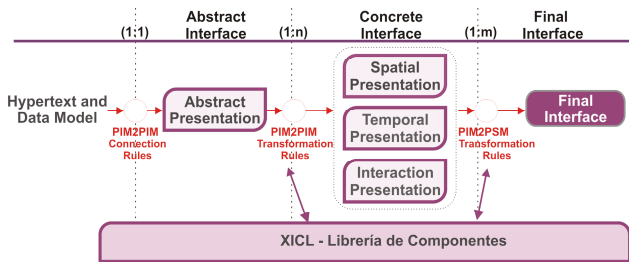


Fig. 1. RUX-Model MDA overview

The development process in RUX-Model has four main stages: connection with the previously defined hypertext model, definition of the Abstract Interface, definition of the Concrete Interface and the specification of the Final Interface, which ends in code generation. In RUX-Model each stage is fed by the previous one, as well as by the Component Library and the mapping possibilities specified in XICL (Figure 1).

The first stage in RUX-Model deals with the connection with the previous hypertext model (e.g., WebML, OO-H or UWE). At this stage, the presentation elements

and the relationships among them are extracted, as well as the defined operations on the Web model. The Connection Rules ensure RUX-Model to adapt to the system requirements shaped in the previous Web model. On the basis of the results offered by the Connection Rules, RUX-Model builds the Abstract Interface, by means of which we can specify an interface which is independent from the platform and the final display device.

From the Abstract Interface, through the application of the first set of Transformation Rules, the Concrete Interface, which allows the appearance, spatial arrangement, temporal and interactive behaviour, is obtained.

Finally, from the Concrete Interface and through the second set of Transformation Rules, we obtain the Final Interface, which has inherited, among others, the connection with the previous Web model.

Model Driven Architecture (MDA) is an approach to software development that provides a set of guidelines for structuring specifications expressed as models. Using the MDA methodology, system functionality may first be defined as a Platform Independent Model (PIM) through an appropriate Domain Specific Language. PIM may then be translated to one or more Platform Specific Models (PSMs) by mean of a set of Transformation Rules.

According to MDA approach, the relationship among each one of the stages of RUX-Model goes as follows (Figure 1): For each hypertext model you get an Abstract Interface (PIM). For each Abstract Interface you get n Concrete Interfaces (PIM). For each Concrete Interface you get m Final Interfaces (PSM).

Thus, by using RUX-Model it is only necessary to specify a common Abstract Interface for the different Concrete Interfaces. Hence, every Concrete Interface can be specified in order to be rendered in one or more specific platforms. The application building process finishes with a stage in which the transformation engine generates the code dependent on the chosen target specific platform.

3.1 Connection Rules

By defining a certain set of rules, called Connection Rules, we ensure the RUX-Model connection integrity with the previous Web model. The set of Connection Rules establishes the way the matching takes place among the elements in the previous Web model and the Abstract Interface.

RUX-Model extracts (from the Web model it connects) the structure and navigation in order to use them to create an initial Abstract Interface model, and to grant the Concrete Interface model access to the “*operation chains*” or “*operation points*”, which represent the operational links in the hypertext navigation models.

The process starts when we choose the set of Connection Rules to be used, which are defined specifically regarding the previous Web model taken.

The following is extracted from the previous Web model:

- The data used by the Web application and its relationships.
- The existing hypertext element groupings.
- The connections between pages, which allow us to put the application in context.

3.2 Transformation Rules and the Component Library

Following MDA, we will focus on defining models and specifying Transformation Rules in order to systematize the RIAs construction process.

In order to understand the way the Transformation Rules are applied in the steps PIM2PIM and PIM2PSM (Figure 1), we need to know the specifications for the Interface Components involved in each of the RUX-Model Interface levels. In RUX-Model an Interface Component is defined by its name, its identifier (required) and a set of properties, methods and events, which are optional. Thus, RUX-Model is made up among others of Abstract Interface Components, Concrete Interface Components and Final Interface Components.

With the aim of making the access and the maintenance of the Interface Components easy, RUX-Model specifies a Component Library, which is responsible for: 1) storing the components specification (name, properties, methods and events) 2) specifying the transformation capabilities for each component from an Interface level in other components in the following Interface level and 3) keeping the hierarchy among components at each Interface level independently to every other level.

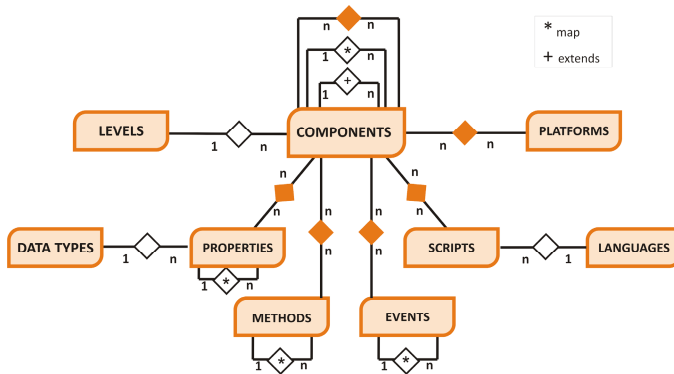


Fig. 2. Component Library E-R description

The Component Library specified in Figure 2 maintains the specifications of the Interface Components and their relationships along the modelling process. The set of Interface Components defined in the library can be increased or adapted by the modeller to its needs and according to the specifications of the project. Properties, methods and events of each Component can be extended at any time.

The transformation specifications contained in the Component Library are added to our own XICL extended specification (see <http://ruxproject.org/xiclxt.pdf> for further information) in order to declaratively specify which are the transformation options available to be carried out on each Interface Component.

XICL extensions allow RUX-Model to keep the definition of several target platforms in a single XML document, as well as the translation of an origin component to one or more target platforms when we specify PIM2PSM Transformation Rules. This is a basic capability of the RUX-Model Transformation Rules not existing in XICL that was designed primarily to create HTML components. XICL mixes spatial

arrangement and interaction, which is not a desirable feature in our model. XICL original component tag has been extended to allow several platforms and reference each component to the platform it belongs to. Also the properties declaration has been modified and unused elements and attributes have been removed. Moreover, a strict key reference definition has been added to XICL to facilitate checking the Transformation Rules.

Abstract to Concrete Interface (PIM2PIM): PIM2PIM Transformation Rules establish the correspondences which are allowed among Abstract Interface Components and among Concrete Interface Components. The set of different components which can form the Abstract Interface is fixed and limited by a number of elements set by RUX-Model as native (mentioned later), even though it is conceptually extendable extending the Component Library. The size of Concrete Interface Components set is variable and depends on the number of RIA elements to be defined dynamically in the Component Library. Applying these rules on the Abstract Interface allows creating a first version of the Concrete Interface.

Concrete to Final Interface (PIM2PSM): This case is different from PIM2PIM in the fact that now both the set of origin components (Concrete Interface) and the set of target components (Final Interface) are changeable and dynamic within the Component Library. In the Component Library, the Concrete Interface Components are defined and we detail a series of final platforms. The methods, events and properties mapping is done individually by each component and target platform that we want to deploy in the Final Interface level. For instance: we define our native Concrete Interface Component called *tabnav*, which could become the *TabNavigator* FLEX component or the *tabpane* Laszlo component.

3.3 Abstract Interface

Abstract Interface lets us obtain an abstraction on the chosen Web model, obtaining a common representation for the hypertext systems currently used in the Web.

This Interface can be initially obtained from the selected Web model by using the Connection Rules. A refinement process is allowed, in which the modeller can add and/or restructure the Abstract Interface if it is considered necessary until the modeller’s goal is achieved.

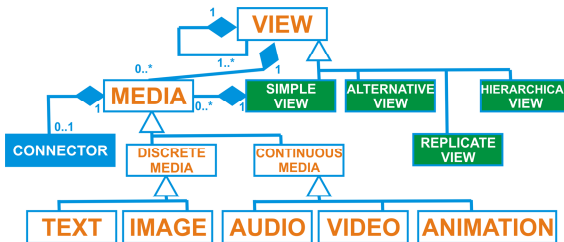


Fig. 3. RUX-Model Abstract Interface formal description

RUX-Model Abstract Interface is composed by connectors, media and views. Figure 3 shows the Abstract Interface elements and the relations among them.

1) **Connectors**. They are related to the data model once it is specified how they are going to be recovered in the hypertext model. Web development methodologies typically give the hypertext model some characteristics which allow, for instance, specifying the way a set of registers recovered from a database is going to be ordered and so on. Each connector owns an attribute called “*sourceid*” among others which identifies its connection with the underlying Web model.

2) **Media**. They represent an atomic information element that is independent of the client rendering technology.

The availability of media components and their grouping is based on OMMMA. In RUX-Model they are categorized into discrete media (texts and images) and continuous media (videos, audios and animations). Each media gives support to Input/Output processes, so it is possible to design multimodal RIAs’ UIs. This is necessary since RIA allows audio/video inputs over the Web.

Common media attributes are “*source*” and “*connectorid*”. These attributes are used to define dynamic data source of the media (from the underlying Web model).

3) **Views**. In the Abstract Interface, a view symbolizes a group of information that will be shown to the client at the same time. In order to group information, RUX-Model allows the use of four different types of containers: simple, alternative, replicate and hierarchical views. In RUX-Model, the root element of the Abstract Interface is always represented by a view.

3.4 Concrete Interface

The Concrete Interface specifies the presentation and behaviour of the UI that has to be modelled, depending on how we want the application to be. The UI of the Concrete Interface deals with a UI common to all the current RIA final rendering technologies and it is formed by three additional “presentations” in order to minimize cross-cutting: Spatial, Temporal and Interaction Presentations. In order to offer the modeller intuitive visual models for the different presentations, RUX-Model provides each one of them with their own intuitive graphic representation that is backed by modelling languages or representations based on standards.

3.4.1 Spatial Presentation

The Spatial Presentation in RUX-Model is responsible for specifying over the Abstract Interface the positioning, dimension and look&feel of the Interface Components in the application space.

RUX-Model gives the modeller a set of native RIA components (see <http://ruxproject.org/nativecomp.pdf> for further information) which have been defined on the basis of a component set which is currently present in nearly all current RIA rendering platforms. The choice of these components is the fusion among the groups of components in the platforms Laszlo, Macromedia Flex, Macromedia Flash and XUL that are well-known RIA development platforms. RUX-Model Concrete Interface Components can be classified in three categories: Controls (components used to gather the user entries or to provide the user with output information), Layouts

(components able to distribute and/or gather the elements in the Concrete Interface) and Navigators (components able to provide navigation among the stack layout and other typical navigation components).

As we have mentioned before, the Component Library is responsible among other things for keeping the Interface components specification, both the native Concrete Interface ones and those new ones that modellers want to add. The RIA native set of components has been specified in RUX-Model only to ease the modeller's work.

Additionally there are descriptions in two declarative languages: the first one, the XICL extension proposed which has already been commented on, and the second one, the usage of dynamic schemes extracted from the Component Library to maintain the hierarchical consistency among the components that compose the spatial presentation.

The spatial presentation specification is simple from the point of view of the modeller, given that the grouping is given first by the Abstract Interface, so the modeller just has to refine the grouping, place the components spatially, and define their dimensions and look&feel.

The textual specification of the RUX-Model spatial presentation RUX-Model uses a subset of the specification proposed by UiML, which is formed by part of the *children* defined in UiML inside the node <interface>. This includes the nodes <structure> and <style>, avoiding the use of <content> and <behavior> nodes.

3.4.2 Temporal Presentation

The goal of the RUX-Model Temporal Presentation is to represent those behaviour which take place without the direct mediation of the user and establish a series of temporal relationships between the different Interface Components in the presentation. It is very useful in order to represent behaviours that take place in a temporal way on the Concrete Interface Components or to specify calls to the underlying business logic in a predefined way based on time events.

The Temporal Presentation is defined on the Spatial Presentation in order to be able to specify how changes affecting the RIA UI will take place.

The temporal presentation is made up of the temporal presentation elements, which are Concrete Interface Components or groups of Concrete Interface Components, whose spatial presentation is already defined.

It is possible to establish the temporal relationships logic between the elements (E) of the temporal presentation and also group temporal presentation element (G) which can contain one or more temporal elements in order to share a temporal logic.

In this sense, to illustrate better the implied processes in temporal relationship logic, we define some concepts implied in this logic.

Real Using Time: linearly elapsed time from when the application starts running in the client until the client finishes the application.

Predefined Using Time: time used to specify the duration of a temporal behaviour established on one or more temporal presentation elements.

The Real Using Time will always be equal to or higher than the Predefined Using Time, given that it is possible for a user to pause, play and restart the temporal behaviour in a temporal presentation element.

Moment: We can at any time “take a picture” of the temporal state of the components in the presentation, defining a precise moment of the Real Using Time. Every *moment* is defined according to the temporary situation at a given instant of all the temporal presentation elements.

Temporal Presentation definition takes place through triads of the kind $E[E':V_0, E'':V_F, \{handler:V_E\}]$ and is graphically represented by an extended sequence diagram that expresses how the elements behave in time and how these behaviours affect other elements. In this triad the values are referred to as follows:

E: Temporal Presentation element target of the indicated temporal logic.

E': Temporal Presentation element with which E is related in order to start its temporal behaviour in a synchronized way.

V₀: Delay in the start of element E regarding the start of element E'. It can contain a real value.

E'': Element with which E is related to finish its temporal behaviour in a synchronized way.

E' and E'' can refer or not to the same presentation element.

V_F: Delay in the end of element E regarding the end of element E''. It can contain:

END: Indicates that it ends when *Real Using Time* finishes.

Real Value: Indicates the time units the temporal presentation element must last.

Handler: Reference name for a handler in a set of defined handlers in RUX-Model.

V_E: Delay when launching the handler from the beginning of element E. It can contain a real value.

Related to the couple **handler:V_E** :

- 1) it can be repeated by modifying one or both values as many times as necessary, in order to indicate the launching of different handlers at different *moments* in the timeline E is acting in.
- 2) taking into account that the couple is compulsorily tied to a predefined temporal event E and gets affected by its temporal definition, the launching of the handler is also affected by such situation in a way that, if E is repeated in time, the launching of the handler will be too.

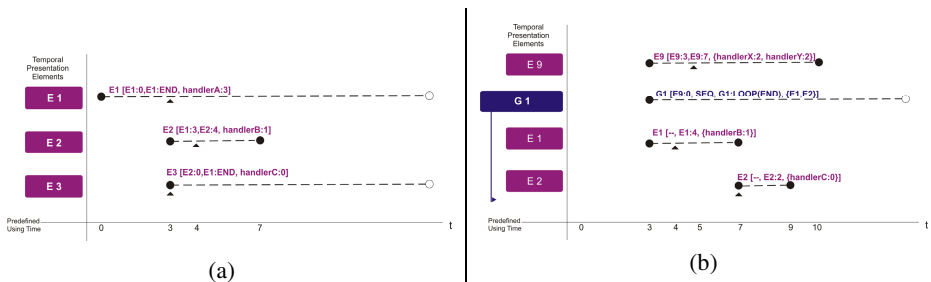


Fig. 4. Temporal Presentation description examples

The graphic representation of the temporal relationships logic is depicted in Figure 4a, where the element E1 starts just when the application begins to run and ends when the

application finishes (here *Real* and *Predefined Using Time* are equal). *Predefined Using Time* of element E2 starts when E1 starts plus a three delay time units from E1 and ends four time units later. E3 is implicitly executed simultaneously to E2 for four time units and finishes at the same time as the *Predefined Temporal Time* of E1.

In RUX-Model temporal representation diagram, the temporal presentation elements are placed on the left, while the temporal relationships logic that affects those elements is placed on the right. For each temporal presentation element in the diagram there is a temporal sequence that starts and finishes in a black dot.

In case it being interesting to focus attention on just a part of the temporal diagram and there are other elements in the segment starting or finishing outside the shown range, this is shown by means of a white circle. This means that the element or group starts before or finishes after, depending on the situation, the shown *Predefined Using Time* range. Each handler launched by the temporal behaviour is symbolized in the diagram by a triangle.

The definition of temporal representation on presentation element groups is done by quartets of the kind $G[E':V_0, M, E'':V_F, \{\text{elements}\}]$. These quartets are graphically represented in the extended sequence diagram and express the behaviour of elements in time and how these behaviours affect other elements or groups of elements. In the indicated quartet the values are referred to as follows:

G: Group of temporal elements, target of the indicated temporal logic.

E': Temporal presentation element related with G to initiate its temporal behaviour in a synchronized way.

V₀: The initial delay of element G with respect to the start of element E', it can contain a real value.

M: Execution mode, which can be PAR, parallel (default value), or SEQ, sequential. Like SMIL, RUX-Model defines mainly two kinds of grouping: Parallel (PAR - plays child elements in parallel) and Sequential (SEQ - plays child elements in sequence).

E'': Element related with G to finish a temporal behaviour in a synchronized way.

V_F: Delay in the end of G regarding the end of element E'', which can contain the following value:

LOOP(n): Indicates that the group temporal logic will be repeated *n* times. *n* can also take the value *END*, thus indicating that it will be repeated until the application stops running.

In Figure 4b an example of grouping is depicted where E9 begins running in the moment three of the *Predefined Using Time* and finishes seven time units later. E9 also triggers two handlers in the instant two from E9 runs. This temporal diagram also specifies that G1 begins running when E9 begins and defines a temporal sequence relation between E1 and E2, looping until the end of *Real Using Time*.

The temporal relationships logic specifies the temporal behaviour of the Concrete Interface Components. For the attributes of a presentation element implied in a temporal behaviour the values wanted for the initial moment and those for the temporal elements in the final moment of the behaviour must be indicated. It is possible to associate values for the attributes of a presentation element related to the values of the attributes that other temporal presentation elements have in any *moment*.

3.4.3 Interaction Presentation

The objective of the Interaction Presentation in RUX-Model is to capture and represent those behaviours triggered by the user interaction. As in previously shown RUX-Model phases, we have provided an intuitive graphic notation and a strong definition language at this point.

In RIAs, capturing the user interaction with the UI is generally carried out by the application components that are able to capture certain event types.

In RUX-Model, an event is the representation of some asynchronous occurrence (such as a mouse click on the presentation of the element) that gets associated with a Concrete Interface Component. An action is some way of responding to an event; a handler is some specification for such an action; a listener is a binding of such a handler to an event targeting some component.

As the event capture definition language, RUX-Model uses XML Events, W3C Standard language for the event capture specification. The XML Events module provides RUX-Model with the ability to uniformly integrate event listeners and event handlers.

XML Events is not limited to expressing conditions such as “if windows1 moves launch action1”. XML Events ease richness in expression in DOM based languages, which allows the event definition to affect all the children of the window (e.g., all the components previously defined in RUX-Model within a *windowlayout*, such as *buttoncontrols*, could get affected by a “drag” type event assigned to the father). In RUX-Model all events are declared globally in order to take advantage of the way events propagate.

XML Events is based on the definition of “listeners” that are used to declare event listeners and register them with specific nodes in the DOM. It has several attributes: 1) event: the event type for which the listener is being registered; 2) observer: the id of the element with which the event listener is to be registered; 3) target: the id of the target element of the event; 4) handler: the URI of a resource that defines the action that should be performed; 5) phase: when the listener will be activated by the desired event (capture or bubbling phase); 6) propagate: whether after processing all listeners at the current node, the event is allowed to continue on its path (continue or stop); 7) defaultAction: whether after processing all listeners for the event, the default action for the event (if any) should be performed or not (perform or cancel); 8) id: event identifier. It’s important to note that XML Events does not specify available event types or actions. This feature is really important in RUX-Model, since event types are declared in the dynamic Component Library.

The graphic notation of a listener is quite simple and it is related with the spatial presentation since events are registered over Concrete Interface Components. Over the spatial presentation, it a shape is depicted that contains the handler name and the event selected. In order to represent the relationship between the observer and the handler a line is also drawn to connect them.

3.4.4 Handlers

Once the way the events are captured, defined and integrated with the handlers is specified, this section focuses on handler specification.

Handlers in RUX-Model follow an Event-Condition-Action (ECA) model. Each ECA is defined by the Event (in RUX-Model defined in the listener declaration) and a

Condition-Action-Tree (CAT) defined in the handler. With ECA, RUX-Model is able to define rules with actions that are triggered when some conditions are met. This behaviour model is more than enough to express both changes in the UI and calls on the business logic expressed in the defined Web Model.

In RUX-Model, handlers are textually represented in a declarative way. The specification is based on a subset of UiML “behaviour” module that is composed of the following basic elements: 1) behaviour: the behaviour root element; 2) rule: this element defines a binding between conditions and actions; 3) condition: element that contains a logical expression based on the `<op>` element. `<op>` may also contain hierarchical `op` elements to compose more complex conditions. UiML `<op>` available logical conditions are `==`, `!=`, `>`, `<`, `&&` and `||`. `<op>` available operators are constant, property, reference, call, `op` and event; 4) action: element that contains one or more elements that are executed in the order they appear in the UiML document. Actions in UiML may carry out procedure calls (`<call>` element), changes on properties (`<property>` element) and other actions like UI restructuring, or event activations (`<restructuration>` and `<event>`). Additional action subelements are `<whentrue>`, `<when-false>` and `<by-default>` conditions.

As we have introduced previously, UiML is not sufficiently powerful to represent RIA and therefore at this point an explanation is given for each extension proposed to adopt UiML as part of RUX-Model syntax.

Concerning rules, RUX-Model uses the UiML “*id*” attribute as handler name for the binding between temporal/interaction presentations and actions to be evaluated and performed. On the one hand, `<event>` element in UiML conditions is less expressive than XML Events listeners, so RUX-Model does not use UiML `<event>` elements (this has been explained inside the Interactive Presentation section). On the other hand, UiML `<op>` element is powerful enough to support the conditions that are necessary to express conditions for actions in RUX-Model, so RUX-Model maintains UiML definition of conditions. Since UiML `<equal>` element is equivalent to `<op name=”=”>` and is only supported in UiML for legacy descriptions, we do not adopt it as part of RUX-Model.

To define actions in RUX-Model, we use a subset of UiML `<action>` element. In RUX-Model there are two kinds of actions: those which modify the UI and those that affect the underlying web business logic. For those actions that affect UI, we avoid using `<restructuration>` element to change UI. RUX-Model at this point is richer than UiML proposal, since RUX-Model facilitates not only property changes available in UiML but also the definition of transitions... calling the Temporal Presentation specified by the RUX-Model.

For those actions that affect the underlying business logic, we use UiML `<call>` element. The name of the procedure and a set of arguments is the typical syntax to call a function in procedural languages and that is the way UiML does it using `<param>` elements. However, RUX-Model needs to communicate asynchronously with the Web model in the way a hypertext page communicates with another one, using GET or POST request methods by a pair of name/value. RUX-Model extends UiML call parameters adding the attribute “*id*”, “*name*”, “*uri*” and “*type*” (request method) to the `<param>` element. UiML `<action>` declaration in actual version (v.3.1) also includes conditions (`<when-true>`, `<when-false>` and `<by-default>`). As we need to

extend UiML, we also improve this feature, taking these conditions not only as <action> children but also as <action> parents.

The proposed ECA model represents in a declarative way the system of conditions and actions of RUX-Model, and helps to solve the majority of the problems with handlers regarding the triggering of operation chains and Spatial Presentation changes. Additionally, using OCL syntax, RUX-Model provides those models requiring an extra expressivity out of the CAT usage domain provided by ECA.

RUX-Model also has a graphical description of handlers in order to ease the modeller's work. RUX-Model has an intuitive diagram for the visual specification of the CAT, inspired in the flow charts visual representation (visit <http://ruxproject.org/catelements.pdf> to see the elements of the CAT diagrams).

3.5 Final Interface

The last process that takes place in RUX-Model is related with the translation of the Concrete Interface to the Final Interface. This last level of the model is defined depending on the specific platform to which we want to associate the whole design. It is closely related and depends on the components registered in the Component Library for each final rendering platform.

Thus, the correspondences that the modeller wants to establish among the Concrete Interface components and the Final Interface component are decided in the Final Interface. Hence, in order to generate the application depending on a specific platform, we have to translate the design carried out in the Concrete Interface into one of the different rich rendering platforms previously defined in the Component Library (e.g., Laszlo, Flex, AJAX, XAML).

Thanks to the fact that the Spatial, Temporal and Definition presentation techniques integrated in the Concrete Interface design allow a textual representation based on XML, it is easier to process them by means of a translation engine in order to obtain the Final Interface. This translation can be automated depending on the options chosen by the modeller and based on a set of Transformation Rules. These rules establish the correspondences allowed among each element in the source presentation model and the target presentation model that we want to obtain.

The formal definition of those models mentioned in these Transformation Rules grants the integrity of the final result in the transformation process.

4 Conclusions and Future Work

This paper introduces RUX-Model, a Model Driven Method for the systematic design of RIAs UIs over existing HTML-based Web Applications in order to give them multimedia support, offering more effective, interactive and intuitive user experiences.

Conceptually, RUX-Model can be used on several Web development models/methodologies. At the implementation level, RUX-Tool has a series of prerequisites about the models that can be used in order to extract from them all the information stored by these models automatically. Currently, RUX-Tool works together with WebRatio (<http://www.webratio.com/>) the WebML CASE Tool, but there is a work in progress with UWE and OO-H CASE Tools.

Once RUX-Model has been defined, future work will deal with the definition of extensions on current Web models in order to be able to represent the distribution of data and business logic between server and client as well as the specification of the new communication models that have appeared related with RIAs. This will make possible to model RIAs with more complex specific capacities that cannot be properly modelled today.

Acknowledgements

PDT06A042 and TIN2005-09405-C02-02 projects.

References

1. Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M., Shuster, J.E.: An appliance-independent XML language, *Computer Networks. The International Journal of Computer and Telecommunications Networking* 31, 1695–1708 (1999)
2. Specht, G., Zoller, P.: HMT: Modeling Temporal Aspects in Hypermedia Applications. In: 1st International Conference on Web-Age Information Management, pp. 256–270. Springer-Verlag, Heidelberg (2000)
3. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: *Designing Data-Intensive Web Applications*. Morgan Kaufmann, Seattle, Washington, USA (2002)
4. Sauer, S., Engels, G.: Extending UML for Modeling of Multimedia Applications. In: IEEE Symposium on Visual Languages, IEEE Computer Society, p. 80 (1999)
5. Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G.: Conceptual Modeling and Code Generation for Rich Internet Applications. In: International Conference on Web Engineering, pp. 353–360 (2006)
6. Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.: The OO-Method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. *Information Systems* 26(7), 507–534 (2001)
7. Koch N., Kraus A.: The Expressive Power of UML-based Web Engineering, *Int. Wsh. Web-Oriented Software Technology*, pp. 105–119 (2002)
8. Yoon, K., Berra, P.B.: TOCPN: interactive temporal model for interactive multimedia documents. In: International Workshop on Multi-Media Database Management Systems, pp. 136–144 (1998)
9. Shih, T.K., Keh, H., Deng, L.Y., Yeh, S., Huang, C.: Extended Timed Petri Nets for Distributed Multimedia Presentations. In: Proceedings of the 15th International Parallel & Distributed Processing Symposium, p. 98. IEEE Computer Society, Los Alamitos (2001)
10. Mueller, W., Schaefer, R., Bleul, S.: Interactive Multimodal User Interfaces for Mobile Devices. In: 37th Annual Hawaii International Conference on System Sciences, vol. 9. IEEE Computer Society, Los Alamitos (2004)
11. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S.: Necessity of methodologies to model Rich Internet Applications. In: IEEE Internat. Symposium on Web Site Evolution, pp. 7–13 (2005)
12. Luyten, K.: *Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development*, University Limburg: School of Information Technology (2004) <http://research.edm.luc.ac.be/kris/research/phd/>

13. de, S.G., Leite, J.C.: XICL - an extensible markup language for developing user interface and components. In: Fourth International Conference on Computer-Aided Design of User Interface. vol. 1 (2004)
14. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V.: UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In: Bastide, R., Palanque, P., Roth, J. (eds.) Engineering Human Computer Interaction and Interactive Systems. LNCS, vol. 3425, pp. 207–228. Springer, Heidelberg (2005)
15. SMIL W3C (2005) <http://www.w3.org/TR/2005/REC-SMIL2-20051213/>
16. XML Events W3C (2003) <http://www.w3.org/TR/2003/REC-xml-events-20031014/>
17. Adobe RIA: <http://www.adobe.com/devnet/ria>
18. Open Laszlo: <http://www.openlaszlo.org>

Improving Communication in Requirements Engineering Activities for Web Applications

Pedro Valderas and Vicente Pelechano

Department of Information System and Computation.
Technical University of Valencia, Spain
Cami de Vera s/n 46022
{pvalderas, pele}@dsic.upv.es

Abstract. We present a requirements engineering environment which provides techniques and tools to improve communication in Requirements Engineering activities. First, a technique based on requirements ontologies is proposed to allow customers to describe their needs. This technique is supported by a tool. This tool provides analysts with structured descriptions of the customers' needs that facilitate analysts to understand the problem to be solved. Next, both a model-to-text transformation and a model-to-model transformation are introduced to automatically obtain a textual requirements specification and a task-based requirements model respectively. The textual specification facilitates customers to validate requirements. The task-based requirements model facilitates programmers to interpret the requirements specification.

1 Introduction

We present a Requirements Engineering (RE) environment for Web applications that provides techniques and tools to improve the process of communication among customers, analysts and programmers. First, this environment introduces a tool which makes customers a set of questions throughout a guided process. This tool analyzes the information provided by customers in order to obtain a structured description of their needs. To do this, the tool uses a set of *requirements ontologies*. The obtained description of the customers' needs is clearly defined by means of concepts of the requirements ontologies which facilitates analysts to understand it. Furthermore, the proposed RE environment introduces two transformations: (1) A *model-to-text transformation* which transforms the ontology-based description of the customers' needs into a requirements specification defined in natural language. This aspect facilitates customers the validation of requirements. (2) A *model-to-model transformation* which transforms the ontology-based description of the customers' needs into a requirements model based on the concept of task. This aspect facilitates programmers to interpret the requirements specification.

The rest of the paper is organized as follows: Section 2 introduces both the concept of requirements ontology and the tool which uses them. Section 3 introduces the model-to-text transformation. Section 4 introduces the model-to-model transformation. Finally, conclusions are comment on in Section 5.

2 Facilitating Customers to Describe Their Needs

We introduce next a strategy which facilitates customers to describe their needs. It is based on two elements: (1) Requirements ontologies and (2) a tool which asks customers for their needs by using these ontologies. We present next both elements.

2.1 Requirements Ontologies

A *Requirements Ontology* specifies the concepts and the relationships between concepts that represent a Web application of a specific type (E-commerce applications, web portals, directories, etc). Figure 1 shows a partial view of the requirements ontology for E-commerce applications. This ontology defines concepts such as *On-Line Purchase*, *Shopping Cart*, or *Products* (concepts that characterize E-commerce applications).

To define ontologies of this kind, we use the approach presented in [1]. According to this approach, two kinds of concepts can be defined, namely lexical concepts (enclosed in dashed rectangles) and nonlexical concepts (enclosed in solid rectangles). A concept is *lexical* if its instances are indistinguishable from their representation. *Date* (see Figure 1) is an example of lexical concept because its instances (e.g. “21/05/2005” and “04/09/2004”) represent themselves. A concept is nonlexical if its instances are object identifiers, which represent real-world objects. *User* (see Figure 1) is an example of nonlexical concept because its instances are identifiers such as “ID1”, which represents a particular person in the real world who is a user. The main concept in a type ontology is marked with “->•”. We designate the concept *On-line Purchase* in Figure 1 as the main concept because it represents the main purpose of an E-commerce application.

Figure 1 also shows a set of relationships among concepts, represented by connecting lines, such as *Product has Property*. The arrow connection represents a one-to-one relationship or a many-to-one relationship (the arrow indicates a cardinality of one), and the non-arrow connection represents a many-to-many relationship. For instance, *Auction offers Item* is a many-to-one relationship (i.e. in each auction only an item can be offered but an item can be offered in several auctions) and *Product has Property* is a many-to-many relationship (i.e. a product can have several properties, and a property can be defined for several products). A small circle near the source or the target of a connection represents an optional relationship. For instance, it is not obligatory for a category to belong to another category. A triangle in Figure 1 defines a generalization/specialization with a generalization connected to the apex of the triangle and a specialization connected to its base. For instance, *Direct Purchase* is a specialization of *On-Line Purchase*.

Finally, we have extended this notation by introducing *abstract concepts*. An abstract concept is a concept that depends on the domain of the Web application and need to be instantiated. These concepts are marked with a vertical line on the right side (see Figure 1, concepts *Product*, *Property* and *Category*). For instance, *Product* is an abstract concept because we know that every E-commerce application must allow users to purchase products; however, we do not know what kind of products they are (they can be CDs, Books, software, etc.). This information depends on the E-commerce application domain and must be instantiated by customers. To do this, a tool has been developed. It is introduced in the next section.

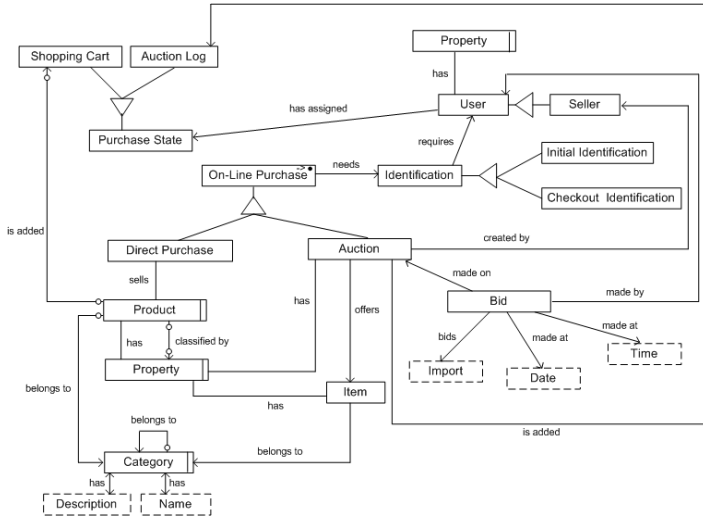


Fig. 1. Requirements Ontology for E-commerce applications

2.2 A Web Application Requirements Elicitation Tool

In this section, we introduce a requirements elicitation tool that supports customers in the description of their needs by means of requirements ontologies. To do this, the tool perform three main steps:

First, the tool allows customers to describe the Web application that they need by using natural language. This description is used by the tool to know the general requirements of the web application and then to select the proper requirements ontology. To do this, we use a technique based on data frames [2]. The data frame approach allows us to describe information about a concept by means of its contextual keywords or phrases, which may indicate the presence of an instance of the concept. We define data frame contextual information for each Web application type that is represented by a requirements ontology. The tool uses this contextual information to recognize the web application type and then select the proper ontology.

Next, the tool must obtain the information that cannot be systematically extracted from a requirements ontology in order to obtain the description of the customers' needs. This information is related to domain-dependent features such as for instance the kinds of products that must be on sale in an E-commerce application (e.g. CDs, DVDs, Books, etc.) (Abstract concepts, see Figure 1). This information must be introduced by customers. To do this, the tool provides them with an appropriate interface. For instance, Figure 2 shows the HTML interface that allows customers to determine which products must be on sale in the running example.

Finally, the information introduced by customers, together with the general features of the application domain (defined in the requirements ontology), allow the tool to obtain a description of the Web application that customers need. This description is defined as a view over the selected requirements ontology where abstract concepts

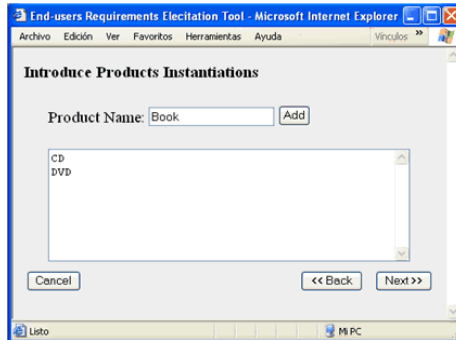


Fig. 2. HTML interface

(e.g. *Product*) are replaced by their instantiations (e.g. *CD*) and relationships among abstract concepts are replaced by relationships among instantiations (e.g. *Product has Property* has been replaced by *CD has Title*). These descriptions are stored in OWL.

3 Obtaining Textual Requirements Specifications

In this section, we introduce a model-to-text transformation that allows analysts to transform the Web application description based on ontology concepts into a textual requirements specification. Furthermore, each textually specified requirement is complemented with a list of real examples where customers can see an implementation of it. This list of real examples facilitates customers both to understand the requirements and to check that it is really what they need.

```

1 <xsl:template match="owl:Class" name="FindShoppingCart">
2 <xsl:variable name="concept" select="@rdf:ID" />
3 <xsl:if test="$concept='Shopping_Cart'">
4 <b>Requirement Number {$num_requirements}</b>
5 <u>Name:</u> Shopping Cart.
6 <u>Description:</u> The E-Commerce Application must allow
7 users to add products to a Shopping Cart. A shopping cart
8 is a 'persistent' store for products that can be accessed
9 from the whole Web application. The option of 'add to
10 shopping cart' is attached to each product in order to
11 allow users to add it. Furthermore, other operations are
12 associated to manage the cart such as eliminating a product,
13 changing quantities, making an order, etc.
14 <xsl:call-template name="ShoppingCartRealExamples" />
15 </xsl:if>
16 </xsl:template>

```

Fig. 3. Example of a XSL Transformation

To achieve this, we have implemented a set of XSL Transformations which take as source the ontology-based description of the customer needs and then create the corresponding textual requirements specification. The XSL Transformations are created in order to match with the concepts and relationships between concepts that appear in the description of the customer's needs. Figure 3 shows the XSL Transformation that generates a textual requirement specification from the concept "Shopping Cart". In order to better understand it we must know that concepts are represented in OWL by means of the label `owl:Class` and the name of each concept is defined by the attribute `rdf:ID`.

The textual requirements specification that is obtained by means of the transformation in Figure 3 can be considered to be a very simple specification that is little useful throughout the rest of development process. This is true. However, this simplicity has been explicitly chosen in order to facilitate customers to understand them. More formal requirements specifications which can be taken as reference point throughout the development process are obtained in the next section.

4 Obtaining Task-Based Requirements Models

In this section, we introduce a model-to-model transformation that allows analysts to transform the Web application description based on concepts of a requirements ontology into a task-based requirements model. This model is presented in [3].

The transformation has been defined by using a graph transformation technique [4]. Graph transformations are graph rewriting rules made of basically a Left Hand Side (LHS) and a Right Hand Side (RHS). They are applied in the following way: when the LHS matches into a host graph G (which in this case represents the source model) then the LHS is replaced by the RHS.

Figure 4 shows two representative examples of transformation rules. Rule 1 transforms the main concept of a type ontology (which indicates the main purpose of a Web application type, see Section 2.1) into the root of a hierarchical task description. Rule 2 match with the concept "Checkout Identification" which indicate the type of identification that the E-commerce application must support (see Figure 1). According to this concept users must identify themselves when checkout. Then, this concept is derived into a task-based representation which indicates that users must first login and then handle payment in order to checkout.

We have chosen a graph transformation technique because several widely validated tools can be found. In particular, we have chosen the AGG (Attributed Graph Grammar System) tool [5]. The AGG tool can be considered to be a genuine programming environment based on graph transformations. It provides 1) a programming language enabling the specification of graph grammars and 2) a customizable interpreter enabling graph transformations. AGG was chosen because it allows the graphical expression of directed, typed and attributed graphs (for expressing specifications and rules). It has a powerful library containing notably algorithms for graph transformation, critical pair analysis, consistency checking and application of positive and negative conditions.

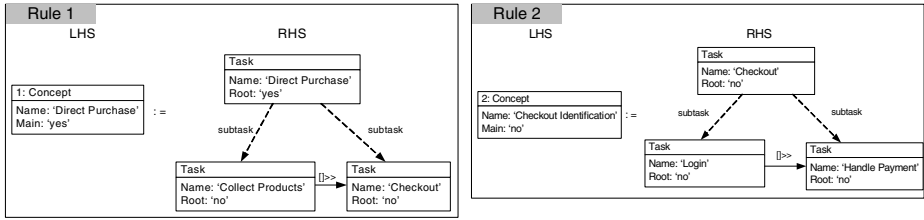


Fig. 4. Examples of transformation rules

5 Conclusions

In this paper we have presented a RE environment in order to improve the communication activity during the RE process.

We have introduced a technique based on Requirements Ontologies in order to facilitate customer to describe their needs. This technique is supported by a requirements elicitation tool which provides customers with an intuitive interface which allows them to generate a structured description of their needs. This helps analysts to understand which Web application customers need. Furthermore, two transformations have been presented: (1) A model-to-text transformation which transforms ontology-based descriptions of the customers' needs into textual requirements specifications. This facilitates customers to validate the requirements specification. (2) A model-to-model transformation which transforms ontology-based descriptions of the customers' needs into task-based requirements models. This provides precise requirement specifications to facilitate programmers interpret them.

References

1. AL-Muhammed, M., Embley, D.W., Liddle, S.: Conceptual Model Based Semantic Web Services. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, Springer, Heidelberg (2005)
2. Embley, D.W.: Programming with Data Frames for every Items. In: Proceedings of AFIPS Conference, Anaheim, California, pp. 301–305 (1980)
3. Valderas, P., Fons, J., Pelechano, V.: Developing E-Commerce Application From Task-Based Descriptions. In: Bauknecht, K., Pröll, B., Werthner, H. (eds.) EC-Web 2005. LNCS, vol. 3590, pp. 65–75. Springer, Heidelberg (2005)
4. Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformation. World Scientific, Singapore (1997)
5. The Attributed Graph Grammar System (AGG) v1.5: <http://tfs.cs.tu-berlin.de/agg/>

Meta-model to Support End-User Development of Web Based Business Information Systems

Buddhima De Silva and Athula Ginige

University of Western Sydney, Locked Bag 1797, Penrith South DC, 1719, NSW, Australia
bdesilva@scm.uws.edu.au, a.ginige@uws.edu.au

Abstract. End-user development is proposed as a solution to the issues business people have when getting web applications developed. In this paper, we have presented a meta-model for web based information systems to support End-user Development. End-users can actively participate in web application development using tools to populate and instantiate the meta-model. The meta-model we created is based on three abstraction levels: Shell, Application, and Function. At Shell Level, we model aspects common to all business web applications such as navigation and access control. At Application Level, we model aspects common to specific web applications such as workflows. At Function Level, we model requirements specific to the identified use cases. Inheritance and Overriding properties of the meta-model provide a balance between ease and flexibility when developing business information systems. The key aspect that underpinned this research work is the view- “software is a medium to capture knowledge rather than a product”. Meta-model will help end-users to participate in web application development activities.

1 Introduction

The characteristics of the web such as ubiquity and simplicity make it a suitable platform to disseminate information and automate business processes. AeIMS research group at University of Western Sydney has been working with businesses in Western Sydney region to investigate how Information and Communication Technologies (ICT) can be used to enhance their business processes [1-3]. In this work, we have identified many issues Business users have to overcome when trying to implement web applications [3]. These issues vary from not being able to get web applications developed to meet needs of the business in a timely manner to development projects running over budget. The development approach should also reduce the gap between what the users actually wanted and what is being implemented in terms of functionality [4]. Researchers have proposed to empower end-users in web application development as a solution to these issues [3, 5-8].

There are different approaches to empower end-users. One approach is to provide end-users with tools to develop any kind of a web application such as informational, search directory and directory look up, workflow and collaboration, e-commerce and web portal, etc. However, such a tool will become very complex because it has to cover a variety of features [3]. The other approach is to develop different tools to

support different types of web applications. Our approach falls in to the second category. We have analysed many business web applications. From this analysis, we identified, the business users need support to store, process and report information to effectively carry out various business processes. The meta-model for this class of business web applications at conceptual level is form being routed based on rules. Example instance of that meta-model is a leave processing system where employees can apply for leave. Meta-model elements are high level abstract concepts such as user, role, form User Interface, business object etc. Being able to develop web applications using these high level concepts help to address the critical concerns of end-user developer relating to details of creating databases, creating web forms, user authentication, etc..

The major finding presented in this paper is the hierarchical meta-model. In section 2 we present the hierarchical meta-model and section 3 discuss the properties of meta-model. In section 4 we review the related work and section 5 conclude the paper.

2 Hierarchical Meta-model of Business Web Applications

As mentioned previously we view Web based business applications as an instance of a meta-model. Theoretically by creating a meta-model and developing tools to populate the instance values we can generate business applications. In practice creating a meta-model to support end-user development is not easy because of the complexities of business applications. To manage the complexity we developed the meta-model at 3 levels of hierarchical abstraction called Shell, Application and Function as shown in figure 1.

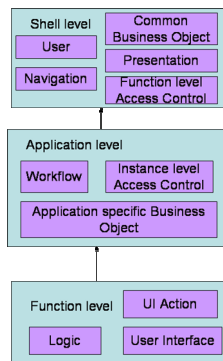


Fig. 1. Hierarchy of Abstraction Levels of the Meta-Model

- Shell Level: The aspects common to all web applications such as user, navigation are modeled at shell level.
- Application Level: The aspects specific to a web application such as workflow, instance level access control are modeled at this level.
- Function level: The function specific aspects which are required to implement the function are modeled at function level.

2.1 Shell Level

We analysed many business applications to identify common functionality required in most of the web applications. These common functionalities are modeled at the shell level. User model, Access Control model, Navigation Model and Business Object Model are four models that we identified at this level. Shell level of meta-model is shown in figure 2.

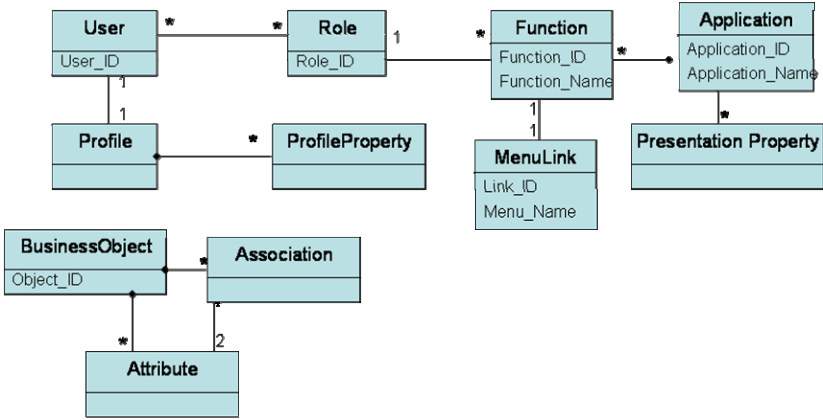


Fig. 2. Shell level of meta-model

Each user has a profile. Profile can have many properties such as name, address, e-mail address, etc. User can play many roles. A role can have one or more users. Each role has functions in an application assigned to it. Application has presentation properties. Each function has a menu link. Business Objects have many attributes and associations. Navigation model is the mechanism that authenticated users can use to access the authorised functions. We have identified two types of functions: State independent functions and State dependent functions. Once a user log in, user will be provided with a menu to access state independent functions which are available to authorized users to access based on the navigation model. On the other hand, the state dependant functions are available to users only if it is waiting for that particular user at that time. We have a menu link to access such functions in all applications at one place.

2.2 Application Level

As mentioned earlier an application consists of many functions. Therefore, application consists of models which support many functions. The Application level of the meta-model is shown in figure 3. Application inherits the function level access control, common business objects and navigation models from the shell level. It consists of workflow model, instance level access control model and application specific object models. Workflow model models the business rules that govern the flow of information. When the business objects are accessed through the state dependant

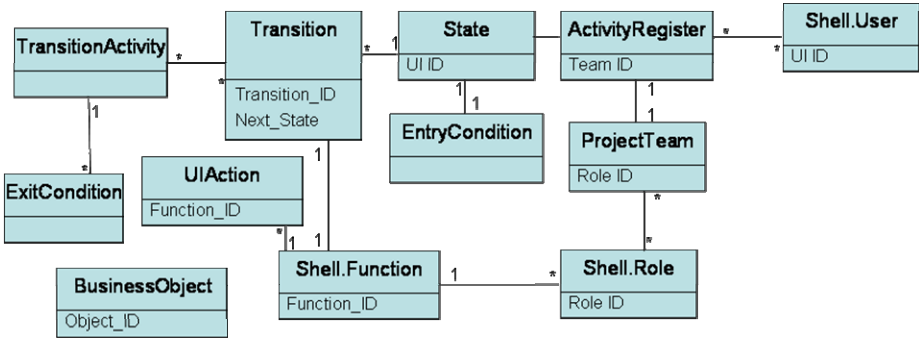


Fig. 3. Application Level of meta-model

functions there could be rules specifying who can access what instances of a Business object. By applying the instance level access rules we can identify the ‘project team’ that participates in actions in state dependant functions in the workflow.

2.3 Function Level

Functions are the way of performing the actions in an application. The User Interface is the mechanism users have to perform the functions.

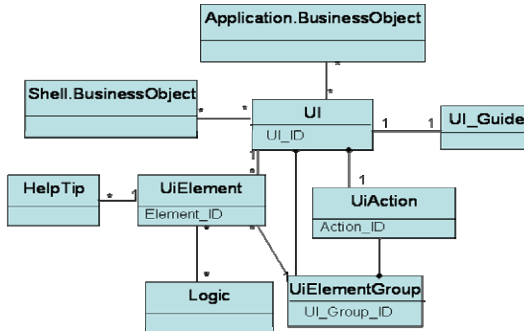


Fig. 4. Function level of meta-model

The function level meta-model is shown in figure 4. User Interface Model consists of UI guide, UIElementGroup, and UIElements, and UI Actions. UI elements can be in input mode or output mode. In a form we have UI elements in Input mode. In a report we have UI elements in output mode. UIGuide provide the guidelines to use the particular interface. At function level we model two types of business rules. One type is the business rules used to derive new object attribute values based on existing object attribute values. The other category is the validation rules applied over values of form field in a user interface.

3 Meta Model Properties

The hierarchical model has two properties inheritance and overriding. These two properties provide a balance between flexibility and ease of use. For example, if end-user wants he/she can develop a web application with default presentation style inherited from the shell. He/she doesn't have to bother about the presentation styles and templates at the application level. However, if an end-user wants a custom look and feel he/she can override the default presentation style provided from shell level at the application level.

4 Related Work

Several meta-models exist for web applications. One of them is UWE meta model [9, 10]. It is designed as an extension based on MOF 1.4. The objective of UWE meta-model is to provide a common meta-model for the web application domain, which will support all web design methodologies. UWE meta-model has similar model elements as ours. However, the main difference between the 2 meta-models is the 3 level, hierarchical abstraction that we used to support end user development. W2000 [11], a successor of HDM [12], has provided model semantics and transformation rules to achieve consistency between models. Muller et al. [13] present a model-driven design and development approach with the Netsilon tool. The tool is based on a meta-model specified with MOF 1.4 and the Xion action language. Recently another two meta models[14, 15] based on MOF and UML 2 profiles are presented for WebML design methodology with the objective of interoperability.

All these meta-models of web applications are towards the precise definition of the semantics of existing web models. Our work is complementary to existing web meta-models; in that we propose a hierarchical organization of meta-model with a different perspective- to help to effectively involve end users in development.

5 Conclusion

In this paper we have presented a hierarchical meta-model enabling business end-user to develop business web applications. The semantics in our meta-model helps end users to begin the development work with little knowledge in computer domain. For example to configure an application they only need the knowledge of the shell meta-model. To participate in development at application level they have to use their domain knowledge and logical thinking; thus should know the application meta-model. We limit our focus to form based web applications as these are the most required by business end users. However, in the future we are planning to expand the meta-model for development of other types of web applications. We are currently planning formal experiment in end user development based on this meta-model. This will help us to better understand the mental model of end users; thus help to refine the tools available for end users to instantiate the meta-model.

Reference

1. Arunatileka, S., Ginige, A.: Applying Seven E's in eTransformation to Manufacturing Sector. in *eChallenges* (2004)
2. Ginige, A.: From eTransformation to eCollaboration: Issues and Solutions. In: *2nd International Conference on Information Management and Business (IMB 2006)* Sydney, Australia (2006)
3. Ginige, J., De Silva, B., Ginige, A.: Towards End User Development of Web Applications for SMEs Using a Component Based Approach. In: *Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, Springer, Heidelberg* (2005)
4. Epner, M.: Poor Project Management Number-One Problem of Outsourced E-Projects, in *Research Briefs, Cutter Consortium* (2000)
5. Ginige, A., De Silva, B.: CBEADS: A framework to support Meta-Design Paradigm. In: *3rd International Conference on Universal Access in Human-Computer Interaction (UAHCI07), China. LNCS, vol. 4554, pp. 107–116, Springer, Heidelberg* (2007)
6. Costabile, M., F., et al.: A meta-design approach to End-User Development. In: *VL/HCC05* (2005)
7. Fischer, G., Giaccardi, E.: A framework for the future of end user development, in *End User Development*. In: *Lieberman, H., Paterno, F., Wulf, V. (eds.) Empowering People to flexibly Employ Advanced Information and Communication Technology, Kluwer Academic Publishers, Boston, MA* (2005)
8. Fischer, G., et al.: Meta Design: A Manifesto for End -User Development. *Communications of the ACM* 47(9), 33–37 (2004)
9. Kraus, A., Koch, N.: *A Metamodel for UWE. 2003, Ludwig-Maximilians-Universität München* (2003)
10. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. In: *Third international conference on Web engineering, Oviedo, Spain. Springer-Verlag, Heidelberg* (2003)
11. Luciano, B., Sebastiano, C., Luca, M.: First experiences on constraining consistency and adaptivity of W2000 models. In: *Proceedings of the 2005 ACM symposium on Applied computing. ACM Press, Santa Fe, New Mexico* (2005)
12. Garzotto, F., Paolini, P., Schwabe, D.: HDM — A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems (TOIS)* 11(1), 1–26 (1993)
13. Muller, P., et al.: Platform independent Web application modeling and development with Netsilon. *Software & System Modeling* 4(4), 424–442 (2005)
14. Schauerhuber, A., Wimmer, M., Kapsammer, E.: Bridging existing Web modeling languages to model-driven engineering: a metamodel for WebML. In: *Workshop proceedings of the sixth international conference on Web engineering. ACM Press, Palo Alto, California* (2006)
15. Nathalie, M., Piero, F., Antonio, V.: A UML 2.0 profile for WebML modeling. In: *Workshop proceedings of the sixth international conference on Web engineering. ACM Press, Palo Alto, California* (2006)

Easing Web Guidelines Specification

Barbara Leporini, Fabio Paternò, and Antonio Scordia

ISTI - CNR

Via G. Moruzzi, 1 - 56124, Pisa

{barbara.leporini, fabio.paterno, antonio.scordia}@isti.cnr.it

Abstract. More and more accessibility and usability guidelines are being proposed, especially for Web applications. Developers and designers are experiencing an increasing need for tools able to provide them with flexible support in selecting, editing, handling and checking guidelines. In this paper, we present an environment for addressing such issues. In particular, an interactive editor has been designed to assist designers and evaluators in abstracting and specifying new and existing guidelines in an XML-based Guideline Abstraction Language (GAL). Our tool has been designed in such a way to be able to check any guidelines specified in this language without requiring further changes in the tool implementation.

Keywords: Guidelines, editor, abstraction language, accessibility, usability.

1 Introduction

Design guidelines are increasingly applied in order to improve Web user interfaces and make them consistent, in particular to attain better accessibility and usability. Indeed, several guidelines have been proposed in the literature for general user interfaces (e.g., [10]) as well as for Web interfaces (e.g. [5, 9, 12]). Thus, in order to assure a certain level of accessibility and usability, developers have to orient themselves among several sets of guidelines.

In order to support designers in checking guidelines, a number of automatic tools have been made available on the market, such as Bobby, now called WebXACT [14], Web ATRC [13], Lift [8], and so on. However, such tools have several limitations. For example, they only support the assessment of a fixed set of guidelines. In order to overcome such limitations, we have investigated new solutions for more flexible support [7]. To this end, we have designed and implemented an environment, MARGENTA (Multi-Analysis of Guidelines by an ENhanced Tool for Accessibility) which is a tool that supports the inspection-based evaluation of accessibility and usability guidelines. One of its main features is that it is guideline-independent. In fact, our tool has been developed considering the limitations of most current tools, in which the guidelines supported are specified in the tool implementation, with a certain number of consequent drawbacks, such as having to change the tool implementation each time a guideline is modified or added. Thus, the tool has been developed in such a way to be able to deal with any type of guidelines, which are defined externally from the tool. To this end, an abstract language for guidelines has been defined. If the

guidelines are expressed through it and stored in external files then the tool can check any of them without modifications to its implementation.

However, we soon realised that many Web designers can have difficulties in handling and formulating guidelines, usually specified through XML-based languages. For these reasons, we have extended the environment with a new tool conceived to assist developers and evaluators in proposing, specifying and modifying guidelines.

In this paper we present our solution to this issue, which includes a Guideline Editor designed to assist developers in specifying new guidelines and handling already existing ones (such as those for accessibility and usability for the Web). We also report on the results gathered through a user test conducted in order to evaluate the editor in terms of user interface as well as functionalities. Based on the collected data during the evaluation, we decided to further improve the guideline editor tool in order to ease its use and the guideline specification process as well.

After a short overview of related work on the topic addressed, in this paper we firstly describe our Guideline Abstraction Language (GAL) used for specifying guidelines. Next, we introduce the main functionalities of the proposed editor for Web Guidelines. We also report on the data and opinions collected through a user test. Then, changes made on the editor to enhance its functionalities are described. Lastly, we draw some conclusions and provide indications for future work.

2 Related Work

National and international legislations more and more promote inclusive policies and practices. With regard to accessibility, the number of proposed guidelines is rapidly increasing, thus requiring an increasing effort by developers who have to handle them. Indeed, in addition to the W3C guidelines almost each Western country has national guidelines for accessibility, and new guidelines focusing on specific types of users have started to appear. To this end, several automatic or semi-automatic supports have been proposed for easily handling guidelines. In particular, several automatic tools have been developed to assist evaluators by automatically detecting and reporting guideline violations and in some cases making suggestions for fixing them. EvalIris [1] is an example of a tool that provides designers and evaluators with good support to easily incorporate new additional accessibility guidelines. Another contribution in this area is DESTINE [2], which supports W3C and 508 guidelines specified through another language. Also, in the BenToWeb project two tools are proposed to support test case management for accessibility test suites [4]: Parsifal - a desktop application, which easily allows editing test description files – and Amfortas - a Web application, which allows controlled evaluation of the test suites by users. A different approach is presented in [6], which supports the creation of an internal design knowledge management tool for Web developers as a means to encourage user-centred development practices.

In order to verify various usability aspects related to accessibility issues, other tools have been proposed. The Functional Accessibility Evaluation (FAE) Tool [3] provides a means to estimate the functional accessibility of Web resources by analyzing Web pages and estimating their use. FAE uses rules for testing each of the functional accessibility features of navigation, text descriptions, styling, scripting and the

use of standards. Such rules are defined on X/HTML tags. This means that they are directly implemented in the source tool code. Consequently, to add or modify a rule the source code must be modified.

Also the approach reported in [11] is based on the separation between the guideline specification and the evaluation engine. Guidelines are stored into a XML-based repository. A guideline can be expressed in terms of conditions to be satisfied on HTML elements (i.e., tags, attributes). The formal guideline is expressed according to the Guideline Definition Language (GDL), a XML-compliant language. Although the GDL allows mapping the informal statements associated with initial guidelines onto formal statements, such a process has to be manually carried out by the developers who have to directly handle the XML constructs specifying the guideline structure. To this end, we propose an editor that supports designers in the guideline specification, even if they do not have any knowledge of XML and its constructs.

In summary, our contribution in this area is represented by an environment, which is able to work with different sets of guidelines, provided they are appropriately specified in external files. This means that developers or evaluators should formalize and specify the guidelines in a specific format based on XML. Since this may require particular abilities as well as a long time, we decided to develop a tool supporting guidelines editing and modifications, which is presented in this paper, since there is a lack of similar tools as indicated in the above discussion.

3 The Guideline Abstraction Language

The tool we propose aims to provide novel support for Web designers, such as providing interactive support for defining new guidelines as well as customizing new groups starting with other existing guideline sets. Since our intent is to provide designers with a tool independent of the guidelines to check, the proposed approach is based on a language able to abstract the usual types of conditions that designers aim to assess. The tool includes an engine able to interpret and check any guideline specified by GAL without requiring any further modifications in its implementation. Currently, in the MAGENTA tool three sets of guidelines can be specified and interpreted at run-time: (1) the Italian law for accessibility requirements; (2) W3C WCAG 1.0 guidelines; (3) accessibility and usability guidelines for the visual impaired.

3.1 Guideline Abstraction

In general, a guideline is defined as a rule or principle that provides guidance to appropriate behaviour. Generally speaking, a guideline is expressed in natural language, which automatic tools cannot handle. Indeed, a guideline should be specified in a rigorous manner so that it can be automatically processed. and managed. In practice, the solution consists of abstracting guideline statements into a well-defined XML-based language. To this end, an XML-Schema has been defined in order to express the general structure of our Guideline Abstraction Language (GAL).

In defining a guideline abstraction, the main features and elements characterizing the guideline itself must be defined. In brief, each guideline expresses a principle to be complied with by applying one or more conditions to checkpoints, which are

constructs in the implementation language. Hence, general features, objects involved in the checking process, and conditions referred to such objects must be structured and specified through our GAL specification language. In particular, we define a guideline in terms of a number of elements:

- checkpoints expressed with objects,
- operations (Exist, Count, Check, Execute), which identify the general processing required to check a guideline,
- conditions to be verified.

Such Abstraction Language has been refined over time in order to better define more complex guidelines. We added new attributes in order to be able to specify more precise conditions. It is possible to select specific objects (tags) having given attributes with specified values (e.g. links with the class="navbar"). Such a selection allows designers to include in the guideline specification also cases in which only objects (tags) with given features (attributes) should be considered. Moreover, the language supports the selection of those objects (tags) being children of other tags in the DOM (Document Object Model) structure. For example, in this context the tags <head> and <body> are children of <html> tag (root). This allows designers to specify specific restrictions associated with the tags considered. For instance, to check graphical links - - in the XML-based guideline we can specify the tag as child of the tag <a>. In this way, the checkpoint only focuses on the tags included between the tags <a> and . This feature has been added to be more precise and able to restrict the selection of objects to involve within the checkpoints.

3.2 Abstraction Examples

In order to understand how the abstract description of a guideline can be obtained through GAL, in this paragraph we provide the reader with some examples.

Example 1: Language identification

When using different languages on a page, any language change should be clearly identified by using the "lang" attribute. In this context, let us consider the guideline "Use mark-up elements to facilitate document language identification. In particular, in multi-language pages designers should use the appropriate attributes to mark links written in a different language.". This means we need to check the presence of the attribute "lang" in the tag <html> as well as the link tags <a> that do not refer to anchors (i.e. tags with the attribute href).

To abstract this guideline, the objects involved in the guideline must be extracted. Firstly, there are two aspects to consider: (1) the object to examine is the tag <html>; (2) the elements to analyse are all links. These two controls indicate that two checkpoints should be specified. For each checkpoint, in addition to the main object involved - i.e. <html> tag in the first case, and <a> in the second one - the corresponding conditions must be defined. The main structure could be abstracted as indicated below:

GDL: [HTML^{tag} (Exist LANG^{attrib})]¹ and [A^{tag} | HREF^{attrib} (Exist LANG^{attrib})]²

In order to satisfy the guidelines, the checkpoints must be applied. A checkpoint defines a series of conditions to verify or to apply in order to satisfy an aspect of a guideline. In our case, each checkpoint is included in the square brackets marked 1 and 2; round brackets are conditions to be verified.

Next figure shows how the guideline example can be specified by using the XML-based GAL language. This guideline example was also used for a task (T5) assigned in a user test conducted to evaluate the editor (see section "Editor evaluation").

Table 1. Example of guideline abstraction with GAL

```
<guideline id="22" summary="Language identification">
<checkpoints rel="and">
<cp id="22.1" summary="Main document language" priority="1">...
<object type="tag">html</object>...
<conditions rel="or">
<evaluate operator="exist" idErr="22.1.1">
<el type="attrib">lang</el></evaluate>
</conditions></cp>
<cp id="22.2" summary="Marking links with different language" priority="3">
<object type="tag">a</object>
<select type="attrib">href</select></eval_object>
<conditions rel="and">
<evaluate operator="exist" idErr="22.2.1">
<el type="attrib">lang</el></evaluate>
</conditions></cp></checkpoints></guideline>
```

Example 2: Equivalent alternatives to auditory and visual content

Let us consider the guideline n. 1 of WCAG 1.0, which is related to alternative contents. This guideline is composed of several checkpoints. For our example we consider the checkpoint 1.1: "Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). This includes: images, graphical representations of text (including symbols), image map regions, animations, applets and programmatic objects, images used as list bullets, graphical buttons, sounds, and so forth.". To simplify our example, we only report the specification of the "image" case. The next figure shows the XML-based specification of the simplified checkpoint 1.1 according to the GAL schema.

Actually the checkpoint should be extended with other conditions (<conditions>) to include all the non-text elements, such as objects, applets, and so on.

The checkpoint involves the tag to which five evaluations are applied: (1) the presence of attribute "alt"; (2) the "non-empty" verification for that attribute; (3) the "not belonging" of alt content to a black list of possible values which are commonly used for alternative descriptions (e.g., "logo" or "file-name.jpg"); (4) the control of the presence of "longdesc" attribute and (5) that its value is not empty. These evaluations are mainly specified with the operators "exist" as well as "check" with the conditions "not_equal" and "not_belong". The black list of values are stored in external files, such as "noImages.dic".

Table 2. Example of abstraction according to GAL

```

<guideline id="1" summary="Alternatives to auditory and visual content">
<checkpoints rel="and">
<cp id="1.1" summary="Provide a text equivalent for images" priority="1">
  <object type="tag">img</object>
  <conditions rel="or">
    <evaluate operator="exist" idErr="1.1.1">
      <el type="attrib">alt</el></evaluate>
    <evaluate operator="check" cond="not_equal" idErr="1.1.2">
      <el type="attrib">alt</el>
      <el type="value">""</el></evaluate>
    <evaluate operator="check" cond="not_belong" idErr="1.1.3">
      <el type="attrib">alt</el>
      <el
type="file">"noimages.dic"</el></evaluate>
  <!-- repeated code for longdesc ->
  </conditions></cp></checkpoints></guideline>

```

The kind of error is coded with the attribute `idErr` defined in GAL. A progressive number is associated to each evaluation statement (`<evaluate>`); it is obtained by adding a progressive number to the checkpoint id (e.g., 1.1.2 where 1.1 is the checkpoint id). The list of possible errors is externally coded by matching the type - (e) error, (w) warning and (a) alert - with corresponding message for each `idErr`.

Example 3: Logical partition of interface elements

As third example let us consider the guideline that requires well structured content in a page in order to simplify Web navigation through screen readers: " All interface

Table 3. Fragments of guideline specification according to GAL

CP	Fragment
1. Headings	<pre> ... <conditions rel="or"> <evaluate operator="count" cond="greater" idErr="1.1.1"> <el type="tag"> h1 </el> <el type="value"> 0 </el></evaluate>... </pre>
2. div blocks	<pre> <conditions rel="or"> <evaluate operator="count" cond="greater" idErr="1.2.1"> <el type="tag"> div </el> <el type="value"> </el></evaluate></conditions></cp> </pre> <p style="text-align: right;">1</p>
3. Paragraphs	<pre> <conditions rel="and"> <evaluate operator="count" cond="greater" idErr="1.3.1"> <el type="tag"> p </el> <el type="value"> </el></evaluate></conditions></cp> </pre> <p style="text-align: right;">0</p>

elements should be well-arranged and structured. Use heading levels, div blocks and paragraphs”. To specify this guideline three checkpoints must be defined: (1) for heading levels; (2) for div blocks; and (3) for paragraphs. The next figure reports fragments of the guideline specification based on GAL.

In the checkpoint 1, heading levels are counted. In practice, the control consists in verifying that the number of heading levels is at least 1. Thus, in this case the effective verification of an appropriate page structure is demanded to the evaluator; the checkpoint points out the usage or not of headings in the page. The evaluator will manually check if the structure has been appropriately applied. A similar checking is applied to the checkpoints 2 and 3. Checking is not easy to perform in completely automatic way. The main goal of this guideline and its evaluation is to stimulate attention on the page structure by the developers.

4 The Guideline Editor

When a guideline must be defined so that a tool can deal with it automatically, the problem is to transform its natural language description into technical terms. The abstraction process is not easy and requires some effort when implementation languages for the Web are considered. In order to facilitate this process, a graphical editor was designed and added to our environment.

4.1 Functionalities

The Guideline Editor (GE) has been designed to assist developers in handling single as well as groups of guidelines. The tool supports new guideline definition and various types of editing. In summary, the editor offers various useful features in order to facilitate multiple guideline sets management and general utilities. In particular, the guideline editor provides support for:

- Defining or modifying single guidelines;
- Importing guidelines from various sets in order to create customised groups;
- Organising, classifying and browsing guidelines into groups;
- Handling several sets of guidelines simultaneously.

Regarding manipulation of single guidelines, the tool allows developers to add new ones or modify/delete existing ones. In practice, the graphical editor guides the designer in specifying all the XML tags necessary to define a guideline. Figure 1 shows an example in which the designer specifies the guideline “The number of links in a navigational bar should be less than 7”. The left part of the graphical representation is devoted to defining the scope of the application, in this case the tag `<a>` with class attribute value equal to `navbar` in HTML documents. The right part indicates the condition to check: in this case that the number of occurrences of the indicate tag should be less than 7. Furthermore, it is also possible to add new guidelines by importing them from other guideline set(s). This allows designers to create customised set(s) of guidelines by reusing already specified ones. More precisely, regarding individual guidelines, the tool makes it possible to:

- Handle general information on the guideline (short name, description, etc.);
- Select the object to be considered in the guideline (i.e., language, tags and restriction on the object itself).
- Handle the checkpoints needed: general information for each checkpoint (e.g., short name, description) and checkpoint relations (and | or);
- Define the conditions to be verified, by selecting operators and conditions to be applied at a runtime (i.e. when the tool will carry out the evaluation).

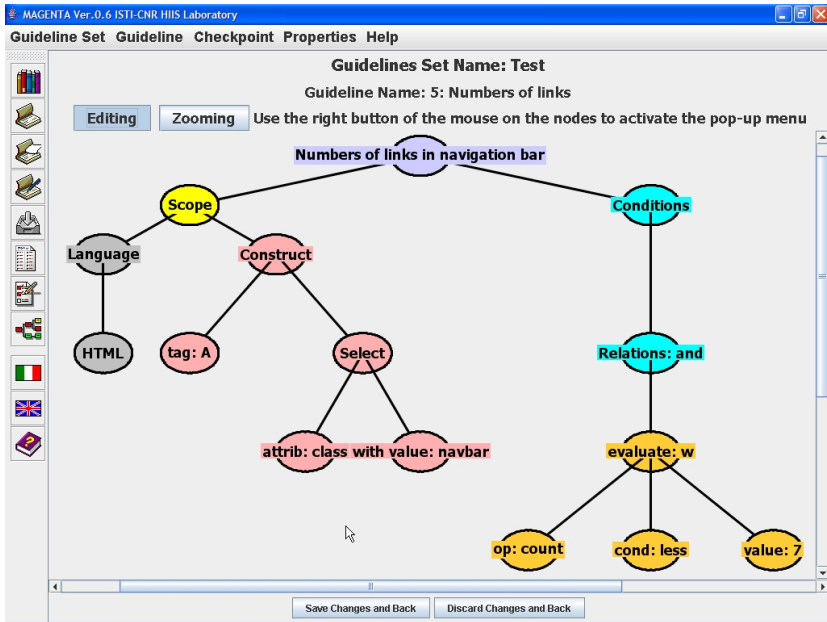


Fig. 1. The Guideline Editor

4.2 User Interface

The Guideline Editor UI is organized in various parts shown in cascade:

- *Main Guideline Management*, where all groups of guidelines are listed and can be handled. It is possible to create a new set, modify or delete an existing one.
- *Guideline set*, in this environment a new specific group of guidelines can be defined. General information on the set, as well as the list of guidelines existing in the group are presented.
- *Guideline specification*, where developers can structure and define a single guideline. Firstly, general data such as the short name and description of the current guideline as well as the main object – tag, attribute or property - involved in the guideline can be written and modified; next, all checkpoints and their conditions to be applied or evaluated can be edited through a direct manipulation graphical editor.

In guideline editing, the elements to be selected are listed in order to make them available for the evaluator and to avoid typing errors. In this way, the user is guided during the abstraction and definition process. For instance, when an object to be checked has been selected, the list-box for selecting the associated attributes displays only its attributes according to the implementation language considered. When the user has selected all the elements that express the objects to be checked, including their properties, checkpoints and conditions, the editor saves them in the XML-based file associated to the current guideline set.

While in the previous version many windows opened during the process of defining guidelines and checkpoints, an updated version utilised a single window to display such information, with the additional possibility to zoom in the parts of interest. In addition, a new interactive representation of the checkpoint structure was introduced in order to simplify the somewhat “complex” structure of a checkpoint via a hierarchical tree structure representation.

5 Editor Evaluation

5.1 Method

To get information about the actual use of the guideline editor prototype, we used two evaluation techniques: think aloud and questionnaires. We observed users interacting with the guideline editor. Users were asked to complete a set of predetermined tasks, while we watched and recorded the users' actions (using the paper-and-pencil analysis protocol). At the end of the test we asked users to fill in a questionnaire.

5.2 Participants

In order to evaluate the guideline editor prototype, ten designers were recruited, 7 men and 3 women. Since the guideline editor is conceived for an expert evaluator or developer, the choice in recruiting potential participants was limited. The user age varied from 22 to 38 years. All users were computer scientists: 40% were university graduates, whereas the other 60% were still students. On average, users had a poor knowledge about guidelines (2.1/5), and a sufficient knowledge of the X/HTML language (3.1/5).

5.3 Tasks

The assigned tasks focused on the main functionalities offered by the editor, such as handling guideline groups or single guidelines. The five tasks assigned to the users were:

T1 Creating a new guideline group named “test”.

The purpose was to create a new group associated to an XML file in which all guidelines elaborated by the user were stored. This file was considered the final result of the test carried out by each user.

T2 Importing various guidelines from existing groups.

The purpose of this task was working with several guidelines to populate the “test” group.

T3 Editing an existing guideline.

This task was mainly assigned to allow users to familiarize themselves with the guideline structure.

T4 Adding a new “guided” guideline.

This task was guided by providing the user with all the necessary information to describe the guideline formalization. The purpose was to observe users when they edited the elements without having to go through the guideline formalization process.

T5 Adding a new “unguided” guideline.

The purpose was to observe the user when formalizing a guideline expressed in a natural language.

5.4 Questionnaires

Since the purpose of our test was to collect information on the Guideline Editor (GE) interface and its functionalities, the questionnaire was basically composed of three kinds of questions: (1) general questions aimed at gathering information on users; (2) interface section, aimed at getting useful comments and impressions about the editor UIs; and (3) functionality section, to collect remarks on the main GE functionalities tested. To collect useful information and comments by the users, we included both open questions and “quantitative” questions (on a 1 to 5 scale).

5.5 Results

The observational data gathered through the think aloud test and subjective judgments collected by the questionnaires provided some useful empirical feedback. Additionally, all XML files created in T1 allowed us to analyse the task accomplishment for each user. Task T1 and T2 were carried out by all users, even if we observed that T2 presented some difficulties when extracting the desired guidelines from the groups. In fact, some users reported that the rendering structure used for grouping guidelines in the import procedure was not particularly clear. Larger labels or a more guided procedure could be more useful for orienting users among many guidelines. Regarding the task T3 (guideline editing) – “verifying that the summary attribute of a table is not empty” - only 50% of users understood that the needed action was to add a new <evaluate> condition (i.e. a new branch in the checkpoint tree). Lastly, regarding the guideline creation requested in task T4 and T5, about half users encountered some problems in carrying out the tasks correctly. We noted some difficulties in formalizing a guideline due to some confusion encountered by the users in the logical flow of the needed actions. Concerning the last task (T5) – “abstracting and formalizing a guideline expressed in terms of natural language” - users worked better with the checkpoint structure than in the other tasks. This may be due to the greater familiarity with the tool acquired during the test (the sessions lasted on average 45 minutes per user).

In order to collect subjective opinions regarding the user interface and editor functionalities, users were asked to express some judgments on several aspects. The scale varied from 1 (lowest – negative value) to 5 (highest – positive value). Regarding the general user interface, the average rating was 3.2. Users expressed intermediate values - on average - also for the more specific UI aspects, such as location (3.6) and clearness (3.2) of elements, icon clearness (3.5) and graphical structures (3.4). The lowest average value reported (2.7) regards the logical flow in structuring a guideline. In the questionnaire, users expressed also some comments about difficulties encountered during the test.

Concerning the editor functionalities, users were asked to express opinions of the main functions tested. The averages are higher (around 4) than those expressed for the interface (around 3). The function considered most difficult by users is related to checkpoint handling. Indeed, handling a checkpoint was the most frequent error in performing the tasks T3 and T4. Task T1 and T2 were those considered the easiest. By analysing open questions, we observed that in general users appreciated the guideline editor. Most difficulties were related to the guideline formalization. This is probably due to the required advanced knowledge of X/HTML in terms of tags and attributes. To address this issue, we could improve the interaction by inserting a more descriptive and clearer list of elements. For example, instead of listing all tags `<a>`, `<frame>` and so on, we could use a list of more explicative items, such as link, frame, etc.. The same can be applied for the attributes. All in all, the editor had a positive impact on the users. However, some improvements were still needed in order to make it more usable and more intuitive.

6 The New Version of the Tool

Based on the gathered opinions and the observed user actions, various difficulties related to the user interface emerged. For this reason we decided to make some changes on the GE tool. Changes basically affected two kinds of aspects: (1) interface functionalities and (2) specification and abstraction process.

With regard to interface functionalities, the major revisions introduced are related to interface components (such as labels, buttons, graphical views, etc.). Concerning the labels, extended names rather than technical tags have been introduced to identify the elements involved in the abstraction process. Indeed, users encountered some difficulties in orientating among the X/HTML tags. For instance, clearer labels (such as "Link", "Image", "Table", and so on) have been used to replace tags such as `<a>`, ``, `<table>`, etc.. Although commands and functions are available on the tool bar as well as on the context menu (i.e. associated to the mouse right click), users showed some confusion in locating the necessary command. For this reason, various command buttons have been repeated in the current work area. In addition, in order to provide a visual structure of a guideline, a graphical tree was used. However, such a tree had various accessibility problems (for example, blind users interacting through screen readers are not able to use the mouse as required by editing such graphical representation), thus an alternative representation has been developed. The basis idea was to provide a more accessible technique consisting in reproducing the visual tree

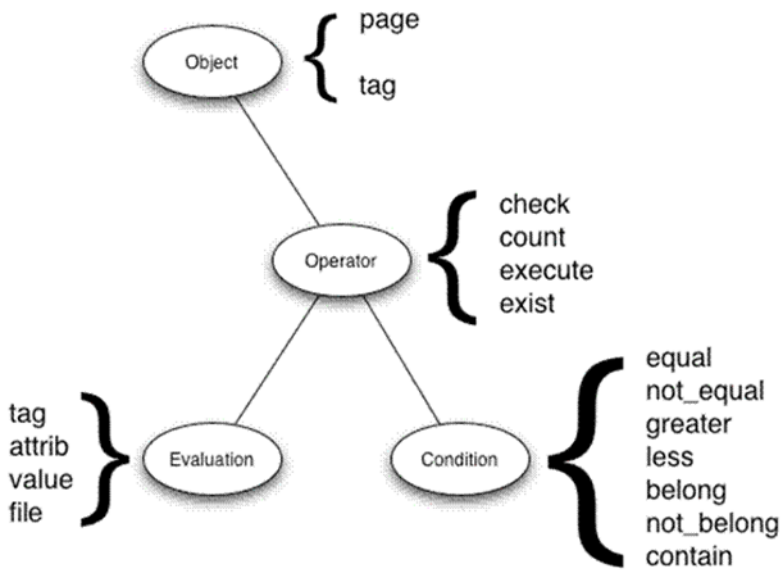


Fig. 2. The Structure of a Guideline in GAL

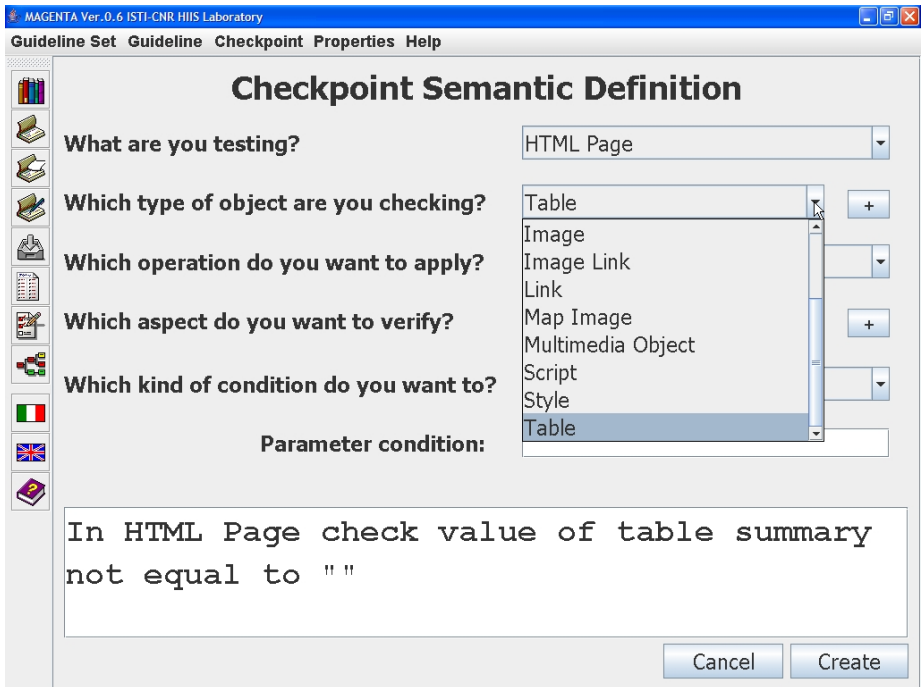


Fig. 3. The Editor of the Guideline Textual Description

in a way similar to the typical “resource explorer”. This type of technique is probably less effective from a graphical viewpoint, but it is accessible even by people with disabilities using assistive technology, such as screen readers.

The second kind of introduced changes was more substantial and was related to the specification and abstraction process. The basic idea was to guide the user through the guideline abstraction process. For this purpose, the structure of the GAL guideline specification was analysed. As Figure 2 shows, there are four main components: the object of the evaluation, the operator indicating the type of processing, the specific aspect to evaluate, and the condition to check.

Thus, we decided to provide support able to guide the designers through dialogue windows. The dialogue window shows the corresponding four main questions to drive the developer in choosing the involved elements, and the possible answers. The main purpose is to lead the developer through the abstraction process by a logic flow. In this way abstracting should be easier, especially when selecting objects in a correct order.

As shown in Figure 3 the main questions are:

- What are you testing?
- Which type of object are you checking?
- Which operation do you want to apply?
- Which aspect to do you want to verify?

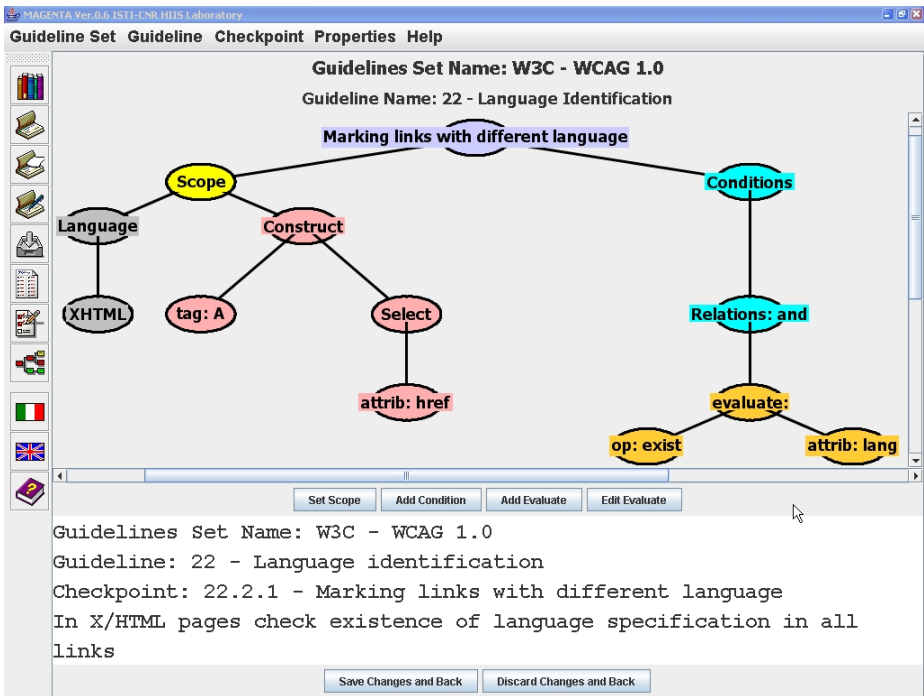


Fig. 4. An Example of Two Representations of the Same Guideline

The possible answers for each question are predefined as well depending on the type of aspect considered. The resulting guideline specified through the answers is also shown in the text area at the bottom.

Consequently, the resulting environment provides two ways to specify the guidelines: the graphical editor and the natural language description obtained through predefined questions. The two possible representations are not independent, thus it is possible to show them at the same time (see an example in Figure 4) and if one representation is modified then the other one is automatically updated to reflect the changes.

7 Conclusions

The increasing number of design guidelines proposed for the Web, in particular for accessibility evaluation, makes the implementation of automatic tools for managing guidelines more and more complex. In order to develop a tool independent of guideline definition, guidelines should be specified separately and interpreted at runtime. To this end, we have introduced a Guideline Abstraction Language to define guidelines using an XML structure. These XML-based files can be managed by our tool, which loads and checks any guideline indicated by the developer through this language without requiring modifications to its implementation. Thus, designers can dynamically select and edit the guidelines to check in their Web site. Since abstracting and defining guidelines requires considerable effort, a Guideline Editor has been developed and integrated in our environment to assist evaluators in these activities. The user interface has been designed to guide developers as much as possible in order to facilitate their tasks. Currently, the editor supports guidelines for documents implemented in the X/HTML and CSS languages.

Future work is planned to support other Web languages such as SMIL and XForms. As we have shown in the paper, the last version of our guideline editor supports two possible representations (graphical and textual), with the associated editing techniques. Future work is also planned to better investigate what types of users prefer each guideline representation.

References

1. Abascal, J., Arrue, M., Fajardo, I., Garay, N., Tomás, J.: Use of Guidelines to automatically verify Web accessibility. In: UAIS, vol. 3(1), pp. 71–79. Springer Verlag, Heidelberg (2004)
2. Beirekdar, A., Keita, M., Noirhomme-Fraiture, M., Randolet, F., Vanderdonckt, J., Mariage, C.: Flexible Reporting for Automated Usability and Accessibility Evaluation of Web Sites. In: Costabile, M.F., Paternó, F. (eds.) INTERACT 2005. LNCS, vol. 3585, pp. 281–294. Springer, Heidelberg (2005)
3. Gunderson, J., Rangin, H.B., Hoyt, N.: Functional Web accessibility techniques and tools from the university of Illinois. In: Proc. of the 8th international ACM SIGACCESS conference on Computers and accessibility (Assets 2006), Portland, Oregon, USA, October 2006, pp. 269–270 (2006)

4. Herramhof, S., Petrie, H., Strobbe, C., Vlachogiannis, E., Weimann, K., Weber, G., Velasco, C.A.: Test Case Management Tools for Accessibility Testing. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A.I. (eds.) ICCHP 2006. LNCS, vol. 4061, pp. 215–222. Springer, Heidelberg (2006)
5. Italian Accessibility Law, Provisions to support the access to information technologies for the disabled (2004)
6. Kuniavsky, M., Raghavan, S.: Common sense and reason: Guidelines are a tool: building a design knowledge management system for programmers. In: Proc. of DUX '05 (2005)
7. Leporini, B., Paternò, F., Scordia, A.: Flexible tool support for accessibility evaluation. *Interacting with Computers* 18(5), 869–890 (2006)
8. Lift. http://www.usablenet.com/products_services/lift_online/lift_online.html
9. Mariage, C., Vanderdonckt, J., Pribeanu, C.: State of the Art of Web Usability Guidelines. In: Proctor, R.W., Vu, K.-P.L. (eds.) *The Handbook of Human Factors in Web Design*, Lawrence Erlbaum Associates, Mahwah (2005)
10. Mayhew, D.J.: *Principles and guidelines in software and user interface design*. Prentice Hall, Englewood Cliffs (1992)
11. Noirhomme-Fraiture, M., Beirekardt, A., Vanderdonckt, J.: Automated Evaluation of Web Usability and Accessibility by Guidelines Review. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 17–30. Springer, Heidelberg (2004)
12. *Web Content Accessibility Guidelines 2.0*, W3C World Wide Web Consortium Recommendation (August-September 2006)
13. Web ATRC, <http://checker.atrc.utoronto.ca/index.html>
14. WebXACT <http://Webxact.watchfire.com/>

A Transformation-Driven Approach to the Verification of Security Policies in Web Designs

Esther Guerra, Daniel Sanz, Paloma Díaz, and Ignacio Aedo

Computer Science Department,
Universidad Carlos III de Madrid (Spain)
{eguerra, dsanz, pdp}@inf.uc3m.es, aedo@ia.uc3m.es

Abstract. In this paper, we present a verification framework for security policies of Web designs. The framework is based on the transformation of the models that conform the system design into a formalism where further analysis can be performed. The transformation is specified as a triple graph transformation system, which in addition creates mappings between the elements in the source and target models. This allows the back-annotation of the analysis results to the original model by means of triple graphical patterns. The verification mechanisms are provided by the designer of the Web design language, together with the language specification. However, the complexities of the formalisms are hidden to the developer who uses the language.

As case study, we apply these ideas to Labyrinth, a domain specific language oriented to the design of Web applications. The analysis is done by a transformation into the Petri nets formalism, and then performing model checking on the coverability graph. The framework is supported by the meta-modelling tool AToM³.

1 Introduction

Domain Specific Languages (DSLs) are becoming popular due to its capability to capture high-level, powerful abstractions of well-studied application domains, and have the potential to increase the user productivity for modelling tasks. Since they are close to the application domain, they are less error-prone than other general-purpose languages and easier to learn, also because the semantic gap between the user's mental model and the design model is smaller. The Web is a typical domain where the use of DSLs is successful [8, 5, 15], as there are domain specific terms (e.g. node, link) not provided by general-purpose languages.

Due to the fact that Web systems provide specialized services that cannot be performed by all the system users, correctness of access control policy for Web becomes a crucial issue. Access control is used extensively in information systems as a security mechanism for protecting sensitive information and resources from unauthorized access. The access policy requires to be expressed during the design stage, using the same abstraction level as the one used to capture the system description, instead of delaying access control to the implementation phase. In addition, this integrated, abstract expression allows to validate the access policy at design time, so that inconsistencies can be detected and corrected.

In order to validate system designs, a common approach is transforming the models into semantic domains (e.g. Petri nets, process algebra or logic) for analysis. Formal methods [4,6] are techniques based on mathematics or logics that help to specify and verify systems. They are usually applied on the early phases of the development, when the cost of fixing errors is lower [4]. Although they offer significant benefits in terms of improved quality, they are not broadly used due to several reasons, among them their high cost and the need of expert personnel in a certain formal method. This expert knowledge is seldom found among the average software engineers.

In this paper, we follow a transformation-based approach for the verification of security policies in Web designs that hides the complexities of the underlying formal methods from the developers, who specify the Web system in a well-known DSL. Internally, these models are transformed into a semantic domain for further analysis, and feedback is given back to them. The verification process is responsibility of the DSL designer. He specifies the DSL by using meta-modelling, and for the verification, he defines triple graph transformation systems that perform the transformation into the semantic domain and create mappings between the elements in both models. Besides, he can define graphical triple patterns in order to specify how the results of the analysis are shown back to the user in the original notation, that is the Web DSL.

The paper follows an example-driven approach by applying the verification framework to Labyrinth [7], a DSL oriented to the design of hypermedia and Web systems. In particular, we have designed a transformation from Labyrinth into Petri nets [20] in order to analyse system properties, such as the availability of navigational paths or hypermedia objects, taking into account the applied role-based access control (RBAC) model [2]. The approach can be adapted to other Web-oriented notations, as it lies on a meta-modelling framework that does not depend on the DSL.

The paper is organized as follows. Section 2 introduces Labyrinth, which is used as case study throughout the paper. Its use is illustrated through the modelling of ARCE, a Web system for emergency management. Section 3 presents our approach to the verification of security policies with back-annotation of results, and how these ideas have been implemented and applied to ARCE. Section 4 compares with related research. Finally, section 5 ends with the conclusions and future work.

2 Security Modelling in Web Systems. A Case Study: Labyrinth

For the purposes of this work we use the Ariadne Development Method (ADM) [8], a Web engineering method that provides a set of meta-models to specify the information structure, navigation paths, interaction mechanisms, presentation features and access control policies. The method comprises three phases: *Conceptual Design*, that is concerned with the identification of abstract types of components, relationships and functions; *Detailed Design*, where system features, processes and behaviours are specified in detail; and *Evaluation*, where a set of criteria are used to assess system usability from the evaluation of prototypes. Our work focuses on those meta-models of the *Conceptual Design* related with the definition of the access control policy.

The ADM lies on a meta-model called Labyrinth [7] that defines the hypermedia abstractions used in the different diagrams of the ADM. A simplified version of its

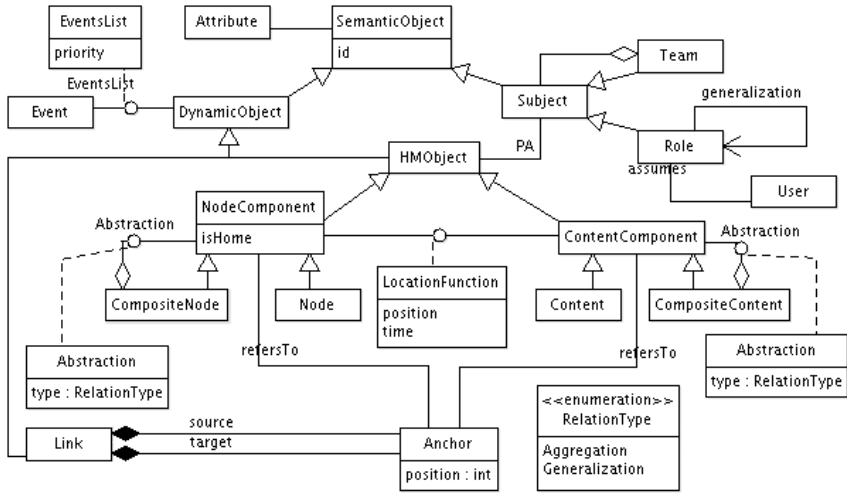


Fig. 1. Labyrinth Meta-model

meta-model with its most salient features is shown in Fig. 1. A complete description of Labyrinth and its contributions can be found in [7]. Here we give a brief description.

Structural features are specified through *nodes* and *contents*. A node is an abstract container where information contents are placed by means of the *location function*, which promotes separation of structure and content. Nodes and contents support composition mechanisms that allow to create complex information structures by using aggregation and generalization. Navigation features are expressed through *anchors* and *links*. An anchor is a link endpoint (whether source or target), and can refer to a content or a node and be shared between links. User modelling is based on the RBAC paradigm [2], and includes *users*, *roles* and *teams*. A role represents a job function within an organization, gathering a set of permissions, and a set of users allowed to exercise them. Roles decouple privileges from users, facilitating the management of authorizations due to the fact that roles are more stable than users in terms of responsibilities. Relationship *assumes* relates users with the roles that are allowed to hold, while the permission assignment (relationship *PA*) relates subjects to the nodes and contents that the role is allowed to visit. Changes in user privileges are managed through role memberships. As some roles can be more general than others, we define a generalization relation that implies inheritance: permissions assigned to the more general roles are inherited by more specific roles. A team represents a collection of heterogeneous roles and/or teams, and captures groups of interest or collaborative teams. Users cannot be assigned to teams, but instead participate in a team through role membership. A permission assigned to a team is propagated to all team components. Note that it is not necessary to explicitly define all permissions in the system, as they are propagated throughout the roles and teams structure as described in [1]. Anchors get the permission of the node or content to which are tied, while links are available for a role if at least one anchor of each end is available, that is, the role has access to the source and target link ends.

2.1 Modelling Example: The ARCE System

This section presents an example that will be used to illustrate the features of the verification framework. The ARCE system is aimed at resource management for international cooperation in case of disaster¹. When an emergency happens, the Civil Protection Department of the affected country uses ARCE to create an emergency report. If international cooperation is needed, the affected country can ask for resources to other countries, which in their turn may offer a contribution. Here we will focus on emergency report management, resource requests and resource contributions.

2.2 ARCE System Design

Part of the nodes making the Web application, as well as their structural relationships, are specified in the *structural diagram* shown in Fig. 2. Composite node Home acts as root of the system, while abstract nodes Requests and Contributions group nodes related to each activity respectively. Each leaf node corresponds to a Web page that includes the contents required to do a particular function in ARCE.

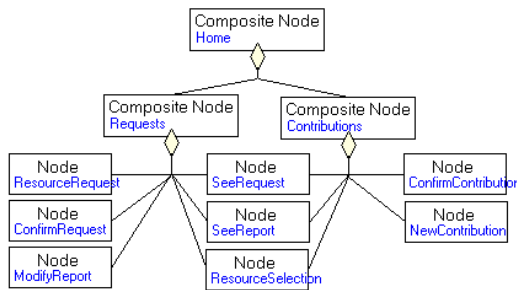


Fig. 2. Excerpt of the ARCE Structural Diagram in ADM

The *navigation diagram* in Fig. 3 captures the navigation paths by means of nodes and links. To request a resource, the user navigates from Home to Resource Request. To add a resource, the user goes to the ResourceSelection node, picks the desired resource and indicates the quantity, and finally returns back to the ResourceRequest node, where an editable list of resources is shown. When the request is ready, the user confirms it and returns Home. Optionally, the user may see or modify the emergency report during the request. To make a contribution, the user navigates from Home to NewContribution, where some details of the offer are filled and the list of resources is prepared. To add a resource to the offer, the users goes to node ResourceSelection, which in this case shows the contributor’s available resources, picks one, selects the quantity and returns to the NewContribution node.

For each node, an *internal diagram* describes its internal structure made by the contents and spatial-temporal relationships that define the presentation aesthetics. For example, the internal diagram for the SeeReport node is shown in the window to the

¹ <http://arce.dei.inf.uc3m.es>

Table 1. Access Policy for ARCE System Nodes and Contents

		Subjects								
		ARCEUser	Requ	ROp	RExp	EMgr	Contr	CExp	CMgr	COp
Nodes	Home	•	▽	▽	▽	▽	▽	▽	▽	▽
	ResourceSelection			•	▽					
	ResourceRequest		•	▽	▽	▽				
	SeeRequest	•	▽	▽	▽	▽	▽	▽	▽	▽
	ConfirmRequest					•				
	SeeReport	•	▽	▽	▽	▽	▽	▽	▽	▽
	ModifyReport				•					
	NewContribution						•	▽	▽	▽
ConfirmContribution								•		
Contents	ReportHeader	•	▽	▽	▽	▽	▽	▽	▽	▽
	GeneralData	•	▽	▽	▽	▽	▽	▽	▽	▽
	TechnicalData				•			•		

With this access policy, for example, role *Emergency Manager* can access to contents *ReportHeader* and *GeneralData* (permissions inherited from team *ARCE User*) but not to *TechnicalData*.

3 A Transformation-Driven Approach to Security Analysis

When modelling security for software systems, there is a need in verifying the correctness of the access policy in terms of the availability of different system entities. In the case of Web systems incorporating RBAC policies (e.g. *Labyrinth*), we are particularly interested in verifying the availability of navigation paths, nodes and contents for different subjects, the absence of nodes or contents that are not available to any subject, which contents are shown to each subject, and checking if it is possible for a subject to reach a node from which no link to any other node is available (i.e. there is a *deadlock*).

In order to be able to answer these questions, we have used a common technique that consists of expressing the operational semantics of the model(s) to be analysed by using a formalism [14,19,23,24]. The used formalism must provide the necessary tools to answer such questions. In this case, we transform the *Labyrinth* designs into Petri nets [20], which provide analysis techniques to investigate system properties such as *deadlock*, *reachability* of states and *invariants*, which are the kind of properties we are interested in. The transformation is performed by a triple graph transformation system (TGTS) [13] that builds, from a *Labyrinth* model, the equivalent Petri net for a system subject. Once the net is obtained, we use analysis techniques based on the *reachability/coverability* graph, as well as *model checking*, in order to verify system properties. Finally, the results are back-annotated and shown to the user in the original notation, which is the one he knows. Next subsections explain this process in detail.

3.1 Transformation from Labyrinth into Petri Nets

Graph transformation [9] is an abstract, declarative, visual, formal and high-level technique to express computations on graphs (and therefore on models). Roughly, graph transformation systems are made of rules having graphs in their left and right hand sides (LHS and RHS respectively). In order to apply a rule to a *host graph*, a morphism (an occurrence or match) of the LHS has to be found in it. Then, the rule is applied by substituting the match by the rule's RHS. The execution of a graph grammar consists of a non-deterministic application of its rules to an initial graph, until no rule is applicable. In addition, rules can be equipped with application conditions that restrict their applicability. One of the more used is the so-called Negative Application Condition (NAC). This is a graph that, if found in the host graph, forbids the rule application. Finally, we can combine meta-modelling and graph transformation allowing abstract graph nodes to appear in rules [10]. In this way, nodes can be matched to instances of any subclass, greatly improving the expressive power of rules.

In model-to-model transformation, it can be useful to manipulate triple graphs instead of the usual ones. Triple graphs are made of three different graphs: the source graph (of the transformation), the target one, and a correspondence graph that relates the elements in the other two graphs. Similarly to graph grammars, TGTSs [13] are used in order to manipulate triple graphs.

In this paper, we have defined a TGTS that builds a Petri net from a Labyrinth model. By using a TGTS we create correspondences between the Labyrinth and the resulting Petri net elements that facilitate the back-annotation of the analysis results in terms of the Labyrinth notation. Fig. 5 shows some rules of this TGTS. The source graph (shown in the upper part of the rules) corresponds to Labyrinth, while the target graph (lower part of the rules) is the Petri nets formalism. We use a compact notation for the rules, in which the LHS and the RHS are presented together. The elements to be added by the rule application (i.e. those that belong to the RHS but not to the LHS) are shown in a gray area and labelled as "new". NACs are omitted for clarity, and in all the rules are equal to the RHS. The general idea of the transformation is creating a net that simulates the behaviour of a subject (i.e. a role or team). For this purpose, a Petri net place is created for each node and content for which the subject is granted, and links between them are transformed into Petri net transitions. The subject is represented as a token; therefore, if a token is in a certain place, that means that the subject is accessing to the node or content(s) that the place represents. Note that visiting a node implies visiting all authorized contents for the subject, and thus, the Petri net marking gives the set of nodes and contents accessed in a given moment by the subject.

The three rules to the left in Fig. 5 perform the flattening of the subject's hierarchy. Rule *Flattening1* creates a correspondence element *CElem* with a morphism to the subject for which the Petri net is calculated. The rule receives such subject as parameter. This is an abstract rule: therefore, the rule is applicable to any subclass of subject (i.e. classes *Team* and *Role*). Then, rules *Flattening2* and *Flattening3* traverse the subject's hierarchy creating correspondence elements with morphisms to each subject's ancestor.

Rule *HMObject2Place* in Fig. 5 creates a place for each hypermedia object (i.e. node or content) to which the subject or its ancestors have permission to access (i.e. a correspondence element to the subject was created by the execution of the flattening

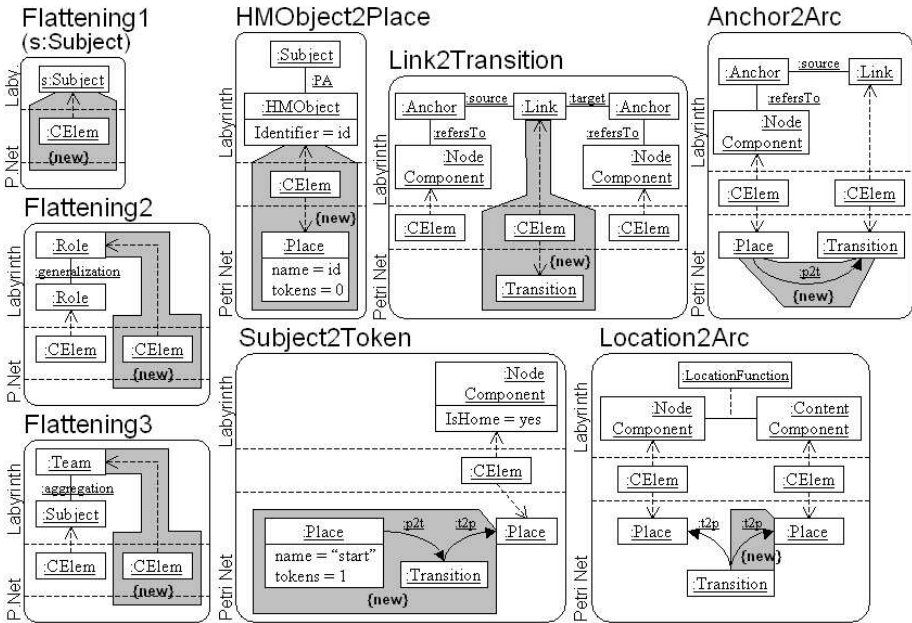


Fig. 5. Some Triple Rules of the Transformation from Labyrinth to Petri Nets

rules). Similarly, rule *Link2Transition* creates a Petri net transition for each link. We only transform those links between hypermedia objects for which the subject has access, that is, objects that have a correspondence element created by previous executions of rule *HMOject2Place*. Another similar rule is used for the case in which the source of the link is a content component instead of a node component. Rule *Anchor2Arc* creates an incoming arc from a place to a transition if the corresponding node is source of the corresponding link. Two similar rules create an arc when the source is a content component, or when the node is target of the link (creating in that case an outgoing arc from the transition). Rule *Subject2Token* creates an extra place with a token (the subject) and a transition from it to the place related to the home page. This transition models the first access of the subject to the system. Finally, each time a transition is fired (i.e. each time a link is traversed), a token must be placed not only in the target node, but also in the target node contents for which the subject is granted. For this purpose, rule *Location2Arc* creates the appropriate arcs to such contents. Similarly, leaving a node implies leaving its contents. Again, two similar rules create the necessary arcs.

Note that some system information is lost in the proposed transformation (e.g. the position of anchors), however, such information is useless for our analysis purposes. Thus, we require from the transformation to preserve the properties under investigation (availability of navigation paths depending on the security policy), as we do in this case.

Fig. 6 shows the Petri net resulting from applying the presented TGTS to the role Emergency Manager in the ARCE Web example.

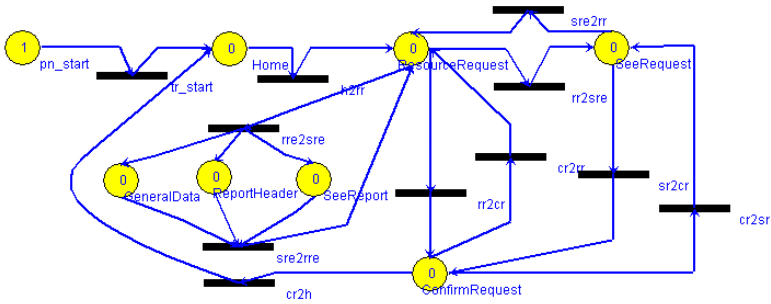


Fig. 6. Resulting Petri Net for Role Emergency Manager

3.2 Analysis and Back-Annotation

In order to verify a property, we first apply to the Labyrinth model the presented TGTS and create the Petri net for a certain subject. Then, we obtain the net's coverability graph (an approximation of the possibly infinite net state space), and apply model checking [20] on the graph in order to verify the property. Properties are expressed in Computational Tree Logic (CTL). CTL formulae are made of atomic propositions, the names of the places of the Petri net, which are true for a given net state if the place contains at least one token. Propositions can be combined with boolean connectors (\wedge , \vee , \neg), path quantifier operators that express if predicates are fulfilled starting from a certain state, and temporal quantifier operators that describe the properties of a branch in the computation tree. Valid path quantifiers are E (exists a path) and A (for all the paths). Valid temporal quantifiers are X (in next state) and U (until). Other quantifiers, such as F (in some state in the future) and G (always), can be expressed in terms of X and U . The result of checking a property on a model is the set of states satisfying the given property.

In the case of Labyrinth, we are interested in verifying the following properties:

1. *A specific navigational path is allowed for a given subject.* Let s be a subject, and $Np = \langle node_1, node_2, \dots, node_N \rangle$ a navigational path where subindex i specifies the order in which nodes are visited. Np can be expressed as the recursive function $next(i) = node_i \wedge EX (next(i+1))$ if $i < N$, and $next(i) = node_i$ if $i = N$. Thus, this property can be written as the CTL expression $E True U (next(1))$, which is evaluated on the coverability graph of the Petri net obtained for subject s . This property allows checking if a subject can perform a task that implies traversing certain navigation path. For example, in order to validate a contribution, the role Contributor Manager should be able to go from node Home to node ConfirmContribution and then return Home.
2. *A specific node or content is never shown to a given subject.* Let s be a subject and hmo a node or content. Then, this property can be expressed as $\neg (E True U hmo)$, which allows detecting elements that should be available for a subject but are not,

as well as checking if a subject has more permissions than required. For example, as no other role than `RExpert` can modify a report, the node `Modifyreport` must not be shown to any other subject.

3. *A specific node or content is never shown.* Let S be the set of system subjects, and hmo a node or content. Then, this property can be expressed as $\bigwedge_{s \in S} \neg (E \text{ True } U \text{ hmo})$. This is an extension of property number 2 to the set of subjects of the system.
4. *A specific node or content is shown in each navigational path for a given subject.* Let s be a subject, and hmo a node or content. Then, this property can be expressed with the CTL expression $A \text{ True } U \text{ hmo}$, which checks that all the possible navigation paths followed by a subject will show the specified object. For example, node `Home` should be accessed in any possible path.
5. *A subject does not reach a deadlock state.* In other words, all nodes define at least one outgoing link. Let s be a subject, then the property can be written as $\neg (E \text{ True } U \text{ deadlock})$, where predicate *deadlock* becomes *true* in states with no successor.
6. *A specific node shows at least one content for a subject.* Let s be a subject and n a node. Then the expression n gives the Petri net markings that satisfy the expression. Note that if the node has a token, then its contents for which the subject is granted have also one and belong to the marking. This property allows detecting nodes that are empty for certain subjects due to a bad design of the access policy.

In order to hide the analysis process to the Web designer (who is proficient in the Web domain and the used DSL) we back-annotate the results to the Labyrinth model. This is possible since we maintain in the correspondence graph the relations between the elements in the source and target models. The elements to back-annotate can be specified as a triple pattern that receives as parameters the Petri net states or transitions to back-annotate, and as output the Labyrinth elements resulting from the back-annotation. For example, Fig. 7 shows to the left the triple pattern used to specify how results are shown to the user in the case of property type number 6. The pattern is executed for each place (the input) obtained as result of the analysis. The output is the set of contents related to those places. Similarly, the pattern to the right in the same figure is used for the back-annotation of properties number 2, 3 and 4. The analysis method used for these properties returns the sequence of firing transitions that leads from the system initial state to the analysed node or content. These transitions are the input of the pattern. The output is the set of links related to the transitions, together with their sources, targets and corresponding anchors. Thus, in the Labyrinth model will be shown each possible navigational path leading to the hypermedia object under study.

3.3 Tool Support

The whole verification framework has been implemented in the `AToM3` tool [17], which allows the specification of visual languages by means of meta-modelling, and the manipulation of graphs by means of graph transformation. Recently, the tool has been enhanced with the possibility of expressing multi-view visual languages [12], which are languages made of different diagram types, such as the presented ADM (or the general purpose UML). Thus, we have defined the whole Labyrinth meta-model in `AToM3`, and then the different diagrams types as subsets of it. The tool provides syntactic and static

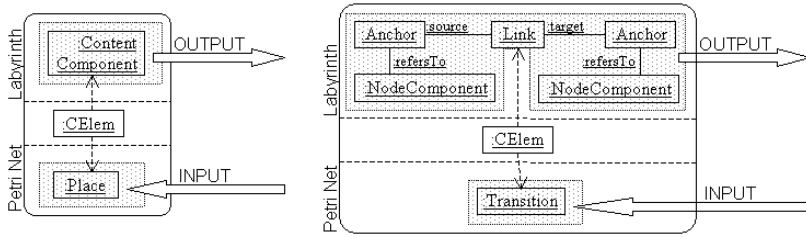


Fig. 7. Some Back-annotation Triple Patterns

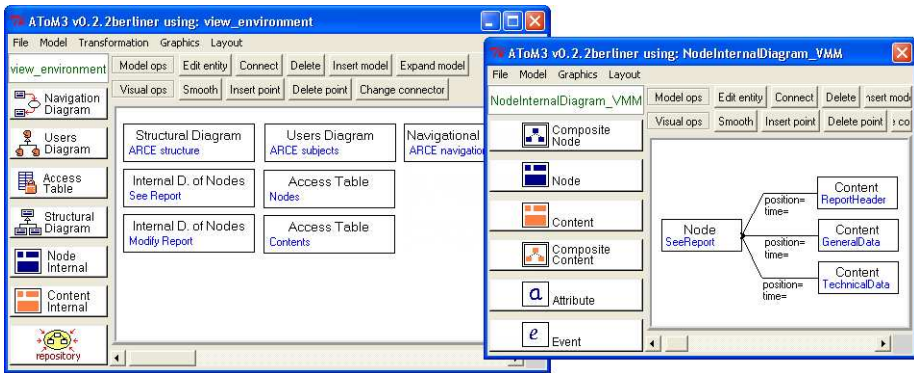


Fig. 8. Generated Environment for Labyrinth

semantics consistency between diagrams by means of automatically generated TGTS that build a *repository* made of the gluing of the different diagrams. For the analysis of the dynamics, it is possible to define semantic views to be generated by means of TGTS from the models. In the case of Labyrinth, we have defined the presented TGTS to transform the repository into Petri nets. In addition, each property to be verified in the semantic view can be specified with: (i) a pre-process method where the request of data required for the analysis (e.g. the name of the subject for which the property is checked) is specified; (ii) an analysis method call; and (iii) a back-annotation mechanism specified either procedurally or by triple patterns. For the analysis of properties in Labyrinth, method calls use analysis functions that calculate the coverability graph in AToM³ and use a model checker implemented in the tool as well. It is up to future work the use of external Petri net tools to perform the analysis.

Starting from this definition, AToM³ generated a modelling environment that allows specifying instances of the different diagram types. This environment is shown to the left of Fig. 8, where the different diagrams of ARCE have been defined. The figure shows to the right the editing of the internal diagram of node *SeeReport*.

The environment automatically creates a repository with the gluing of the system diagrams. In the repository interface (background window in Fig. 9), a button is generated for each Labyrinth concrete class and for each analysis property. Buttons derived

from classes allow adding new entities to the repository. Buttons derived from properties allow checking properties, and the result is shown according to the defined back-annotation mechanism, hiding the internals of the analysis process. If the back-annotation is specified by a pattern, the output elements obtained from its application are highlighted in the original model, as well as summarized in a dialog window. Fig. 9 shows the checking of a property in the Labyrinth repository and the back-annotation of the results. In addition, a button is generated that allows showing the result of executing the TGTS. This can be used for simulation purposes.

3.4 Verifying ARCE Access Policy

The generated environment has been used for the modelling of ARCE. The verification of the availability of navigation paths for different roles (property of type number 1 in subsection 3.2) implied just clicking on the button *Check NavigationPath* in the repository interface, which is shown in Fig. 9. The name of the subject and the sequence of nodes in the navigation path to be checked are requested to the Web designer. Then, the environment internally builds the corresponding CTL expression, performs the analysis, and the result is shown in a dialog window.

Similarly, checking to which contents of a node a subject could access (property number 6) was done by clicking on the button *Check Node Contents*. The name of the subject and node are requested to the user. Then, the node contents are shown highlighted (by the execution of the left pattern in Fig. 7) to the analysis result), as well as summarized in a dialog window. Fig. 9 shows the result obtained after checking the property for role *Emergency Manager* and node *SeeReport*. Note how content *TechnicalData*, although was specified in the internal diagram of the node, is not accessible for this role due to the specified access policy.

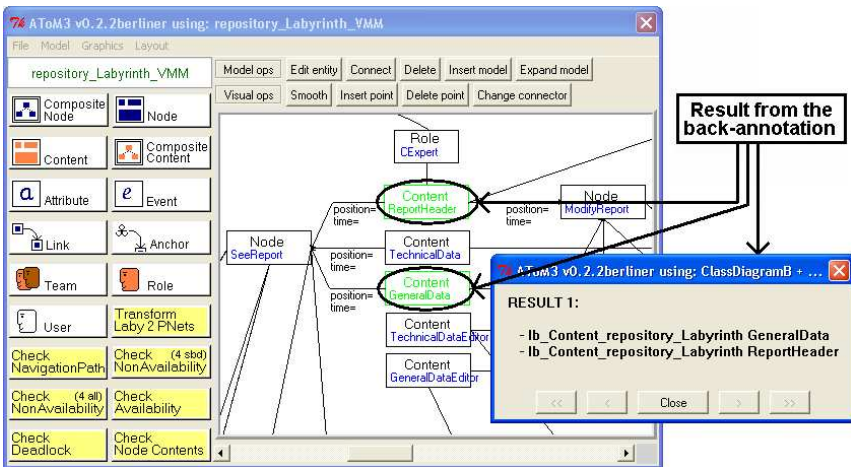


Fig. 9. Back-annotation of Property Checking. The result is highlighted in the model and summarized in a dialog window.

Table 2. Some Properties Verified in ARCE

Prop. Type	Verified Property		Expected Result	Observed Result
	Subject	CTL Expression		
1	ROp	$E \text{ True } U (\text{Home} \wedge EX (\text{ResourceRequest} \wedge EX (\text{ResourceSelection} \wedge EX (\text{ResourceRequest} \wedge EX \text{Home}))))$	true	true
	EMgr	$E \text{ True } U (\text{Home} \wedge EX (\text{SeeReport} \wedge EX (\text{SeeRequest} \wedge EX (\text{ConfirmRequest} \wedge EX \text{Home}))))$	true	false
	COp	$E \text{ True } U (\text{Home} \wedge EX (\text{NewContribution} \wedge EX (\text{SeeReport} \wedge EX (\text{ModifyReport} \wedge EX (\text{SeeReport} \wedge EX (\text{NewContribution} \wedge EX (\text{ResourceSelection} \wedge EX (\text{NewContribution} \wedge EX \text{Home}))))))))$	false	false
	CMgr	$E \text{ True } U (\text{Home} \wedge EX (\text{ConfirmContribution} \wedge EX \text{Home}))$	true	false
2	CExp	$\neg (E \text{ True } U \text{ModifyReport})$	no	no
	RExp	$\neg (E \text{ True } U \text{ModifyReport})$	yes	yes
	COp	$\neg (E \text{ True } U \text{SeeReport})$	yes	yes
	CMgr	$\neg (E \text{ True } U \text{NewContribution})$	no	no
3	–	$\bigwedge_{s \in \{\text{ARCEusr,Requ,Contr,ROp,RExp,EMgr,COp,CExp,CMng}\}} \neg (E \text{ True } U \text{SeeReport})$	no	no
4	ROp	$A \text{ True } U \text{ModifyReport}$	no	no
	ROp	$A \text{ True } U \text{Home}$	yes	yes
5	ROp	$\text{neg } (E \text{ True } U \text{deadlock})$	true	true
	CMgr	$\text{neg } (E \text{ True } U \text{deadlock})$	true	true
6	EMgr	SeeReport	yes	yes

Table 2 summarizes some properties that were checked in the ARCE design by using the presented approach and environment. For each kind of property described in section 3.2, we provide some example concrete CTL formulae, together with the expected and observed results. Results for properties 2, 3, 4 and 6 are back-annotated to the original model. For the rest of properties, the result is given as a true/false dialog window.

4 Related Work

The use of Petri nets for the formal specification, simulation and analysis of software systems (among them Web and security systems) is spread. For example, in [11] navigational paths are modelled by using Petri nets, where temporal links are also considered. [3] presents a formal XML firewall security model using RBAC based on Petri nets. In [22] security analysis of extended role based access control systems is modelled by using coloured Petri nets. In all these cases, the designer models the system directly as a Petri net, where verification is performed. In this paper we also use Petri nets for system verification, but propose the use of DSLs that include concepts especially suitable for the domain to be modelled (e.g. node, link, anchor), which makes system specification easier for the Web designer. Verification is provided by translating the specific domain models into Petri nets and then performing model checking on the net's coverability graphs. However, the transformation and analysis are hidden to the Web designer, as the results are back-annotated and shown in the original notation.

Approaches to model transformation for the analysis of systems by its translation into a semantic domain are frequent, mainly oriented to the validation of UML models [14,19,23,24], some of them providing support for back-annotations as well. Quite similar to ours is the approach followed in [23], where reference models are used to interrelate the elements of the source and target models in a single graph, allowing the back-annotation of analysis results. The reference model is similar to the notion of correspondence graph in triple graphs that we are using in this work, though our approach maintains the two graphs cleanly separated (i.e. we have separate models and meta-models for Labyrinth and Petri nets). In addition, triple graphs are more flexible as no additional structure is needed in the models in order to maintain the correspondences.

Validation techniques have been also applied to RBAC in works like [16,18] in order to check inconsistencies in terms of (non-)existence of permissions or verification of the RBAC model itself. These approaches are general or domain-independent, in the sense that the RBAC and system meta-models are separated. On the contrary, we base on a DSL that includes elements for the modelling of access policies in its meta-model. These elements are specially suite for the Web domain. Other works, such as [25,21], also allow to include domain-dependent modelling entities in the verification process. However, these inclusions seem to be done by hand, without an automatic mechanism able to transform a system model to the chosen formalism, so that back-annotations are hard to implement.

5 Conclusions and Future Work

In this paper we have presented a formal verification framework for security policies on Web systems that hides the complexity of the formalisms from the Web designer. The framework has been illustrated by its application to Labyrinth, a DSL oriented to the design of Web applications. We have designed a transformation from Labyrinth into the Petri nets formalism, which allows checking model properties such as reachability or deadlocks. The analysis of properties is made by performing model-checking of the coverability graph by using temporal logic formulae. Analysis results are back-annotated by using triple patterns.

The present work uses a simplified version of the complete Labyrinth meta-model. It is up to future work the extension to the complete meta-model, which includes for example categorization of permissions. We are also studying the transformation into Timed Petri nets that include temporal constraints (e.g. temporal anchors and security constraints). The transformation into coloured Petri nets would possibly allow to generate a single net for the whole system instead of for each subject.

Acknowledgements. Work sponsored by projects TIN2006-09678 of the Spanish Ministry of Education and Science (MEC); TSI2004-03394 of the MEC and an agreement between Univ. Carlos III (UC3M) and Dir. Gral. de Protección Civil y Emergencias (DGPCE); and ARCE, supported by an agreement between UC3M and DGPCE.

References

1. Aedo, I., Díaz, P., Montero, S.: A methodological approach for hypermedia security modelling. *Inform. Software Technol.* 45(5), 229–239 (2003)
2. ANSI INCITS 359-2004. Role Based Access Control (2004)
3. Ayachit, M.M., Xu, H.: A Petri Net based XML Firewall Security Model for Web Services Invocation. In: *Proc (547) Communication, Network, and Information Security* (2006)
4. Berry, D.M.: Formal methods: the very idea. Some thoughts about why they work when they work. *Science of Computer Programming* 42(1), 11–27 (2002)
5. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: *Designing Data-Intensive Web Applications*. Elsevier Science (2003)
6. Clarke, E.M., Grumberg, O., Long, D.: "Model Checking". In: *Proceedings of the International Summer School on Deductive Program Design* Marktobendorf, Germany, 1994. In M. Broy, "Deductive Program Design", NATO ASI Series F, vol. 152, Springer, Heidelberg (1996)
7. Díaz, P., Aedo, I., Panetsos, F.: Labyrinth, an abstract model for hypermedia applications. description of its static components. *Information Systems* 22(8), 447–464 (1997)
8. Díaz, P., Montero, S., Aedo, I.: Modelling hypermedia and web applications: the Ariadne Development Method. *Information Systems* 30(8), 649–673 (2005)
9. Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G.: *Handbook of Graph Grammars and Computing by Graph Transformation*. vol. 1. World Scientific (1997)
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: *Fundamentals of Algebraic Graph Transformation*. In: *Monographs in Theoretical Computer Science*, Springer, Heidelberg (2006)
11. Furuta, R., Na, J.: Applying programmable browsing semantics within the context of the world-wide web. In: *Proc. of Hypertext 02*, pp. 23–24. ACM Press, New York (2002)
12. Guerra, E., Díaz, P., de Lara, J. (eds.): A formal approach to the generation of visual language environments supporting multiple views. In: *Proc. VL/HCC'05*, pp. 284–286. IEEE Computer Society Press, Los Alamitos (2005)
13. Guerra, E., de Lara, J.: Attributed typed triple graph transformation with inheritance in the double pushout approach. Tech. Report UC3M-TR-CS-06-01, Universidad Carlos III (2006)
14. Guerra, E., de Lara, J.: Model View Management with Triple Graph Transformation Systems. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) *ICGT 2006*. LNCS, vol. 4178, pp. 351–366. Springer, Heidelberg (2006)
15. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. In: Lovelle, J.M.C., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) *ICWE 2003*. LNCS, vol. 2722, pp. 497–506. Springer, Heidelberg (2003)
16. Koch, M., Parisi-Presicce, F.: Visual Specifications of Policies and Their Verification. In: Pezzé, M. (ed.) *ETAPS 2003 and FASE 2003*. LNCS, vol. 2621, pp. 278–293. Springer, Heidelberg (2003)
17. de Lara, J., Vangheluwe, H.: *AToM³*: A tool for multi-formalism modelling and meta-modelling. In: Kutsche, R.-D., Weber, H. (eds.) *ETAPS 2002 and FASE 2002*. LNCS, vol. 2306, pp. 174–188. Springer, Heidelberg (2002)
18. Li, N., Tripunitara, M.V.: Security Analysis in Role-Based Access Control. *ACM Transactions on Information and System Security* 9(4), 391–420 (2006)
19. Machado, R.J., Lassen, K.B., Oliveira, S., Couto, M., Pinto, P.: Execution of UML Models with CPN Tools for Workflow Requirements Validation. In: *Proc. of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*. pp. 231–250 (2005)
20. Murata, T.: Petri nets: Properties, analysis and applications. In: *Proceedings of the IEEE*, vol. 77(4) pp. 541–580 (1989)

21. Schaad, A., Lotz, V., Sohr, K.: A Model-checking Approach to Analysing Organisational Controls in a Loan Origination Process SACMAT 2006. pp. 139–149 (2006)
22. Shin, W., Lee, J.G., Kim, H.K., Sakurai, K.: Procedural Constraints in the Extended RBAC and the Coloured Petri Net Modeling. *IEICE Trans. on Fundamentals* 88(1), 327–330 (2005)
23. Varró, D., Varró, G., Pataricza, A.: Designing the automatic transformation of visual languages. *Sci. Comp. Programming* 44(2), 205–227 (2002)
24. Xie, F., Levin, V., Browne, J.C.: ObjectCheck: A Model Checking Tool for Executable Object-oriented Software System Designs. In: Kutsche, R.-D., Weber, H. (eds.) ETAPS 2002 and FASE 2002. LNCS, vol. 2306, pp. 331–335. Springer, Heidelberg (2002)
25. Zhang, N., Ryan, M., Guelev, D.P.: Evaluating Access Control Policies Through Model Checking. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 446–460. Springer, Heidelberg (2005)

Efficiently Detecting Webpage Updates Using Samples

Qingzhao Tan¹, Ziming Zhuang², Prasenjit Mitra^{1,2}, and C. Lee Giles^{1,2}

¹ Computer Science and Engineering

² Information Sciences and Technology

The Pennsylvania State University, University Park, PA 16802, USA

qtan@cse.psu.edu, {zzhuang, pmitra, giles}@ist.psu.edu

Abstract. Due to resource constraints, Web archiving systems and search engines usually have difficulties keeping the local repository completely synchronized with the Web. To address this problem, sampling-based techniques periodically poll a subset of webpages in the local repository to detect changes on the Web, and update the local copies accordingly. The goal of such an approach is to discover as many changed webpages as possible within the boundary of the available resources. In this paper we advance the state-of-art of the sampling-based techniques by answering a challenging question: *Given a sampled webpage that has been updated, which other webpages are also likely to have changed?* We propose a set of sampling policies with various downloading granularities, taking into account the link structure, the directory structure, and the content-based features. We also investigate the update history and the popularity of the webpages to adaptively model the download probability. We ran extensive experiments on a real web data set of about 300,000 distinct URLs distributed among 210 websites. The results showed that our sampling-based algorithm can detect about three times as many changed webpages as the baseline algorithm. It also showed that the changed webpages are most likely to be found in the same directory and the upper directories of the changed sample. By applying clustering algorithm on all the webpages, pages with similar change pattern are grouped together so that updated webpages can be found in the same cluster as the changed sample. Moreover, our adaptive downloading strategies significantly outperform the static ones in detecting changes for the popular webpages.

1 Introduction

Search engine relies on crawlers to harvest webpages and the downloaded webpages are stored in the local repositories. These local copies of webpages are later retrieved to answer relevant user queries. Due to the fact that webpages change independently and that the crawling and indexing process takes time to complete, searching the Web could be somewhat similar to searching for the stars through a telescope: *what is seen is the past*. Although ideally the archiving systems and search engines may synchronize the local copies of webpages with their online counterparts, due to resource constraints it is not feasible for crawlers to constantly

monitor and download every webpage in the local repositories. In this case, a round-robin polling strategy will waste resources in downloading unchanged webpages. To be more efficient, crawlers can periodically re-visit a portion of the webpages that are more likely to have changed since previous visit.

We investigate a typical scenario in which due to resource constraints, a crawler is only allowed to periodically download a fixed number of webpages and update the corresponding local copies when a change has been found. We define this fixed number of pages as the *download resources* and the periodical interval as the *download cycle*. Thus, the crawler's goal is to *maximize* the number of updated webpages detected in each *download cycle* with the fixed amount of *download resources*. Because a crawler can verify whether a webpage has changed only *after* it has been downloaded, the challenge therefore becomes how to predict, as accurately as possible, the change probabilities for the webpages in the local repository. Those webpages with higher change probabilities will be assigned a higher priority for download.

Cho and Ntoulas proposed a sampling-based algorithm [7] to address the above challenge. In their approach, a small number of webpages are sampled from the search engine's local repository. Their counterparts on the Web are then downloaded and examined whether they have changed. Based on the results, the crawler can determine which webpages in the local repository are more likely to change, and those webpages are also downloaded. Cho et al. theoretically and empirically proposed the optimal sample size and downloading policy based on the sampling results. While their method has been proved effective in predicting and downloading updated webpages, it is carried out in the unit of a website, i.e., pages are sampled from each website and the sampling results are used to decide whether to download other pages within the same website. However, webpages' update behavior could be carried out very differently throughout a website [14]. Within one website, some webpages are more dynamic than others.

In this paper, we investigate and improve the sampling-based techniques to detect webpage changes by making the following contributions:

- We propose a new sampling algorithm to detect webpage updates. In our algorithm, sampling is done at the webpage level and each webpage is equally likely to be selected as a sample. In addition, each sample has a tunable downloading granularity.
- We propose three downloading policies for change detection. Specifically, given a sampled webpage, we exploit its linkages (*link-based downloading policy*), directory structure (*directory-based downloading policy*), or content-based features (*cluster-based downloading policy*) to detect updates on other webpages. We also show empirically that a website may not be a good granularity for change detection.
- We propose two biased sampling algorithms based on two metrics, the change history [8] and PageRank [15]. We predict webpages updates based on their change history using the first metric, and based on their popularity using the second one.

The rest of the paper is organized as follows. In Section 2 we describe the context of our work. We propose our sampling algorithm in Section 3. In Section 4 and 5 we investigate details of the parameter space in the proposed algorithm and further propose techniques to adaptively optimize the parameter. In Section 6 we present and discuss the empirical results. We outline our conclusion in Section 7.

2 Related Work

Existing studies have investigated the dynamics of the Web pages by showing that webpages have a broad spectrum of change intervals varying from a few hours, a few weeks, to a year [12,3,9,16,6]. As a result, search engines frequently update their indices and the search results could be quite unstable [17]. In order for search engines to keep up with the dynamic Web, server-side approaches require that the Web servers keep a file with a list of URLs and their respective modification dates [2]. Before visiting a site, a crawler downloads the URL list, identifies the URLs that were modified since its last visit, and retrieves only the modified pages. This approach is very efficient and avoids waste of Web server bandwidth and crawler resources, but requires modifications to the server-side implementation.

On the other hand, client-side techniques are independent of server-side implementation. First, probabilistic models have been proposed to approximate the observed update history of a webpage and to predict its change in the future [10,13]. Most of these change-frequency-based refresh policies assume that the webpages are modified by *Poisson processes* [18]. Besides page's change frequency, several metrics have been proposed to guide the crawler how to choose webpages to re-download, such as *freshness* and *age* [10], “*embarrassment*” metric [23] and user-centric metric [22]. However, a limitation common to all these approaches is that they need to gather enough accurate historical information of each webpage.

To address this problem, sampling-based approaches are proposed to detect webpage changes by analyzing the update patterns. Their sampling method samples and downloads data in the unit of a website. Specifically, they proposed a greedy downloading policy, in which a number of pages are first sampled from each website. After getting the change status of the samples, the crawler re-visits all the webpages in the website with the largest number of changed samples, and then the website with the second largest number of changed samples, and so on, until the download resources are used up. Different from their approaches, we analyze sampling and downloading in different granularities, which is based on webpages linkages, directory structure, and content-based features.

Recently, a classification-based algorithm [1] takes into account both the history and the content of pages to predict their change behavior. Their experiment results on real web data indicate that their solution had better performance than the change-frequency-based policy [9]. The key difference between their algorithm and ours is that they assume the crawler knows the change history of

a set of webpages. And these webpages are used as training data to learn the change pattern of other new pages. While in our approach, we do not assume the crawler have any existing historical information. Therefore, our approach is more practical than the classification-based method.

3 Sampling-Based Update Detection

We present a sampling-based update detection algorithm based on the assumption that *webpages that are relevant (discussed later) to an updated webpage are also likely to be updated as well*. The goal of the algorithm is to *maximize* the number of updated webpages downloaded in each *download cycle*, with a fixed amount of *download resources*. Please note that our sampling-based algorithm can detect an “*update*” including the creation, change, or removal of a webpage. In this section, we describe in detail the sampling and downloading process of our change detection algorithm. This process constitutes the main part of the algorithm.

Let L_0 denote the initial set of webpages in the local repository during the initiation of the algorithm. We also denote the *download resources* for each *download cycle* as R , and subsequently updated versions of L_0 in the following n download cycles as $L_i (i = 1, \dots, n)$. Algorithms 1 and 2 show the sampling and downloading process during each download cycle, respectively. There are two tunable parameters, the *download probability* φ and the *download granularity* d , which are explained in details in Section 4 and 5. The algorithm proceeds as follows. The sampling process randomly chooses a set of pages as *samples* from the local repository. For each sampled page $p_s \in L_{i-1}$, its current version on the Web is downloaded and compared with the local version p_s . Based on this comparison, the process triggers the downloading process with the probability φ . During the execution of the downloading process, p_s is used as a *seed* page and its neighbors within d are downloaded. Please note that in the downloading process p will be downloaded in either of the following cases. In the first case, p is downloaded as a **sample**. In the second case, p is a **neighbor within distance d of a changed sample**, and is downloaded with a probability φ .

4 Tuning Download Granularity

The download granularity d determines that, given a changed sample, which additional webpages and how many of them the crawler should choose to download. We propose three definitions of d and investigate which definition gives us the best performance, in terms of maximizing the probability of discovering and downloading webpages that have been updated.

4.1 Link-Based Downloading Policy

Consider the directed graph created from the World-Wide Web where the nodes are webpages and a directed edge between two nodes p_1 and p_2 corresponds to

Algorithm 1. Sampling Process in the i^{th} Download Cycle**Input:** L_{i-1} , download probability φ , download granularity d , download resources R **Procedure:**

- 1: **repeat**
- 2: Randomly sample a webpage p_s from L_{i-1} ;
- 3: Download p_s and check its change status;
- 4: With probability φ , $L_i \leftarrow \text{Download}(p_s, d)$;
- 5: **until** R is used up
- 6: **if** $|L_i| < |L_{i-1}|$ **then**
- 7: $L_i \leftarrow (L_{i-1} - L_i)$;
- 8: **end if**

Output: L_i **Algorithm 2.** Download(p_s, d)**Input:** Seed webpage p_s and download granularity d **Procedure:**

- 1: download a set of webpages P^d within d from p_s ;

Output: P^d

a link from the webpage represented by p_1 to the webpage represented by p_2 . In this policy, when the algorithm identifies a sampled webpage, say p_s has been updated, it crawls all webpages whose nodes are within a distance d of the node representing p_s . We refer to this approach as the *link-based downloading policy* (LB). The intuition behind this policy is the topical locality in the Web [11].

The depth, d_{LB} , between two webpages p_s and p can be formally defined as follows. Let DG denote the directed graph generated based on the topology of a website, by considering the webpages as nodes and the hyperlinks as the directed edges between the nodes. Let \overline{DG} denote the directed graph with the same set of nodes as DG but all the directed edges reversed. We then define d_{LB} as:

$$d_{LB} = \begin{cases} h & \text{if } p_s \rightsquigarrow p \text{ in } DG, \\ -h & \text{if } p_s \rightsquigarrow p \text{ in } \overline{DG}, \\ 0 & \text{if } p_s \text{ and } p \text{ have a common parent,} \\ \infty & \text{otherwise.} \end{cases} \quad (1)$$

where $p_s \rightsquigarrow p$ indicates that a path exists from p_s to p (i.e. p is reachable from p_s), and $h > 0$ denotes the length of the shortest path between p_s and p in the number of hops. When $d_{LB} > 0$, the crawler follows only the out-links; when $d_{LB} < 0$ the crawler follows only the in-links. d_{LB} between two siblings (i.e. webpages that share a common parent) is defined as 0.

Figure 1 depicts the partial topology of a website. If there are more than one paths from p_s , take the shortest one (e.g., p_{10}). d_{LB} of p_s 's siblings is always 0 although there are paths from p_s to them (e.g., p_5). When $d_{LB} = 1$, the crawler

in our algorithm downloads all the webpages that have $d_{LB} \leq 1$, i.e. p_s, p_4, p_5, p_6, p_7 , and p_{10} .

4.2 Directory-Based Downloading Policy

Alternatively, we define the download granularity d based on the directory structure of the Web server, and refer to this approach as the *directory-based downloading policy* (DB). In this policy, we consider the directory structure as a hierarchical tree structure and webpages as the leaf nodes (illustrated in Figure 2). Formally, let p_s denote the seed webpage at $http://u_1/u_2/\dots/u_m$, p denote a webpage at $http://v_1/v_2/\dots/v_n$, and $u_c = v_c$ denote their nearest common ancestor, i.e. $\exists c, 0 \leq c \leq \min(m, n), \forall i \leq c, u_i = v_i$ and $\forall i > c, u_i \neq v_i$. d_{DB} can then be defined as follows:

$$d_{DB} = [(m - c) + (n - c) - 2] * \text{sign}(n - m). \quad (2)$$

The constant 2 is a scaler used to leverage the value of d_{DB} so that webpages at the same level as p_s in the directory tree will have $d_{DB} = 0$. The function $\text{sign}(n - m)$ indicates whether the webpage is on the level upper or lower than the level of the seed webpage.

We use the following example to illustrate the definition of d_{DB} . Assume in Figure 2 the webpage at $http://www.abc.com/dir1/index1.htm$ is chosen as the seed p_s for the downloading process. The values of d_{DB} from p_s to other webpages are also shown in the figure. Intuitively, the download granularity is a directory on the Web server. For example, $d_{DB} = 0$ indicates downloading all the webpages in the same directory as the seed; $d_{DB} > 0$ indicates downloading every webpage in the *upper* directories; and $d_{DB} < 0$ indicates downloading every webpage in the *lower* directories.

4.3 Cluster-Based Downloading Policy

We now cast the problem of discovering webpages that are updated concurrently with a seed page into a clustering problem, which we refer to as the *cluster-based downloading policy* (CB). There are two main steps in this policy. We first extract features that are relevant to the webpages' change pattern. Recent studies have found that some characteristics of webpages are strongly correlated with their change frequencies. For example, Douglis et al. [12] noted that actively-changing webpages are often larger in size and have more images. Fetterly et al. [14] observed that the top-level domains of the webpages are correlated with their change patterns. Based on previous work and our observations, the feature vector used for the clustering algorithm include the following features: content of the webpage and URLs (see details below), number of non-markup words, number of incoming and outgoing links, number of images, name of the top-level domain in the URL, depth of the URL (i.e. number of slashes), size of the file, etc. We construct a word-level vector to represent the content of the webpage. To avoid its dominance over other features, we first cluster all the webpages based on the

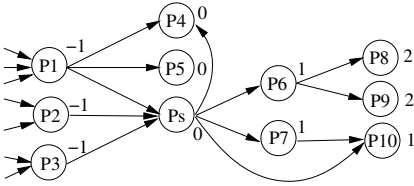


Fig. 1. d_{LB} from p_s to other webpages in the link topology. Webpages further away from p_s have larger $|d_{LB}|$ s.

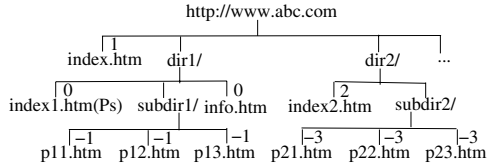


Fig. 2. d_{DB} from p_s to other webpages in the directory structure. Under DB , d_{DB} is defined at a directory level.

$tf \cdot idf$ vector of all the words. The IDs of the 10 largest clusters are chosen as the labels of 10 dimensions in the feature vector. Then for each webpage p , the values of these 10 dimensions are computed as follows. Suppose the webpage belongs to the cluster C_p , the feature’s cluster label is C_f . The value of the feature f_{C_f} is computed as follows: $f_{C_f} = 0$ for the webpages in $C_p = C_f$, $f_{C_f} = 1$ for the webpages in C_p which has the shortest distance to C_f , $f_{C_f} = 2$ for the webpages in C_p which has the second shortest distance to C_f , and so on. We use the distance of the centroids of each cluster to represent the distance of two clusters. A similar method is used to generate another 10 dimensions of the feature vector, using all the words in the URLs.

Second, with each webpage represented by a feature vector, we apply the Repeated Bisection Clustering algorithm [20] to construct hierarchical clusters, each of which contains webpages with a similar update pattern. A k -way clustering solution is obtained via a sequence of cluster bisections: first the entire set of webpages is bisected, then one of the two clusters is selected and further bisected; such process of selecting and bisecting a particular cluster continues until k clusters are obtained. Note that the two partitions generated by each bisection are not necessary equal in size, and each bisection tries to ensure a criterion function is locally optimized. Based on the clusters obtained from the algorithm, we use the distance between the centroids of two clusters to compute d_{CB} , similar to the method used to compute the values of the URL/webpage features.

5 Adaptive Download Probability

In Algorithm 1, a download probability $\varphi \in [0..1]$ is used. The larger the value of φ , the more likely it is that the sample is used as a seed to trigger the download process. We first consider the baseline case in which φ is a fixed binary variable $\{0, 1\}$ for all webpages. Note that the sample itself will always be downloaded regardless of its change status. Therefore, $\varphi = 0$ indicates that no webpages other than the sample is downloaded, which is equivalent to randomly download webpages from the Web. When $\varphi = 1$, all of the *neighbors* of the sample are downloaded. Both of these two extreme cases are intuitively not good downloading strategies to optimize performance. In the following discussions, we

propose adaptively assigning different φ s for various sampled webpages based on three characteristics, current change status, change history, and webpage popularity.

5.1 Adapting to Current Change Status

A straightforward improvement over random download is to adapt the *download probability* φ to the sampled webpage’s *current* change status as follows:

$$\varphi = \begin{cases} 1 & \text{if the sampled webpage has changed,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

With the above definition, whether or not a sample has changed in the current download cycle determines whether to crawl and download its neighbors. While this is an effective approach with relatively easy implementation, it considers only the most recent change status of the sample, with no regard to information such as the temporal patterns of change and the intrinsic quality of the sampled webpage.

5.2 Adapting to Change History

Here we discuss how to adapt the *download probability* φ toward the change history of a sampled webpage. Figure 3 shows three webpages with different change patterns. Assume that at time T_{i+1} these three pages are downloaded. By comparing their $(i + 1)^{st}$ and i^{th} versions, the crawler observes that all of them have changed and a naive crawler concludes that they are equally likely to change again at time T_{i+2} . However, by looking at their change history between T_0 and T_{i+1} , we observe that $P1$ has only one change, while $P2$ and $P3$ both have three changes. Moreover, $P2$ has two of its three changes at the beginning of $[T_0, T_{i+1}]$. On the contrary, $P3$ has no change at the beginning but three changes toward the end. Through these historical change patterns we could determine the ascending order of their change probabilities as $P1 < P2 < P3$.

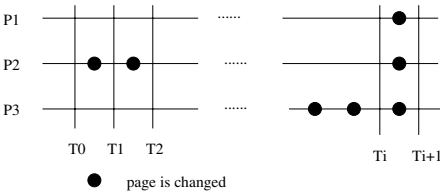


Fig. 3. Three webpages with different change patterns over time. Considering only their latest updates will lead to a misconception that they have a similar change pattern.

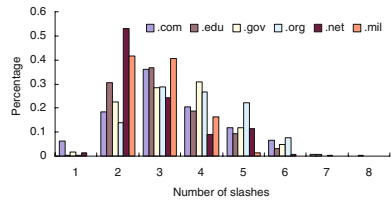


Fig. 4. Distribution of webpages by the number of slashes in the URLs for each top-level domain. Majority of URLs have depth smaller than 7.

We model the change of the webpages using a *Poisson process* [18], which has been shown effective in experiments with real webpages [4,10]. The *Poisson process* is often used to model a sequence of events that happen randomly and independently at a fixed rate over time. In the *Poisson process*, the time to the next event is exponentially distributed. In our scenario it is reasonable to assume the updates of a webpage p follows a Poisson process with its own change rate λ_p . This means a webpage changes on its own, independent of other pages. This assumption may not strictly be true but is a workable first approximation. Note that the change rate may differ from page to page. For each webpage p , let T denote the time when the next event occurs as a *Poisson process* with change rate λ_p . Then, we obtain the probability that p changes in the interval $(o, t]$ by integrating the probability density function:

$$Pr\{T \leq t\} = \int_0^t f_p(t)dt = \int_0^t \lambda_p e^{-\lambda_p t} dt = 1 - e^{-\lambda_p t}. \quad (4)$$

We set the parameter *download probability* φ to be $Pr\{T \leq t\}$ where $t = 1$, which means one download cycle. Therefore,

$$\varphi = Pr\{T \leq 1\} = 1 - e^{-\lambda_p}. \quad (5)$$

Obviously, φ depends on the parameter λ_p . We compute λ_p based on the change history of the webpage p within n download cycles:

$$\lambda_p = \frac{\sum_{i=1}^n \mathbf{I}(L_i[p])}{n}. \quad (6)$$

where $\mathbf{I}(p_i)$ is an indicator function defined as follows:

$$\mathbf{I}(L_i[p]) = \begin{cases} 1 & \text{if } L_i[p] \neq L_{i-1}[p], \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Next, in order to take into account the distribution of change events, we attach different importance to changes occurred in different download cycles. To formalize this, we extend the definition of λ_p in Equation (6) by assigning a weight w_i to each download cycle and define:

$$\lambda_p = \frac{\sum_{i=1}^n w_i \cdot \mathbf{I}(L_i[p])}{n}, \sum_1^n w_i = 1. \quad (8)$$

Typically, $w_a < w_b$ is satisfied when $a < b$, which indicates that changes occurred in the more recent download cycles are more important. When all w_i are equal, Equation (8) reduces to Equation (6). We refer to Equation (6) as the *History-Adaptive strategy* (HA), and refer to Equation (8) as the *Weighted-History-Adaptive strategy* (WHA).

5.3 Adapting to Webpage Popularity

PageRank [15] is a probability distribution to indicate the likelihood that a person randomly traversing the Web will eventually arrive at any particular

page. The PageRank of a webpage, p_i , is calculated as follows:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in IL(p_i)} \frac{PR(p_j)}{OL(p_j)}. \quad (9)$$

where p_1, p_2, \dots, p_N are the webpages on the Web, $IL(p_i)$ is the set of pages that link to p_i , $OL(p_j)$ is the number of out-going links on page p_j , and N is the total number of webpages in the local repository.

Intuitively, PageRank is a good estimator for the *popularity* of a webpage. As the third adaptive strategy, we propose to set φ for each sampled webpage in proportion to its *popularity* as reflected in the PageRank. Note that under this definition, whether to crawl and download the neighbors of a sample is independent of whether the sample has changed. We refer to this as the *PageRank-Adaptive strategy* (PRA).

6 Evaluation and Discussion

We carried out extensive experiments on a large dataset to evaluate the sampling-based update detection algorithm and the various parameter settings we proposed.

6.1 Data Collection and Evaluation Metrics

All of our experiments were carried out on a collection of real webpages from the WebArchive project¹. To obtain the historical snapshots of these webpages, we implemented a special spider to crawl the Internet Archive², which has archived more than 55 billion webpages since 1996. Excluding the webpages that were not available in the Internet Archive, we eventually constructed a dataset containing approximately 300,000 distinct webpages that belong to more than 210 websites, with their historical snapshots dated between Oct. 2002 and Oct. 2003. We have the largest number of websites in the .com domain and the largest number of webpages in the .edu domain (see Table 1 for the distribution by different top-level domains). Overall, our dataset is diverse enough to evaluate our proposed algorithms.

Figure 4 shows the *depth* distribution of the crawled webpages, which is measured in the number of slashes in the URLs. We can see from this figure that the majority of URLs have depth of 2, 3, and 4, while only a few URLs have depth of 7 and 8. Based on this observation, we tuned the download depth d from -6 to 8 for DB in our experiments.

The following two metrics were used in the evaluation of our proposals.

- *ChangeRatio*: The fraction of downloaded and changed webpages D_i^c over the total number of downloaded webpages D_i in the i th download cycle [12]. In our experiments we measure the per-download-cycle *ChangeRatio* C_i as well as the *average ChangeRatio* \bar{C} which is the mean C_i over all download cycles.

¹ <http://webarchive.cs.ucla.edu/>

² <http://www.archive.org/>

Table 1. Distribution of top-level domains in the collected data; .com and .edu together account for more than 70% of the webpages

domain	total	unique websites	avg urls per site
.edu	107204 (37.7%)	68	1576.5
.com	100725 (35.4%)	92	1094.8
.gov	38696 (13.6%)	23	1682.4
.org	22391 (7.9%)	16	1399.4
.net	12972 (4.6%)	12	1081.0
.mil	1998 (0.7%)	1	1998.0
Sum	284692 (including 706 misc. URLs)		

- *Weighted ChangeRatio*: The above general *ChangeRatio* metrics treat every webpage as equally important. However, in practice some webpages may be more *popular* than others so that once they have changed, such change has a higher priority to be detected by the crawler. Motivated by this observation, we also introduce a *Weighted ChangeRatio* [7] by giving different weights w_p to the changed pages, formally defined as

$$C_i^w = \sum_{p \in D_i} w_p \cdot \frac{\mathbf{I}_1(p)}{|D_i|} \quad (10)$$

in which $\mathbf{I}_1(p)$ is an indicator function:

$$\mathbf{I}_1(p) = \begin{cases} 1 & \text{if } p \in D_i^c, \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

In our experiments, we use a webpage’s PageRank [15] as an indicator for the popularity of the webpage.

6.2 Results and Discussion

We now present our findings in applying the sampling-based algorithm on the aforementioned data collection. For all of the experiments, the download cycle is set incrementally from 2 weeks, 4 weeks, to 8 weeks, and the download resources R is set to be 25K, 50K, and 100K. Note that at the end of each download cycle, our crawler chooses some webpages to download, each of which is downloaded only once during one cycle. When a webpage is crawled, the corresponding historical snapshot with the same time-stamp from the Wayback Machine is checked whether has changed or not; if there is no exact match in the archive, the version with the closest time-stamp is checked instead.

Comparison of downloading policies. To compare the performance under different definitions of *download granularity* d , we implemented our proposals, LB , DB , and CB , with the download probability φ set to 1 for the changed samples and 0 for the unchanged ones. We tuned d from -6 to 8. We present here

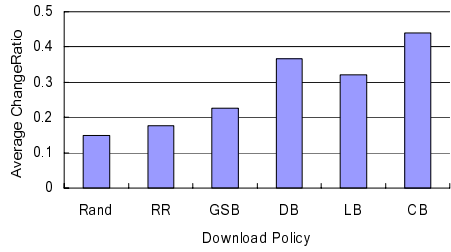
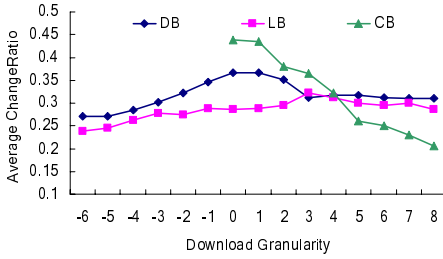


Fig. 5. Comparison of three downloading policies in \overline{C}

Fig. 6. Comparison of existing downloading policies with ours

only the setting for the download cycle to be 8 weeks and the download resource limit to be 25,000 webpages, because the results under other settings are similar. Later on, we will show more details with the different settings of download cycle and resources.

As illustrated in Figure 5, overall DB outperforms LB in terms of \overline{C} . Based on the empirical results, it is interesting to see that following the hyperlinks might not be a good strategy for the crawler to discover newly updated webpages, as webpages with similar change patterns are put in the same or a nearby directory rather than linked with each other. Compared with CB, however, DB is less effective when $0 \leq d \leq 3$. As DB is much more efficient than CB, in the following experiments, we fixed the downloading policy as DB to further investigate in detail the other facets of the parameter space.

We also compared our proposed algorithm with three other existing downloading policies. First, we used the Random Node Sampling (RNS) method proposed in [21] as a baseline in our comparison. With the RNS method, the crawler uniformly re-downloads random webpages in each download cycle. Second, we included the round-robin (RR) method for comparison, which is currently adopted by many systems [5, 19] because it is simple to implement. In RR, the webpages are downloaded in a round-robin fashion in each download cycle. The advantage of this method is that every page is guaranteed to be downloaded within a certain period of time. Finally, we implemented the greedy sampling-based technique (GSB) over site level, setting the sample size to be \sqrt{Nr} as proposed in [7]. Here N is the average number of pages in all sites, and r is the ratio of download resources to the total number of webpages in the local repository. For our approaches, we use the optimal setting of d , i.e., $d_{DB} = 1$, $d_{LB} = 3$, and $d_{CB} = 0$.

We present the results in Figure 6. From this figure we can see that our approaches all perform better compared to RNS, RR, and GSB. More specifically, RR has similar performance as the baseline algorithm, RNS. The performance of GSB is better than RNS and RR. Our sampling-based algorithm can detect about three times as many changed webpages as RNS/RR, and about twice as many changed webpages as GSB, which is a sampling-based approach at a site level.

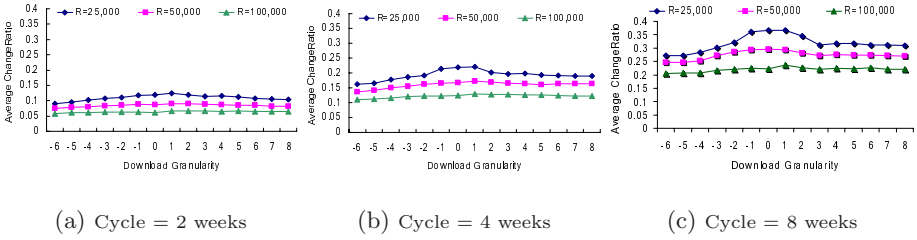


Fig. 7. *ChangeRatio* as a function of download depth d under different settings of download cycles and download resources R . Overall, a smaller download granularity and fewer download resources perform the best in a longer download cycle.

Effect of download granularity. Figure 5 has already shown \overline{C} is strongly influenced by the download granularity d . For all the downloading policies, \overline{C} goes up when d increases, and it drops after reaching its peak at $d = d_{opt}$. Empirically, $d_{opt} = [0, 1]$ for DB, $d_{opt} = 3$ for LB, and $d_{opt} = 0$ for CB. This indicates that given a changed sample, the most likely changed webpages are usually stored in the same or the upper directory of the sample, or are within three hops away following the out-links of the changed sample. CB has the highest \overline{C} when $d_{CB} = 0$ because the webpages within the same cluster have similar change pattern. Then the \overline{C} largely decreases when $d_{CB} > 0$. This is because d_{CB} is defined at a cluster level, which contains average 3,000 webpages in our experiments. Thus, the change of d_{CB} leads to significant decrease of \overline{C} when $d_{CB} > 0$. Moreover, for both LB and DB, \overline{C} with positive d is higher than that of negative d . This means that given a changed sample, changed pages can be found in the sample’s upper directories more than lower ones, and its out-links more than in-links.

The aforementioned results were obtained by setting the download cycle to be 8 weeks and the download resources to be 25,000 webpages. Next, we further evaluate the influence of d by varying the other two parameters: download cycle and download resources. We still assign the download probability φ as in Equation (3) ($\varphi = 1$ for the changed samples and 0 for the unchanged ones). Figure 7 shows \overline{C} as a function of d under different settings of the download cycle and the download resource R . From the graph, we can confirm the trend that we discussed in Figure 5. Furthermore, there are three observations. First, the longer the download cycle, the better performance in terms of \overline{C}_i . This can be explained by the fact that more webpages are likely to have changed in a longer timeframe. Second, the shapes of the three \overline{C}_i curves under the same settings are roughly the same, indicating that our proposal could be potentially applied toward crawlers with different download abilities and scalabilities. Third, it appears that a smaller R produces better results. This shows that our sampling-based algorithm works fine even if R is limited.

Impact of adaptive download probability. We present our findings on the impact of adaptively tuning the download probability φ . We tuned the download

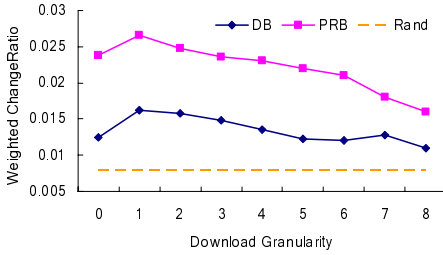


Fig. 8. Comparison of the adaptive PageRank-adaptive strategy and the non-adaptive DB in C_i^w

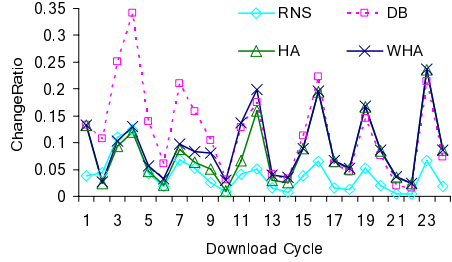


Fig. 9. Comparison of the long-term performance over 24 cycles. C_i of the two history-adaptive strategies eventually outperformed that of the DB.

probability φ with the following three adaptive strategies: *PRA*, *HA*, and *WHA*. For *PRA*, we retrieved from CPAN Google PageRank API³ to obtain each page’s PageRank, which is an integer between 0 and 10. We then set φ to be $1/11, 2/11, \dots$, and $11/11$, for webpages with *PageRank* of 0, 1, \dots , 10, respectively. For *HA* and *WHA*, we assign φ according to Equation (5). Specifically for *WHA*, to calculate the weighted λ_p according to Equation (8), we assign the weight w_i for the i^{th} download cycle as $w_i = i/(1 + \dots + n)$ where n denotes the total number of cycles the crawler has processed thus far.

We compared these adaptive strategies with DB. Here we present the results of a typical setting, i.e., the download cycle is set to 2 weeks, the download resources are set to 25,000 webpages, and d is positive. Figure 8 shows the comparison results measured in C_i^w . We see that *PRA* clearly outperforms DB by almost 100% in C_i^w .

From Figure 9, we see that at the beginning, the two *history-adaptive* strategies that take into account the webpages’ change history performed worse than DB. Specifically, from cycle 2 to cycle 5, their *ChangeRatios* were similar or even slightly worse than the random sampling policy. However, as time went by and more information about the change history could be gathered, the *ChangeRatios* for the two history-adaptive strategies gradually became higher in the following download cycles, and eventually outperformed DB. This reconfirms that in the long-run, algorithms that take into account the past changes of the webpages will have a better prediction about future changes to these webpages. Between the two history-adaptive strategies, *WHA* performed better than *HA*. Moreover, *WHA* outperformed DB in download cycle 16, earlier than *HA* did in cycle 18. These results indicate that a crawler which takes into account the change history of the webpages should pay more attention to the more recent changes than the older ones. Toward the right-hand side of the figure, the difference in *ChangeRatio* among the three investigated adaptive strategies became less significant. We believe this is because the 24 download cycles were still not long enough for the two history-adaptive strategies to demonstrate their advantages over DB.

³ <http://search.cpan.org/>

We plan to explore the history-adaptive strategy over a longer timeframe in our future work.

7 Conclusion

In this paper, we studied a challenging problem in automatic Web content archiving: how to make the local repository as up-to-date as possible under a fixed amount of available download resources? Motivated by the observation that *relevant* webpages tend to have similar change patterns, we proposed a sampling-based algorithm which uses the changed samples as the *seeds* to discover more changed webpages. We investigated in detail the parameter space and conducted extensive experiments to evaluate our proposals. We found that given a changed sample, the most likely changed webpages are usually stored in the same cluster as that of the changed sample. And the PageRank adaptive strategy is good at discovering changes to the popular webpages. Finally, for long-term performance, the weighted-history-adaptive strategy will outperform the others once the crawler has acquired sufficient information about the historical change patterns.

References

1. Barbosa, L., Salgado, A.C., de Carvalho, F., Robin, J., Freire, J.: Looking at both the present and the past to efficiently update replicas of web content. In: WIDM '05, pp. 75–80 (2005)
2. Brandman, O., Cho, J., Garcia-Molina, H., Shivakumar, N.: Crawler-friendly web servers. In: PAWS '00 (2000)
3. Brewington, B.E., Cybenko, G.: How dynamic is the Web? In: WWW '00 (2000)
4. Brewington, B.E., Cybenko, G.: Keeping up with the changing web. *Computer* 33, 52–58 (2000)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW'98, pp. 107–117 (1998)
6. Castillo, C.: Effective web crawling. *ACM SIGIR Forum* 39, 55–56 (2005)
7. Cho, J., Ntoulas, A.: Effective change detection using sampling. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, Springer, Heidelberg (2003)
8. Cho, J., Garcia-Molina, H.: Synchronizing a database to improve freshness. In: SIGMOD '00, pp. 117–128 (2000)
9. Cho, J., Garcia-Molina, H.: The evolution of the web and implications for an incremental crawler. In: VLDB '00, pp. 200–209 (2000)
10. Cho, J., Garcia-Molina, H.: Effective page refresh policies for web crawlers. *ACM TODS* 28, 390–426 (2003)
11. Davison, B.: Topical locality in the web. In: SIGIR '00 (2000)
12. Douglis, F., Feldmann, A., Krishnamurthy, B., Mogul, J.C.: Rate of change and other metrics: a live study of the world wide web. In: USENIX Symposium on Internet Tech. and Syst. (1997)
13. Edwards, J., McCurley, K., Tomlin, J.: An adaptive model for optimizing performance of an incremental web crawler. In: WWW '01, pp. 106–113 (2001)

14. Fetterly, D., Manasse, M., Najork, M., Wiener, J.L.: A large-scale study of the evolution of web pages. In: WWW '04, Budapest, Hungary (2004)
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
16. Ntoulas, A., Cho, J., Olston, C.: What's new on the web?: the evolution of the web from a search engine perspective. In: WWW '04, pp. 1–12 (2004)
17. Selberg, E., Etzioni, O.: On the instability of web search engines. In: 6th Recherche d'Informations Assistee par Ordinateur (RIA/O) Conference (2000)
18. Grimmett, G., Stirzaker, D.: Probability and Random Processes, 2nd edn. Oxford University Press, Oxford, England (1992)
19. Heydon, A., Najork, M.: Mercator: A scalable, extensible web crawler. World Wide Web 2, 219–229 (1999)
20. Karypis, G., Han, E.H.S.: Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In: CIKM '00, pp. 12–19 (2000)
21. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD '06, Philadelphia, USA, pp. 631–636 (2006)
22. Pandey, S., Olston, C.: User-centric web crawling. In: WWW '05, pp. 401–411 (2005)
23. Wolf, J.L., Squillante, M.S., Yu, P.S., Sethuraman, J., Ozsen, L.: Optimal crawling strategies for web search engines. In: WWW '02. pp. 136–147 (2002)

Auto-Generating Test Sequences for Web Applications*

Hongwei Zeng and Huaikou Miao

School of Computer Engineering and Science, Shanghai University, 200072, China
zenghongwei@shu.edu.cn, hkmiao@shu.edu.cn

Abstract. We propose a formal model, representing the navigation behavior of a Web application as the Kripke structure, and an approach to test generation. The behavior model can be constructed from the object structure of a Web application and then a set of test sequences is derived automatically from the behavior model with respect to some coverage criteria for the object structure by using the model checking's capability to construct counter-examples.

Keywords: Web application, test generation, model checking, consistency relation.

1 Motivation and Related Work

With such pervasive and rapid growth of web applications, it becomes increasingly important to ensure the correctness of Web applications through a validation and verification process. However, in today's fast-paced Web application development culture due to market pressure, testing usually falls by the wayside in practice simply because it is perceived as being too time-consuming and lacking a significant payoff. Automated testing has got considerable attention in the academic and industrial communities. Model checking, an automatic, model-based, property-verification approach, has the advantage to provide diagnostic sequences (called counter-examples) in the stack as soon as violations of properties are detected, and so provides an automatic way to generate test sequences.

This paper addresses automated test generation of Web applications. We propose a formal model that represents the navigation behavior of a Web application and can be constructed from the object structure of the Web application. Then model checking is performed to generate automatically test sequences.

Testing Web applications is a relatively new research direction. Haydar et al.[1] presented an approach for modeling and verifying a web application using communicating finite automata. Donini et al.[2] partitioned the usual Kripke structure into windows, links, pages and actions to support for automated verification of a web application. Both approaches aim at verification of a Web application, while our approach focus on testing.

* This work is supported by National Science Foundation of China (grant No. 60673115) and National 973 Program (2002CB312001).

Kung[3] modeled a web application in terms of the object, behavior, and structured perspectives and derived automatically both structural and behavioral test cases. The method proposed in [4] was based on a UML model and considered the testing and validation of the developed web system. Andrews et al. [5] used a hierarchical FSM and proposed a system-level testing technique that combines test generation based on FSM with constraints. Differing from those testing approaches, our approach enables the use of model checking and the automation of test case generation. Connections between test generation and model checking has been considered previously in the literature [6-9]. To our knowledge, however, no work applied this approach to Web application testing.

2 Modeling Web Applications

A typical Web application comprises a collection of Web pages and associated software components. A page is displayed to the user, and the navigation links toward other pages. Software components may be ASPs or JSPs, CGIs, Java Beans, even remote web services etc. There are multiple types of relationships among pages and software components. A *link* from page to page can be enabled by a attribute *href*; Frame and frameset elements make a page *consist of* several other pages; Pages can *call* components by sending requests enabled by *href* attributes or *action* attributes of *Forms*, and software components can *build* dynamically new pages as responses to the requests and so on.

Aiming at the test of the external behavior of a Web application from client's point of view, we consider only Web pages, software components interacting directly with the Web pages, and their relationships. An Object Relation Diagram (ORD) [3] is employed to represent the object structure of a Web application.

Definition 1 An $WA_O = (V, L, E)$ is a directed graph, where $V = V_{page} \cup V_{comp}$ such that V_{page} is a set of Web pages and V_{comp} is a set of software components that accept requests from Web pages or be able to build dynamic Web pages, $L = \{consist\ of, link, call, build\}$ includes four labels representing the relationship types, and $E \subseteq V \times L \times V$.

To apply model checking to generate automatically test sequences of Web applications, we choose NuSMV[10] as our model checker and propose a formal approach to modeling Web applications as a Kripke structure.

Definition 2 The behavior model of a Web application, denoted by WA_B , is a quadruple (S, R, L, S_0) . $S = S_{page} \cup S_{req}$ is a finite set of states where S_{page} is a set of Web pages and S_{req} is a set of requests sent to the application. $S_0 \subseteq S$ is the set of initial states. Reasonably, a *blank* page is used as the only initial state in S_0 where the request for the home page of a Web application can be sent. $R \subseteq S \times S$ is a transition relation such that for each $s \in S$ there is a state $s' \in S$ satisfying $(s, s') \in R$. $L: S \rightarrow 2^{AP}$ is a state labeling function that labels each state with a set of atomic propositions (*AP*) that are true.

Here, Web pages and requests are considered as states of the Kripke structure. Requests can result from typing in of a URL in the browser's address bar, clicking on hyperlinks, submitting forms and initiating frames with their *src* attributes. *AP* includes those atomic propositions specifying whether requests are enabled in Web page states

or triggered in request states. For a request r , $link-r$, $call-r$ and $src-r$ denote ‘ r is enabled’ by a *hyperlink* to a Web page, a *call* (a form action or a hyperlink to a software component) and the src attribute in a frame respectively, and $tri-r$ means ‘ r is triggered’. In addition, each Web page has an additional atomic proposition specifying the name of the corresponding page, denoted by prefixing $page-$ to the page name.

The algorithm that constructs WA_B from WA_O is outlined briefly as follows:

1. Constructs the initial state s_0 with atomic propositions $page-blank$ and $link-<home\ page>$, state s_1 with $tri-<home\ page>$ and state s_2 with $page-<home\ page>$, and then construct two transitions (s_0, s_1) and (s_1, s_2) .
2. $\forall v \in V_{page}$, constructs a state $s \in S_{page}$ with $page-v$, $\forall v \in V_{comp}$, constructs a state $s \in S_{req}$ with $tri-v$.
3. For any $(v, l, v') \in E$ in WA_O , we assume that s and s' are states of WA_B corresponding to node v and v' respectively. For edge $(v, consist\ of, v')$, a state s_n with $tri-v'$ and two transitions (s, s_n) and (s_n, s') are created, and $src-v'$ is added to state s . For edge $(v, link, v')$ is converted similar to edge $(v, consist\ of, v')$, but $link-v'$ instead of $src-v'$ is added to state s . For edge $(v, call, v')$, a transition (s, s') is created, and $call-v'$ is added to state s . For edge $(v, build, v')$, only a transition (s, s') needs to be created.

3 Generating Trap Properties

The key to the generation of test sequences by means of model checking lies in constructing a set of trap properties[2] that will cause the violation of model checking and output of counter-examples. Similar to adequacy criteria, the completeness of the trap properties must be considered. In this context, a set of trap properties is complete if it includes all trap properties derived from the WA_O of a Web application under test in terms of *consistency relation*. Typically, an application is considered consistent with its specification when it implements at least what is specified.

Definition 3 A Web application is consistent with its WA_O if (i) it can reach all objects specified in WA_O , and (ii) it implements all relationships specified in WA_O .

The first criterion requires that each node of WA_O has a corresponding reachable state in WA_B . To generate a test sequence for a node, a *trap node property*, denoting that there is not any path reachable to the state for the node, needs to be defined.

For each Web page v , there exists a state with the name of the page in WA_B , i.e., $page-v$ holds. The trap node property is defined in CTL formula as

$$\mathbf{AG} \ !page-v \tag{1}$$

However, a component acts as a bridge which receives a request from a Web page and generates another new page as the response to the request. The trap node property for a component node v requires that no any request for the component is triggered actually, which is expressed as:

$$\mathbf{AG} \ !tri-v \tag{2}$$

The second criterion requires that all edges in WA_O should be tested to check whether or not all legal navigations are implemented. Therefore, trap properties for each edge of WA_O are generated. The generation algorithm of trap edge properties is outlined briefly as follows:

- For each edge $(v, \text{consist of}, v')$ where both v and v' are Web page nodes, three states need to be considered in determining if the edge is implemented at least once in WA_B : the first state denotes that v with a frame page v' , i.e., $page-v$ and $src-v'$ hold, the second state satisfying $tri-v'$ denotes that a request for v' is sent out, and the third state in which $page-v'$ holds, meaning that the page v' is obtained and rendered. As the result, we define a trap edge property in the CTL formula:

$$\mathbf{AG!}((page-v \wedge src-v') \wedge \mathbf{EX} (tri-v' \wedge \mathbf{EX} page-v')) \quad (3)$$

- For each edge (v, link, v') , the trap edge property can be defined similarly but $link-v'$ instead of $src-v'$.

$$\mathbf{AG!}((page-v \wedge link-v') \wedge \mathbf{EX} (tri-v' \wedge \mathbf{EX} page-v')) \quad (4)$$

- For each edge (v, call, v') where v is a Web page node and v' is a component node, two states need to be considered in determining if the edge is implemented at least once in WA_B : the state in which $page-v$ and $call-v'$ hold, and the state satisfying $tri-v'$. The trap edge property is defined as:

$$\mathbf{AG!}((page-v \wedge call-v') \wedge \mathbf{EX} tri-v') \quad (5)$$

- For each edge (v, build, v') where v is a component node and v' is a Web page node, two states need to be considered in determining if the edge is implemented at least once in WA_B : the state in which $tri-v$ holds, and the state satisfying $page-v'$. The corresponding trap edge property is expressed as:

$$\mathbf{AG!} (tri-v \wedge \mathbf{EX} page-v') \quad (6)$$

4 Conclusion

This paper proposes a formal approach to test generation of Web applications by using the model checking technique. We define an implementation to be consistent with its design if the implementation does what it should do. Then, a collection of trap properties with respect to consistency relation is generated automatically from the object structure by using node and edge coverage criteria. The generated trap properties are model checked on the behavior model to reveal its inconsistencies illustrated by counter-examples that are used to form test sequences.

Now, we only consider if a Web application does what it should do. It is clearly a fault, however, if an application implements an unspecified behavior. It is necessary to complement our testing approach with verification.

References

- [1] Haydar, M., Petrenko, A., Sahraoui, H.: Formal Verification of Web Applications Modeled by Communicating Automata. In: de Frutos-Escrig, D., Núñez, M. (eds.) FORTE 2004. LNCS, vol. 3235, pp. 115–132. Springer, Heidelberg (2004)
- [2] Donini, F.M., Mongiello, M., Ruta, M., Totaro, R.: A Model Checking-based Method for Verifying Web Application Design. *Electronic Notes in Theoretical Computer Science* 151(2), 19–32 (2006)
- [3] Kung, D.C., Liu, C.H., Hsia, P.: An Object-Oriented Web Test Model for Testing Web Applications. In (APAQS 2000). In: Proceedings of the 1st Asia-Pacific Conference on Web Applications, pp. 111–120. IEEE Press, New York (2000)
- [4] Tonella, P., Ricca, F.: Testing processes of web applications. *Annals of software engineering* 14(1), 93–114 (2002)
- [5] Andrews, A., Offutt, J., Alexander, R.: Testing Web Applications by Modeling with FSMs. *Software Systems and Modeling* 4(3), 326–345 (2005)
- [6] Gargantini, A., Heitmeyer, C.L.: Using Model Checking to Generate Tests from Requirements Specifications. In: ESEC/FSE99. Proceedings of Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Toulouse, France, pp. 146–162. ACM Press, New York (1999)
- [7] Heimdahl, M.P.E., Rayadurgam, S., Visser, W., Devaraj, G., Gao, J.: Auto-generating Test Sequences Using Model Checkers: A Case Study. In: Petrenko, A., Ulrich, A. (eds.) FATES 2003. LNCS, vol. 2931, pp. 42–59. Springer, Heidelberg (2004)
- [8] Hong, H.S., Lee, I., Sokolsky, O., Cha, S.D.: Automatic Test Generation from Statecharts Using Model Checking. In: Proceedings of the 1st International Workshop on Formal Approaches to Testing of Software (FATES '01), Aalborg, Denmark, pp. 15–30 (August 2001)
- [9] Belli, F., Güldali, B.: Software Testing via Model Checking. In: Aykanat, C., Dayar, T., Körpeoğlu, İ. (eds.) ISCIS 2004. LNCS, vol. 3280, pp. 907–916. Springer, Heidelberg (2004)
- [10] McMillan, K.L., “The SMV System for SMV version 2.5.4 (October, 2006) <http://www.cs.cmu.edu/modelcheck/smv/smvmanual.ps>

A Survey of Analysis Models and Methods in Website Verification and Testing

Manar H. Alalfi, James R. Cordy, and Thomas R. Dean

School of Computing, Queens University
Kingston, Ontario, Canada
{alalfi, cordy, dean}@cs.queensu.ca

Abstract. Models are considered an essential step in capturing system behavior and simplifying the analysis required to check or improve the quality of software. Verification and testing of websites requires effective modelling techniques that address the specific challenges of web applications (WAs). In this study we survey 21 different modelling methods used in website verification and testing. Based on our survey, a categorization, comparison and evaluation for such models and methods is provided.

1 Introduction

Web applications (WAs) are evolving rapidly, using many new technologies, languages and programming models to increase interactivity and usability. This inherent complexity brings challenges for modelling, analysis, verification and testing. Some of these challenges are:

- WAs interact with many components that run on diverse hardware and software platforms. The integration of such components is extremely loose and dynamically coupled, which provides powerful abstraction capabilities to the developers, but makes analysis for testing and verification extremely difficult.
- WAs are heavily dynamic, due to dynamically generated components, dynamic interaction among clients and servers, and continual change in web technologies.
- WAs may have several entry points, and users can engage in complicated interactions that the WA cannot prevent. WAs are often interfaced to database systems and provide the same data to different users. In these cases, access control mechanisms become an important requirement for safe and secure access to WA resources, and the process of implementing and applying such rules is considered a great challenge.
- Some information in WAs is transmitted using hidden fields and special channels, due to the stateless behavior of the HTTP protocol. It's a challenge to provide a precise analysis for WAs that takes this information into account.

Most of the early literature concentrates on the process of modelling the design of WAs, using forward engineering-based methodologies designed to simplify the process of building a highly interactive WAs. Other research uses reverse

engineering to extract models from existing WAs in order to support their maintenance and evolution. This survey studies a range of different analysis models that are currently applied in the field of verification and testing of WAs. Design modelling methodologies are outside the scope of our study.

While reviewing different methods, we found that some methods focus on modelling the navigational aspects of WAs. Others concentrate on solving problems arising from user interaction with the browser in a way that affects the underlying business process. Still others are interested in dealing with static and dynamic behavior. In this paper, we attempt to categorize these methods according to the level of WA modelling - navigation, behavior, and content. In each category, methods are sorted according to the kind of notation employed. A comparison and evaluation of 21 different methods is described. The rest of this paper is organized as follows: Section 2 lists the desirable properties of WA modelling. Section 3 describes the set of comparison and categorization criteria used in our study, and gives a comparative analysis of 21 different modelling methods. Conclusions and open problems are discussed in Section 4.

2 Desirable Properties for Website Modelling

We can think of WAs from three orthogonal perspectives: web navigation, web content and web behavior. Desirable properties of WAs can fall within these three dimensions and can be classified into:

- *Static navigation properties.* Most of the early literature on web analysis and modelling concentrates on dealing with static links, treating WAs as hypermedia applications. It addresses checking of properties such as broken links, reachability, consistency of frame structure, and other features related to estimating the cost of navigation, such as longest path analysis.
- *Dynamic navigation properties.* This kind of analysis focuses on aspects that make the navigation dynamic, where the same link may lead to different pages depending on given inputs. The inputs could be user inputs transferred via forms, or system inputs depending on a state in the server such as date, time, session information, access control information or hidden fields.
- *Interaction navigation properties.* This kind of analysis focuses on properties that are related to user navigation that happens outside the control of the WA, such as user interaction with the browser. This includes features such as the back button, the forward button, and URL rewriting.
- *Static content properties.* Consistency of the web page content with respect to syntax and semantics.
- *Dynamic content properties.* This analysis requires the ability to check the syntax and semantics of dynamically generated content that results from the execution of scripts by the server. Some technologies are able to generate new connections, which may be to a remote site. In addition, new web components could be generated at run time, and these must also be analyzed.
- *Security properties.* This issue is related to access control mechanisms that can be employed on the web content or web links. It can also involve the

backend when the database contains data reserved to specific users. These properties are also tied to session control mechanisms.

- *Instruction processing properties.* This includes client- and server-side execution. Client-side execution is any process that changes the state of the application without communication with the web server. Server-side execution refers to all instructions processed on a web server in response to a client's request. A modeling method should be able to model these features and recognize whether execution is done on the server or on the client.

3 Comparison and Categorization Criteria

In our study we reviewed 21 different modelling methods that are applied in the field of testing and verification of WAs. Following is a brief description of the main comparison criteria used in our review:

1. *Feature Type.* We note the WA features that are being captured by the proposed models, and the properties that the modeling methods are capable of checking. These features are categorized into:
 - *Static Features.* This includes static properties of WAs, and is mainly concerned with links that connect an HTML page to other HTML pages. When the user clicks on a static button or link, a request is sent to the server to fetch a page. The server responds by retrieving the required page from its storage and sending it back to the client. Properties to be checked in this category relate to static navigation and static content.
 - *Dynamic Features.* These features include both dynamic links and dynamic content. Dynamic links describe the connection between HTML pages and server code that must to be executed to generate the required information, build it into an HTML page, and return it to the client. The processing done by the server may depend on user or system inputs. User inputs are usually sent by filling a form or by hidden fields in the HTTP request. System inputs depend on the server state, such as server time, or on interaction with other resources such as database servers and web objects. The output could be constructed as new content, or as a link to a new HTML page. Properties in this category are those related to dynamic navigation properties, dynamic content properties, security properties, and instruction processing properties.
 - *Interaction Features.* This includes properties related to user interaction with the browser. The browser's influence on the navigation behavior of the WAs should be taken into consideration while modelling or analyzing WAs, as the web browser provides the interface to the WAs, and can change the navigation behavior while a user browses a WA.
2. *Notation.* Modeling methods use different notations; some of them are formal, while others are either semi- or informal. The main notation used by each method is noted.

Table 1. Summary of Methods Categorized by Modelling Level

Method Name	Feature type	Notation	Level	Application	Source code required	Model optimization	Tool support
GFKF03[14]	Interaction	Abstract model, use lambda calculus	Interaction Behavior	WA interaction with the browser	No	No	Prototype
LK04 [3]	Interaction	WebCFG	Interaction Behavior	Verification	Yes	Yes	Implement a model checker
CZ04 [20]	Interaction +Static	Labeled transition	Interaction + static (Navigations)	Testing and verification	No	Yes	None
BA05 [2]	Interaction	UML(WS structure) OCL (behavior of the model)	Interaction Behavior	Verification for user interaction(Amazon + Orbitz bug)	No	Yes	UML2Alloy
ABF05 [16]	Static	Partial rewriting	Content	WS verification Tool(GVerdi)	Yes	No	GVerdi
Con99 [9]	Static	Extended UML	Structure (Navigation)	Analysis	No	No	Rational Rose Tools
BMT04 [22]	Static + dynamic	UML-meta Model + UML state diagram	Structure (Navigation)	Analysis & Testing	Yes	No	WebUML
RT00 [7]	Static	Directed graph	Structure (Navigation)	Analysis + can be use for verification & testing	No	No	ReWeb
da01 [10] and dAHM01[26]	Static	Directed graph With Webnodes	Structure (Navigation)	Verification	No	No	MCWeb
SDMP02 [5]	Static + dynamic	Web graph	Structure (Navigation)	WA design Verification	No	No	AnWeb
SDM+05[6] and CMRT06 [23]	Static + dynamic	(WAG)WA graph + extension to Kripke structure	Structure (Navigation)	WA design Verification	No	No	WAVer + SMV tools
WP03 [25]	Static + dynamic	Extended StateCharts	Structure (Navigation)	Design Verification	Yes	Yes	SWCEditor
HH06 [19] FARNav	Static + dynamic	StateCharts	Adaptive Navigation	design and implementation Verification + testing	No	Yes	Existing SVM model-checking tools
SM03 [12]	Static + dynamic	SDL	Structure (Navigation)	Testing and verification	Yes	No	Existing SDL Support tool
KLH00 [21] WTM	Static + dynamic	Control flow graph, data flow graph, and finite state machines OSD(object state diagram)	Static and dynamic Behavior, Dynamic Navigation	Testing	Yes	No	None
BFG02 [11] Veriweb	static + dynamic	Directed graph	Navigation + Behavior	WS testing	Yes	Yes	VeriSoft + web Navigator + ChoiceFinder + SmartProfiles
HPS04 [8]	Static+ dynamic	System of communicating automata	Navigation + Behavior	WA Verification	No	Yes	Fame Work with GUI + network monitoring tool + analysis tool
AOA05 [17] FSMWeb	static + dynamic	hierarchies of Finite State Machines (FSM)	Navigation + Behavior	System level testing	No	Yes	Prototype
WO02 [18]	Interaction + static + dynamic	Regular expression	Interaction + dynamic Behavior	Can be used for testing + implementation + impact analysis	Yes	No	None
TR04 [13] And TR02 [15]	Static + dynamic	(model navigation layer) + CFG (client & server code)	Structure (Navigation)+ Behavior	Testing	Yes	No	ReWeb + TestWeb
KZ06 [24]	Static + dynamic	Extended UML (UWE)	Structure (Navigation) + Behavior	Design Validation and Verification	No	No	ArgoUWE + Spin or UPPAAL

Interaction Behavior Modelling Methods

Content Modeling Methods

Navigational Modeling Methods

Hybrid Modeling Methods (More than one level)

3. *Level of Modelling.* WA modelling can be viewed from different perspectives. We compare the modelling methods here according to three basic levels: content, structure (navigation), and behavior. These three levels in turn could have a static or a dynamic flavor.
4. *Application of the Model.* In our study we focus on methods that are concerned with modelling WAs for the purpose of testing or verification; this also could include design verification.
5. Is Source Code required?. Modeling methods may require doing a white-box or a black-box analysis. This determines whether or not the existence of the source code is required for the analysis. The kind of analysis for each reviewed method is specified.

6. *Model Optimization.* Complex systems in general may have a state explosion problem or generate a large complex model. Such models require some kind of optimization. In WAs, this problem becomes a major challenge to the success of any method that attempts to model a scalable web system.
7. *Tool Support.* We list if the method being described is supported either with a proposed tool, or with a pre-existing tool.

Our study resulted in two different views of the methods we surveyed, a general categorization by modelling level, and a detailed comparison by property coverage. Table 1 summarizes the first one, where the 21 methods are categorized according to the level of WA modelling. A second comparison between methods was done based on the more specific details of methods in the same category in particular, and other methods in other categories in general. This second comparison is based on a combination of feature type and the level of WA modeling, using the comparison criteria outlined in Section 3 as desirable properties for website modeling. The second comparison could not fit in this short paper and can be found in our technical report [1].

4 Conclusions and Open Problems

Little work has been done to compare different modelling methods in the field of web development. To the best of our knowledge this is the first study which focuses on a comprehensive review and comparative study of modelling methods that are currently applied in website verification and testing. Previous work has focussed more on the development process in general, and on the design phase in particular. Our analysis is based on two sets of criteria, with results summed up in two concise tables. We found that this field is still in its infancy. While much has been done, up until now there is no complete modelling method that is able to capture all of the desirable properties of WAs at all modelling levels. An integration of different modelling methods may be required in order to generate a new complete model that could be verified using model checking. There is also a need for work on security modelling techniques that are able to deal with the complex, distributed structure of WAs, taking into account concurrent access to web servers and other shared resources.

References

1. Alalfi, M., Cordy, J.R., Dean, T.R.: A survey of analysis models and methods in website verification and testing. Technical report, School of Computing, Queen's University (2007)
2. Bordbar, B., Anastasakis, K.: MDA and analysis of web applications. In: Draheim, D., Weber, G. (eds.) TEAA 2005. LNCS, vol. 3888, pp. 44–55. Springer, Heidelberg (2006)
3. Licata, D.R., Krishnamurthi, S.: Verifying interactive web programs. In: ICASE, pp. 164–173. IEEE Computer Society, Los Alamitos (2004)

4. Sciascio, E.D., Donini, F.M., Mongiello, M., Piscitelli, G.: Anweb: a system for automatic support to web application verification. In: ICSEKE, pp. 609–616 (July 14–19, 2002)
5. Sciascio, E.D., Donini, F.M., Mongiello, M., Totaro, R., Castelluccia, D.: Design verification of web applications using symbolic model checking. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 69–74. Springer, Heidelberg (2005)
6. Ricca, F., Tonella, P.: Web site analysis: Structure and evolution. In: ICSM, pp. 76–86 (2000)
7. Haydar, M., Petrenko, A., Sahraoui, H.A.: Formal verification of web applications modeled by communicating automata. In: de Frutos-Escrig, D., Núñez, M. (eds.) FORTE 2004. LNCS, vol. 3235, pp. 115–132. Springer, Heidelberg (2004)
8. Conallen, J.: Modeling web application architectures with UML. *Communications of the ACM* 42, 63–71 (1999)
9. de Alfaro, L.: Model checking the world wide web. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 337–349. Springer, Heidelberg (2001)
10. Benedikt, M., Freire, J., Godefroid, P.: Veriweb: Automatically testing dynamic web sites. In: IWWWC (May 2002)
11. Syriani, J.A., Mansour, N.: Modeling web systems using SDL. In: Yazıcı, A., Şener, C. (eds.) ISCIS 2003. LNCS, vol. 2869, pp. 1019–1026. Springer, Heidelberg (2003)
12. Tonella, P., Ricca, F.: A 2-layer model for the white-box testing of web applications. In: IWWE, pp. 11–19. IEEE Computer Society, Los Alamitos (2004)
13. Graunke, P.T., Findler, R.B., Krishnamurthi, S., Felleisen, M.: Modeling web interactions. In: Degano, P. (ed.) ESOP 2003 and ETAPS 2003. LNCS, vol. 2618, pp. 238–252. Springer, Heidelberg (2003)
14. Tonella, P., Ricca, F.: Dynamic model extraction and statistical analysis of web applications. In: IWWSE, pp. 43–52. IEEE Computer Society, Los Alamitos (2002)
15. Alpuente, M., Ballis, D., Falaschi, M.: A rewriting-based framework for web sites verification. *ENTCS* 124, 41–61 (2005)
16. Andrews, A.A., Offutt, J., Alexander, R.T.: Testing web applications by modeling with fsms. *Software and System Modeling* 4, 326–345 (2005)
17. Wu, Y., Outt, J.: Modeling and testing web-based applications. Technical report, George Mason University (2002)
18. Han, M., Hofmeister, C.: Modeling and verification of adaptive navigation in web applications. In: ICWE, pp. 329–336 (2006)
19. Chen, J., Zhao, X.: Formal models for web navigations with session control and browser cache. In: ICFEM, pp. 46–60 (2004)
20. Kung, D.C., Liu, C.H., Hsia, P.: An object-oriented web test model for testing web applications. In: COMPSAC, pp. 537–542 (2000)
21. Bellettini, C., Marchetto, A., Trentini, A.: Webuml: reverse engineering of web applications. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 1662–1669. Springer, Heidelberg (2004)
22. Castelluccia, D., Mongiello, M., Ruta, M., Totaro, R.: Waver: A model checking-based tool to verify web application design. *ENTCS* 157, 61–76 (2006)
23. Knapp, A., Zhang, G.: Model transformations for integrating and validating web application models. In: Modellierung, pp. 115–128 (2006)
24. Winckler, M., Palanque, P.A.: Statewebcharts: A formal description technique dedicated to navigation modelling of web applications. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 61–76. Springer, Heidelberg (2003)
25. de Alfaro, L., Henzinger, T.A., Mang, F.Y.: MCWEB: A model-checking tool for web site debugging. In: WWW, pp. 86–87 (2001)

Building Semantic Web Portals with WebML

Marco Brambilla and Federico M. Facca

Dipartimento di Elettronica e Informazione
Politecnico di Milano
P.za Leonardo da Vinci 32, I-20133 Milano, Italy
{marco.brambilla, federico.facca}@polimi.it

Abstract. Current conceptual models and methodologies for Web applications concentrate on content, navigation, and service modeling. Although some of them are meant to address semantic web applications too, they do not fully exploit the whole potential deriving from interaction with ontological data sources and from Semantic annotations. This paper proposes an extension to Web application conceptual models toward Semantic Web. We devise an extension of the WebML modeling framework that fulfills most of the design requirements emerging for the new area of Semantic Web. We generalize the development process to cover Semantic Web and we devise a set of new primitives for ontology importing and querying. Finally, an implementation prototype of the proposed concepts is proposed within the commercial tool WebRatio.

1 Introduction

Evolution of Web applications toward complex Web-based Information Systems dramatically increases the complexity of the requirements and of the technological issues associated to the design and development phases. Modern Web applications comprise distributed data integration, remote service interaction, and workflow management of activities, possibly spawned on different peers. In this scenario, a wider attention to semantics of data and applications is mandatory to allow effective design and evolution of complex systems, that can be possibly set up and manipulated by different organizations. Indeed, if semantics of data and applications is known, their integration becomes more feasible. Moreover, explicit semantic annotation of Web applications can facilitate content search and access and foster a future generation of Web client that exploit the semantic information to provide better browsing capabilities to customers.

The Semantic Web is an evolution of the World Wide Web, promoted by Tim Berners-Lee to bring “semantics” to the human-readable information so as to make them machine-readable and allow better and easier automatic integration between different Web applications. To address this challenge many semantic description languages arose, like RDF, OWL and WSMML; some of them are currently W3C Recommendations. All these languages allow to formally model knowledge by means of ontologies: the resulting formal models are the starting point to enable easy information exchange and integration between machines.

These languages are suitable for reasoning and inferencing, i.e., to deduct more informations from the model by applying logic expressions. This makes the modeling task easier since not all the knowledge has to be modeled. These languages are supported by a wide range of tools and APIs, that support design of knowledge (e.g. Protégé, OntoEdit), provide storing facilities (e.g. Sesame and Jena), and offer reasoning on the data (e.g., Racer and Pellet). Based on these modeling languages, a set of querying languages have been devised too; among them, we can mention SPARQL, a W3C recommendation.

Unfortunately, although the theoretical bases and some technological solutions are already in place for Semantic Web support, the techniques and methodologies for Semantic Web application design are still rather rough. This leads to high costs of implementation for Semantic Web features, even if embedded within traditional Web applications. These extra costs are related not only to the design of the architecture and deployment of the Semantic platforms, but only to the repetitive and continuous task of semantic annotation of contents and application pages.

We claim that conceptual modeling can increase dramatically the efficiency and efficacy of the design and implementation of such applications, by offering tools and methodologies to the designer for specifying semantically-rich Web applications. In [1] we presented our vision on the needs and the opportunity of applying Web Engineering methods to the development of Semantic Web Services in the context of the WSMO framework [2]. In particular we showed how, starting from a rich and annotated model of a Web Service, it is possible to automatically generate both the implementation of the Web Service and a large part of its semantic description. Now we focus on the extension of WebML as a Model Driven method to model and develop Semantic Portals. A Web Portal is a Web site providing personalized capabilities to its visitors and is designed to use distributed and different sources. A Semantic Web Portal adopts semantic Web technologies to better integrate distributed data sources and to provide semantic descriptions of its contents so as to make them machine-readable.

The introduction of Semantic Web applications brought a new set of requirements, to be fulfilled by methodologies and tools for such applications: e.g. easy reuse of existing ontological models, support for semantic web languages, advanced ontology query paradigms, easy specification of semantically rich descriptions of services, contents, and interfaces. Some of these requirements have been addressed by existing modeling methodologies for semantic Web applications and Semantic Portals [3,4,5,6,7].

The paper is organized as follows: Section 2 specifies the new requirements of Semantic Web; Section 3 presents the case study used throughout the paper; Section 4 briefly summarizes the principles of the WebML language; Section 5 presents the extensions to the language for supporting Semantic Web features, in terms of extensions to development process, content model, and hypertext model; Section 6 exemplifies the approach on the running case; Section 7 describes the implementation experiments; Section 8 discusses the related work; and finally Section 9 draws some conclusions.

2 Requirements for Semantic Web Engineering

To collect the requirements that a Semantic Web application should comply with, we analyzed some current online Semantic Web Portals (e.g., [SQUID](#)) and we extracted the following set of needs:

- **Support of semantic languages.** Semantic Web applications should be aware of and support (i.e., be able to query and manage) different Semantic Languages and metamodels (RDFS [\[1\]](#), OWL [\[2\]](#), WSML [\[2\]](#), ...).
- **Semantic application models.** Semantic Web applications should be designed and specified by means conceptual models that include and support semantic descriptions.
- **Flexible integration.** Semantic Web applications should embrace the philosophy of flexibility and heterogeneity integration of Semantic Web.
- **Classes and instances access and queries.** Both domain ontology classes and instances should be easily and seamlessly accessible by Semantic Web applications, through appropriate querying primitives. Notice that, while queries in data-driven Web applications are only on data instances, a Semantic Web application may exploit structure querying too.
- **Inference and verification.** Ontology-based web applications should exploit available inferencing systems on ontological data, both for semantic queries and verification of data.
- **Semantic data sources.** A Semantic Web application relies on semantic data (e.g., ontologies) that offer a machine understandable data description that may be not only used to populate and generate Web pages, but also to automatically enrich such Web pages with semantic annotations.
- **Importing and reuse of ontologies.** Semantic Web applications shall allow to: *(i)* import new (possibly distributed) data conforming to the Web application ontology; *(ii)* to seamlessly integrate new ontologies, not fitting the default ontology; and *(iii)* to reuse existing and shared ontologies.

From the previous set of requirements, we derived the following requirements for the conceptual metamodels pursuing the design of Semantic Web applications:

- Metamodels should be aware of and support semantic languages.
- Metamodels themselves should be “semantic”, i.e., grant self-annotation and explicit semantic extraction.
- Metamodels should allow flexible integration of heterogeneous sources and applications.
- Metamodels should allow transformations towards a query language able to capture all the aspects of ontologies, including inference, verification, query on instances, and query on classes.
- Metamodels should easily allow: to specify semantic data sources as underlying level of the application; to exploit these sources for populating Web pages, and for (automatically) annotating such Web pages.
- Metamodels shall be able to import and reference distributed data and ontologies, aiming at the reuse and sharing of the knowledge.

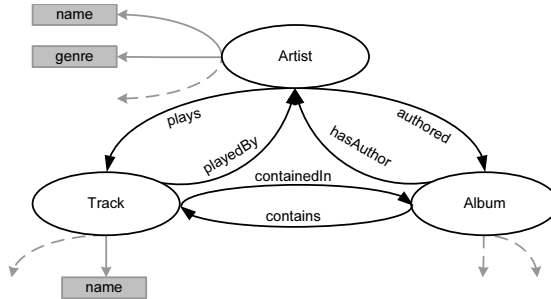


Fig. 1. A fragment of the MusicBrainz ontology representing Artist, Album, Track and their relationships

3 A Semantic Web Portal for the Music Domain

To discuss our approach, a running example will be used throughout the paper. To implement a realistic scenario, we will consider the reuse of one of the existing ontologies available on the Internet that can be easily used and integrated to create new Semantic Web Portals. We integrate two ontologies for the musical domain to build a Web application offering access to this kind of contents, considering also users profile information. In particular, we exploit the MusicBrainz ontology [12] for the music domain information; the MusicMoz [13] hierarchy to classify music genres; the RDF Site Summary [14] for music news; and the Friend Of A Friend (Foaf) ontology [15], a widely used formalism to describe for user's profiles and relationships among them. A fragment of the MusicBrainz ontology is reported in Figure 1. This application is similar to other existing Semantic Web applications (e.g., [9]), that provide personalized access to the contents exploiting distributed semantic information. The presented application, although rather simple because of space reasons, can be considered a Semantic Web Portal since it aggregates different sources of information spanned across the Internet. In Section 6 we will show how to model such application with the extended version of WebML.

4 WebML: An Overview

Our approach to Semantic Portals specification is based on the WebML language. WebML (Web Modeling Language) is a methodology for Web application design that comprises the definition of a development process and of a modeling language (composed by several metamodels for describing orthogonal aspects of a Web application) [16]. WebML offers a set of visual primitives for defining conceptual schemas that represent the organization of the application contents and of the hypertext interface. The WebML primitives are also provided with an XML-based textual representation, which allows specifying additional detailed properties, not conveniently expressible in the visual notation.

For specifying the data underlying the application, WebML exploits the Entity-Relationship model, which consists of *entities* (classes of data elements), and *relationships* (semantic connections between entities).

WebML also allows designers to describe hypertexts, called *site views*, for publishing and managing content. A site view is a piece of hypertext, which can be browsed by a particular class of users. Multiple site views can be defined for the same application. Site views are then composed of *pages*, and they in turn include containers of elementary pieces of content, called *content unit*, typically publishing data retrieved from the database, whose schema is expressed through the E-R model. WebML offers a set of predefined units for extracting data from the datasource, submitting data to the application, and specifying the navigation behaviors. Finally, WebML models the execution of arbitrary business actions, by means of operation units. An *operation unit* can be linked to other operations or content units. WebML incorporates some predefined operations for creating, modifying and deleting the instances of entities and relationships, and allows designers to extend this set with their own operations.

Units may be connected through *links*, represented as oriented arcs between source and destination units. The aim of links is twofold: defining the navigation (possibly displaying a new page or a piece of content in the same page) and the data parameters passing from the source to the destination unit.

WebML-based development is supported by the WebRatio CASE tool [17], which offers a visual environment for designing the WebML conceptual schemas, storing them in XML format, and automatically generates the running code (through XSLT model transformations), which is deployed as pure J2EE code.

5 Extending WebML Towards Semantic Web Portals

This section discusses the extensions to the WebML metamodels that are needed for complying with the new requirements of semantic web applications, according to the specifications of Section 2. The extensions apply to any aspect of the WebML methodology:

- **Development process:** extensions to describe the tasks related to the design of ontologies and semantic aspects of the web applications/services;
- **Data model:** extensions to support semantic data sources (i.e., ontologies);
- **Hypertext model:** extensions to support querying on ontologies, with particular attention to advanced and inferencing queries;
- **Presentation model:** extensions to support semantic annotations of the applications.

5.1 Extensions to the Development Process

The methodology adopted in the development of “traditional” Web applications needs to be extended with additional tasks that formalize the new design steps required by the injection of semantics within Web applications. Figure 2 depicts

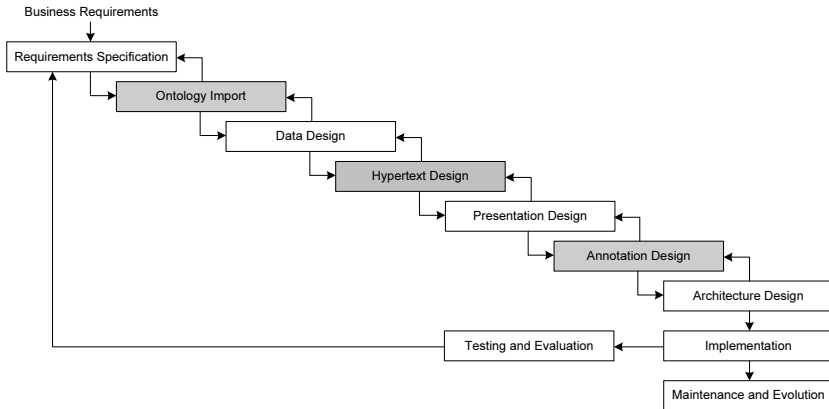


Fig. 2. The extended development process for Semantic Web applications

the extended version of the development process for Web applications. The original version was proposed in [16] and is adopted with slight variations by most of the existing Web Engineering approaches. The gray blocks represent the new tasks we introduced to fulfill Semantic Web application requirements:

- **Ontology Import.** This task addresses the selection and the importing of existing domain ontologies that may be exploited for the domain of the Web application under development. The imported ontologies can be possibly modified or merged to better suite the Web application purposes.
- **Hypertext Design.** This step, which already existed in the original approach, needs to be extended to specify how to query ontologies by means of proper primitives. Notice that the design of interactions with relational data remains unchanged.
- **Design Annotation.** In this phase the Web engineer specifies how the hypertext pages will be annotated using existing ontological knowledge. This step enriches the Hypertext Design and relies on the Presentation Design for deciding the actual position and display style of annotations.

Notice that the waterfall representation may be adjusted for some design experiences, considering that in some cases some steps are not needed at all (e.g., if only imported ontologies are necessary, the data design step can be skipped). We did not depict all the variants for sake of clarity.

5.2 Extensions to the Data Model

The existing metamodels for Semantic Web applications either evolved from existing ones by extending their data source coverage to ontologies, or have been born with native support of semantic data sources.

Although ontology support is obviously necessary for semantic Web application design, we think that relational data sources can still provide great added

value to Web applications, since the representation of every piece of information as a semantic concept is not realistic in short/medium terms. Indeed, relational databases are still effective to describe substantial parts of web applications. Therefore, allowing seamless interaction between ontologies and databases is a desiderata of current Semantic Web applications (see Section 2.2 too). Notice that now we do not aim at extending the data model of WebML so as to model ontologies (see [18]), but at allowing Web Portals to query imported semantic knowledge together with relational sources. By adopting a model-based paradigm like WebML, interaction between ontology instances and database instances can be quite straightforward and will be exploited within the hypertext model.

5.3 Extensions to the Hypertext Model

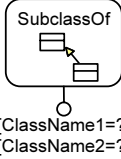
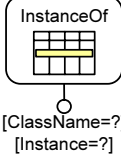
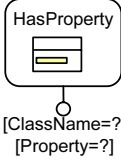
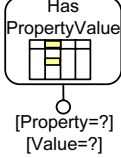
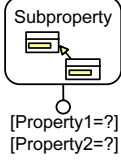
Ontological Queries Within WebML Hypertext Primitives. The main asset of WebML is the ability of effectively capturing the parameters passed between the different components (units) of the Web application. We believe that this added value is still valid even when we are considering Semantic Web Applications. Indeed, no matter if the underlying data are ontologies or databases, the basic hypertext primitives (units) and the parameter passing mechanism remain the same. Thus, parameters on the links become the actual contact point between traditional and semantic data and provide the mechanism for orthogonalizing data issues and hypertext issues.

The basic WebML primitives for data access (e.g., **Index** unit, **Multidata** unit, the **Data** unit) have a general purpose meaning and are perfectly fitting in the role of query and navigation primitives for both relational and ontology sources. They only need a few simple extensions for supporting the additional expressive power and the different data model of the ontological sources.

Let us consider the **Index** unit: besides extracting lists of relational instances, we want to use it to produce lists of instances of a particular class within an ontology model. Some requirements were highlighted about this kind of queries: (i) the possibility to show only direct instances or also inferred instances; (ii) the need for querying both instances and classes, thus mixing instances and classes in the results too. The same discussion applies to **Multidata** unit and **Data** unit. Another primitive already introduced by WebML is the **Hierarchical Index** unit. This unit acquires a first class role in the context of Semantic Web applications, because it provides a mechanism to browse and publish a portion of an ontology in a hierarchical tree representation; for instance, given a class, it allows to publish the hierarchical tree underlying it, comprising subclasses and instances.

Although the basic primitives remain valid, they require some extensions to cover the new kind of datasources. Indeed, queries on ontological data require a much wider expressive power and some different modeling rules for the information with respect to relational data. This affects the notations that the primitives must use for defining the conditions and the selection of the data. Typical examples of new features and queries of the ontological models are:

Table 1. Summary of new WebML inference units

Name	Symbol	Input	Output
subClassOf	 [SubclassOf=?] [ClassName1=?] [ClassName2=?]	c_1, c_2	true if c_1 is subclass of the class c_2
		$c_1, ?$	the list of superclasses of the class c_1
		$?, c_2$	the list of subclasses of the class c_2
instanceOf	 [InstanceOf=?] [ClassName=?] [Instance=?]	i, c	true if i is an instance of the class c
		$i, ?$	the list of classes to which the instance i belongs
		$?, c$	the list of instances of the class c
hasProperty	 [HasProperty=?] [ClassName=?] [Property=?]	c, p	true if the class c has the property p
		$c, ?$	the list of properties of the class c
		$?, p$	the list of classes having the property p
hasPropertyValue	 [HasPropertyValue=?] [Property=?] [Value=?]	p, v	the list of URIs where property p has value v
		$p, ?$	the list of possible values for the property p
		$?, v$	the list of properties with value v
subPropertyOf	 [Subproperty=?] [Property1=?] [Property2=?]	p_1, p_2	true if the property p_1 is subproperty of p_2
		$p_1, ?$	the list of superproperties of the property p_1
		$?, p_2$	the list of subproperties of the property p_2

- there is *no distinction between relationships and attributes* within the set of properties of a class. E-R style relationships might be considered as ontological properties having an *URI* as value, and attributes to ontological properties having a literal as value.
- several Semantic Web framework (e.g., OWL, RDF) assume that any instance of a class may have an arbitrary number (zero or more) of values for a particular property.
- Cardinality constraints and classes can be defined when specifying properties. In this case, it is possible to publish as values also structured objects and not only atomic attributes.

In general, while in a E-R model the selection of attributes to be published is straightforward, in the ontology model some navigation over the model may be required to publish the data. The extensions to the existing WebML primitives

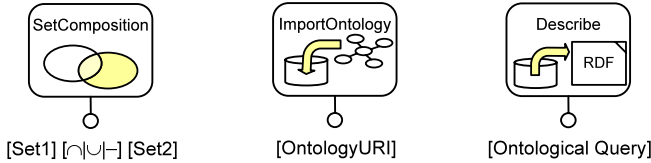


Fig. 3. Symbols of the new WebML semantics management units

provide these advantages. By means of the extended WebML primitives we can express visual queries over an ontology.

Advanced Data Access Primitives. The evolution of the basic data access primitives, introduced in the previous paragraph, is still not enough to exploit the rich set of possible queries over semantic instances and classes. Therefore, we introduce a new set of operational primitives to describe advanced queries over ontological data. We introduced these new units largely inspired by SPARQL [19] and RDF Schema syntax [11].

A first set of new units allow advanced ontological queries. The units are aggregated primitives that, depending on the type of parameters, execute differently. The complete summary of the behavior of these units is available in Table 4. These units (*SubClassOf*, *InstanceOf*, *HasProperty*, *HasPropertyValue*, *PropertyValue*, *SubPropertyOf*) aim at providing explicit support to advanced ontological queries. They allow to extract classes, instances, properties, values; to check existence of specific concepts; and to verify whether a relationship holds between two objects.

Besides the units for ontological data query, we introduce also three new units depicted in Figure 3. The *Set Composition* operation unit, is able to perform classic set operations (i.e., union, intersection, difference) over two input sets of URIs, considering the hierarchy of the URIs involved. E.g. suppose we have two set of classes: $A = \{ProgressiveRock, Jazz, Metal\}$ and $B = \{Rock, JazzFusion\}$. In this case, the set operation will give the following results: $A \cap B = \{ProgressiveRock, Metal, JazzFusion\}$ and $A \cup B = \{Rock, Jazz\}$ since *Rock* is superclass of *ProgressiveRock*, and *Jazz* is superclass of *JazzFusion*.

The *Import Ontology* unit imports at run time an ontological data source that must be consistent with one or more of the ontology models used at design time of the web application (it's validated against them before being added): according to the designer choice, it is possible to store only the url of the newly imported ontology (i.e., it will be accessed remotely for each query) or to import the ontology in the local OWL/RDF repository (i.e., it will be accessed locally, but modifications to the original data will not be propagated to the application). Notice that the navigational model of the Web application does not change at runtime, thus if the imported ontology contains new pieces (e.g., a new class unrelated with already existing classes) that were not considered in the hypertext, these pieces of knowledge will not be reachable.

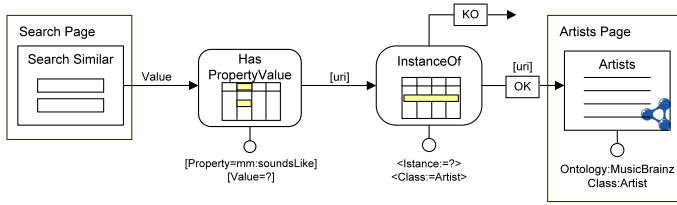


Fig. 4. A piece of Semantic Web application described by the new WebML units

The **Describe** unit returns the RDF description of an URI, thus enabling data exporting and semantic annotation of pages.

The above mentioned querying units can be used to compose reasoning tasks over ontological data: e.g., suppose that we want to find the set of common properties between two classes (e.g., **Track** and **Song**), first we can use two **HasProperty** units to extract the two set of properties characterizing the two classes; and finally we can find the common set of properties by means of the **SetComposition** unit, that will be in charge of calculating the intersection between the two sets.

Figure 4 reports a fragment of the portal that allows to retrieve artists or albums whose names sound in a similar way to the name specified by the user. The **value** submitted in the form is passed to the **HasPropertyValue** unit that extracts a set of URIs of instances (albums or artists) that have **value** as value of the *mm:soundsLike* property. The set of URIs is then passed to the **InstanceOf** unit that checks if they are instances of the class **Artist**. In this case, the URIs are passed over through the **OK** link to an **Index** unit showing list of **Artists**, otherwise the URIs are passed on the **KO** to publish a list of **Albums** (not shown in the figure).

6 Modeling the Semantic Web Portal for Music Domain

Thanks to the extensions introduced so far, we are now able to model a Semantic Web Portal scenario like the one proposed in Section 3 for the music domain. Figure 5 reports a fragment of the WebML model (including the semantic extensions) for that application. The publication units with the RDF symbol () rely on ontological data sources (e.g., *Artists* index unit), while the other units publish data from the a relational database (e.g., *User Data* data unit). The integration between the two kinds of content happen at the parameter level: once the results are transferred as parameters through links, they become homogeneous pieces of contents. The user starts his navigation from the *User Home Page*, where he can find his *Foaf Profile*; he can import a profile if it is not available yet. This part of the application actually shows how integration between ontological data sources and relational data can be achieved using parameters transported over links: when the user imports his Foaf profile, he actually stores the uri of the profile in the *User* relational entity; this uri is later used to publish his Foaf profile from the ontology repository according to the database schema.

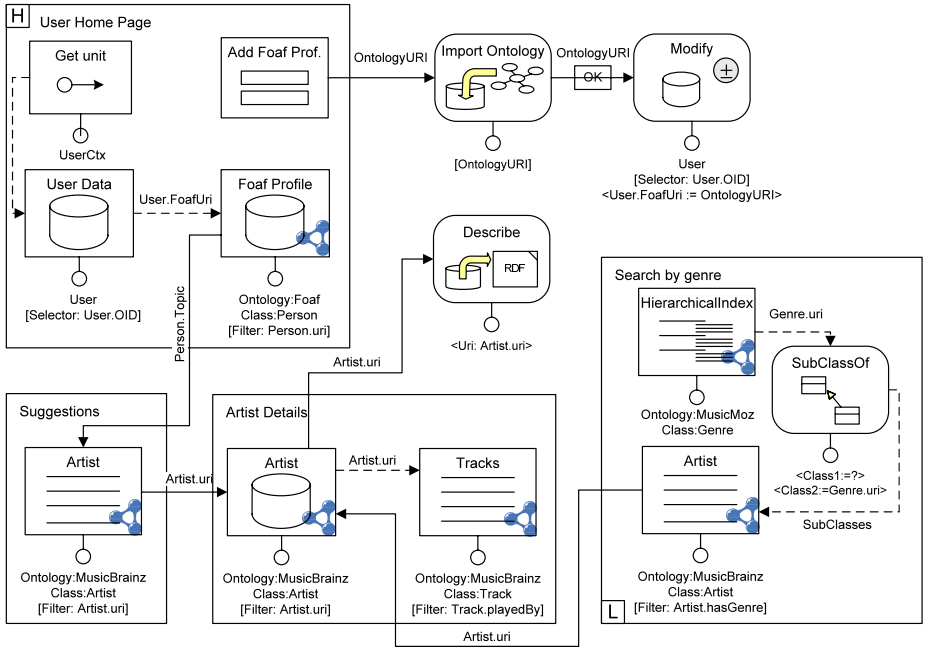


Fig. 5. A portion of a WebML diagram for a Semantic Music Portal

Navigating the outgoing link from *Foaf Profile*, the user can access the *Suggestion* page showing an index of *Artists* corresponding to his preferences. From here the user can navigate to the *Artist details* page, where detailed informations about the selected *Artist* and his *Tracks* are presented. The user can ask for the exporting of the RDF description of the artist he is currently browsing. Finally, the *Search by genre* page provides a hierarchical representation of the class *Genre*, and then displays all the artists that are related to the selected genre. The *SubClassOf* unit extracts indirect sub-genres of the chosen one, thus allowing to display associated artists.

7 Implementation and Architectural Issues

This section discusses the implementation and architectural issues related to the proposed extensions to the WebML metamodel. These issues are discussed with respect to the reference implementation of the Webratio toolsuite. For our prototype implementation we adopted the Jena framework [20] to interact with OWL/RDF ontologies. We provided reasoning support by means of the Jena integrated reasoner, and by means of the integration of Pellet [21] with the Jena framework. The design environment offered by Webratio has been extended exploiting the plug-in mechanism of the toolsuite: we devised a general purpose

```

<SWINDEXUNIT class="mf:Track" id="swinu1"
  name="Tracks" ontology="onto1">
  <DisplayedProperties
    property="mf:title"/>
  <DisplayedProperties
    property="mf:descriptor"/>
  <SortProperties order="ascending"
    property="mf:title"/>
  <Filter boolean="or">
    <FilterCondition id="fselector1"
      property="mf:playedBy"
      predicate="eq" name="Artist"/>
  </Filter>
</SWINDEXUNIT>

<descriptor service="org.webml.onto.
SWIndexUnitService">
  <onto>onto1</onto> ...
  <input-params>
    <input-param type="mf:Artist"
      name="swdau3.Artist" />
  </input-params>
  <query type="SELECT">
    DISTINCT ?instance ?p1 ?p2
    WHERE {?instance rdf:type mf:Track .
      ?instance mf:title ?p1 .
      ?instance mf:descriptor ?p2 .
      ?instance mf:playedBy ?fs1 .
      FILTER (?fs1 = $swdau2.Artist$)}
    ORDER BY DESC(?p1)
  </query>
</descriptor>

```

Fig. 6. Design time (left) and runtime (right) descriptors for a semantic index unit

ontology data access layer to be exploited by every unit; moreover, we developed a runtime Java component and an XML descriptor for each unit.

Ontological Units implementation Each unit is implemented by means of a generic class representing the runtime component that is executed for every instance of that kind of unit. Then, for each new unit (including the revisited traditional units that access ontologies) we developed an XML descriptor specifying its parameters, its properties, and the binding to the implementation classes, and so on. To better clarify the structure of the descriptor, we show an example of an ontological index unit descriptor (see left part of Figure 6). By mean of an associated XSLT transformation, design time descriptors are translated to runtime descriptors that include automatically generated template of SPARQL queries (right part of Figure 6). Units are implemented by Java components that behave according to the logics specified in the runtime descriptors, defined for each instance of the unit.

Ontology Data Access Layer. To handle interaction with ontologies we defined a new data access layer, comprising a set of general purpose Java classes to be reused by all the new units for querying the ontology repositories. These classes provide facilities to import ontologies and to select OWL/RDF classes, properties, and instances (possibly filtered by one or more conditions). The main aspects of the class structure are represented in Figure 7. The `OntologyModelService` enables connections to local and remote ontologies specified at design time or imported at runtime by mean of the `Import Ontological Source` unit. Three abstract classes offer the query services corresponding to the query methods offered by SPARQL on the ontology contents: the `AbstractSelectQueryService` class perform selection over data (SPARQL SELECT query); the `AbstractDescribeQueryService` retrieves the RDF describing a given URI (DESCRIBE query), the `AbstractAskQueryService` verifies simple predicates (ASK query). The `AbstractAskQueryService` is extended by the `AskQueryService` that is used by some of the advanced querying units to verify predicates (e.g., to check whether a class is subclass of another). In general, unit services use or implement these services.

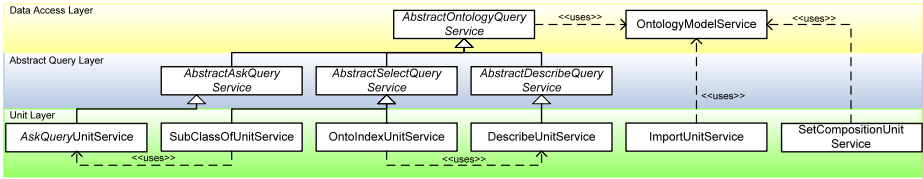


Fig. 7. A UML class diagram that shows part the of class hierarchy of the new implemented units

8 Related Works

While design methodologies for traditional Web applications offer rather mature and established solutions, Semantic Web application methodologies are still in a development phase. Realizing the benefits of the Semantic Web platform (e.g., interoperability, inference capabilities, in-creased reuse of the design artifacts, etc.) traditional design methodologies are now focusing on designing Semantic Web applications: e.g., OOHd evolved in SHDM [5]. New methodologies like XWMF [3], OntoWebber [4] and Hera [6] were specifically designed by considering the Semantic Web peculiarities.

Table 2 reports a summary that compares the features of the previously cited models for Semantic Web Portals and the WebML extensions presented in this paper. All the models, except XWMF, have a complete development methodology that covers all the needed aspects to create a Semantic Web applications. They also offer a wide support for ontology languages: basically all the models support both RDF and OWL (except for XWMF).

However, our extension is the only one that leverages on Semantic Web query languages to offer advanced query primitives that allows both query on schema and instances, together with simple reasoning patterns over data. The others models, in some cases (e.g., Hera) offers query on data schema and instances. Hera and OntoWebber offer direct to support to integration by mean of an integration model that can be used to query different data schema using the same query, while our proposal offers only a basic integration of different data sources thanks to the parameter flow between the different units in the hypertext. WebML offers the chance to integrate relational, XML and ontology data sources, while other methodologies seems to support explicitly only ontologies (off course, this issue can be solved adopting extraction techniques to import other data sources within ontologies). SHDM does not allow to import ontologies but only to create them from UML diagrams. Then, it offers a tricky way to link these ontologies to the external ones. Even if all the analyzed models are based on an ontology representation, only our proposal and a WSDM extension [22] provide an approach to annotate pages so as to make them machine readable.

Most of the new methodologies offer runtime frameworks that include or allow integration of reasoners, while some of them do not clarify if the reasoning is supported also at design time. An important factor to assure the success of a Web information System design methodology is the existence of CASE tool support, since

Table 2. Comparison of methodologies for modeling Semantic Web Portals

Requirement	XWMF	OntoWebber	SHDM	Hera	WebML+Sem.
Methodology	Partial	Yes	Yes	Yes	Yes
Semantic Model Description	Yes	Yes	Yes	Yes	Partial
Advanced query support	No	Partial	Partial	Partial	Yes
Flexible integration	No	Partial	Yes	Yes	Partial
Heterogeneous data sources	No	No	Partial	Partial	Yes
Distributed data sources	No	No	No	Yes	Yes
Reuse of ontologies	Yes	Yes	Partial	Yes	Yes
(Automatic) Annotation	No	No	No	No	Yes
Reasoning Support	No	No	Yes	Yes	Yes

a powerful methodology that is not accompanied by adequate tools will make the designer tasks very difficult to fulfill. While most of the traditional design methodologies have powerful CASE tools, no established tool support is provided for Semantic Web design, although all the cited methodologies offer some basic tools. Among them, the most complete are Hera and SHDM. Our methodology is completely supported by a commercial tool, Webratio [17] that we extended with the new components to enable design of Semantic Web applications.

9 Conclusions

In this paper we presented an extension to the WebML methodology and models for supporting the design and the specification of Semantic Web applications. The described solution provides a full coverage of the development process, and allow the designer to specify basic and advanced queries on ontological data sources, to import existing sources, and to annotate Web pages with semantic descriptions of the contents and of the models. Our approach provide substantial added value with respect to the existing frameworks for Semantic Web application design, although some of them offer more advanced solutions on some aspects (e.g., seamless integration of different ontologies). We support our proposal with a prototype implementation within the CASE tool WebRatio. Finally we showed how the proposal can be adopted to develop a Semantic Web Portal for the musical domain reusing existing knowledge.

Future work includes providing a integration layer to allow for seamless integration of different ontologies, extended testing of the new framework and integration of existing Eclipse based solutions for ontology editing with in the CASE tool.

Acknowledgments

We would like to thank Emanuele Della Valle and Irene Celino for the useful discussions on the engineering of Semantic Web Portals and their support in the definition of the case study.

References

1. Brambilla, M., Celino, I., Ceri, S., Cerizza, D., Della Valle, E., Facca, F.M.: A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, Springer, Heidelberg (2006)
2. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer-Verlag, New York, Inc., Secaucus, NJ, USA (2006)
3. Klapsing, R., Neumann, G., Conen, W.: Semantics in Web Engineering: Applying the Resource Description Framework. IEEE MultiMedia 8(2), 62–68 (2001)
4. Jin, Y., Decker, S., Wiederhold, G.: OntoWebber: Model-Driven Ontology-Based Web Site Management. In: Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L. (eds.) The first Semantic Web Working Symposium. Proceedings of SWWS'01, Stanford University, California, USA, July 30 - August 1 (2001), pp. 529–547 (2001)
5. Lima, F., Schwabe, D.: Application Modeling for the Semantic Web. In: 1st Latin American Web Congress (LA-WEB 2003), Empowering Our Web, Sanitago, Chile, 10-12 November 2003, pp. 93–102. IEEE Computer Society, Los Alamitos (2003)
6. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Eng. 2(1-2), 3–26 (2003)
7. Maedche, A., Staab, S., Stojanovic, N., Studer, R., Sure, Y.: Semantic portal - the seal approach. In: Fensel, D., Hendler, J., Lieberman, H., W.W. (eds.) Spinning the Semantic Web, pp. 317–359. MIT Press, Cambridge, MA (2003)
8. MIND LAB: Mindswap - Maryland Information and Network Dynamics Lab Semantic Web Agents Project (2007) <http://www.mindswap.org>
9. Music Technology Group, Universitat Pompeu Fabra: Foafing the music (2007) <http://foafing-the-music.iaua.upf.edu>
10. AIFB, University of Karlsruhe: ontoworld.org (2007) <http://ontoworld.org>
11. W3C: Rdf vocabulary description language 1.0: Rdf schema (2007) <http://www.w3.org/TR/rdf-sparql-query>
12. MusicBrainz: Musicbrainz project (2007) <http://musicbrainz.org>
13. MusicMoz: Musicmoz - open music project (2007) <http://musicmoz.org/>
14. RSS-DEV Working Group: Rdf site summary (rss) 1.0 (2000) <http://web.resource.org/rss/1.0/>
15. Miller, L., Brickley, D.: Foaf project (2007) <http://www.foaf-project.org>
16. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kauffmann, Seattle, Washington, USA (2002)
17. WebModels s.r.l.: Webratio tool. (2007) <http://www.webratio.com>
18. Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A.: Creating Semantic Web contents with protege-2000. IEEE Intelligent Systems 16(2), 60–71 (2001)
19. W3C: Sparql query language for rdf (2007) <http://www.w3.org/TR/rdf-sparql-query>

20. Jena Team: Jena a semantic web framework for java (2007)
<http://jena.sourceforge.net>
21. Parsia, B., Sirin, E., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: An owl dl reasoner (Technical report)
22. Casteleyn, S., Plessers, P., Troyer, O.D.: Generating semantic annotations during the web design process. In: ICWE '06. Proceedings of the 6th international conference on Web engineering, New York, NY, USA, pp. 91–92. ACM Press, New York (2006)

Engineering Semantic-Based Interactive Multi-device Web Applications

Pieter Bellekens, Kees van der Sluijs, Lora Aroyo*, and Geert-Jan Houben**

Technische Universiteit Eindhoven
PO Box 513, NL 5600 MB, Eindhoven, The Netherlands
{p.a.e.bellekens,k.a.m.sluijs,l.m.aroyo,g.j.houben}@tue.nl

Abstract. To build high-quality personalized Web applications developers have to deal with a number of complex problems. We look at the growing class of personalized Web Applications that share three characteristic challenges. Firstly, the semantic problem of how to enable content reuse and integration. Another problem is how to move away from a sluggish static interface to a responsive dynamic one as seen in regular desktop applications. The third problem is adapting the system into a multi-device environment. For this class of personalized Web applications we look at an example application, a TV recommender called SenSee, in which we solve these problems in a metadata-driven way. We go into depth in the techniques we used to create a solution for these given problems, where we particularly look at utilizing the techniques of Web Services, Web 2.0 and the Semantic Web. Moreover, we show how these techniques can also be used to improve the core personalization functionality of the application. In this paper we present our experience with SenSee to demonstrate general engineering lessons for this type of applications.

1 Introduction

The ICT landscape is developing into a highly-interactive distributed environment in which people interact with multiple devices (e.g. portable devices such as mobile phones or home equipment such as TVs) and multiple applications (e.g. Web browsers and dedicated Web services). Globally the industry is being driven by the shift away from old models - moving from physical to digital. New methods emerge for getting content such as TV programs via the Web. For example, more and more people want to watch or manage TV content on their PCs, and thus make a bridge between a TV and a PC: each device doing what it can do best. As we can see, technologies in these fields are rapidly progressing, but the user is lost. The information overload is enormous and the content presented is hardly adapted to the prior knowledge, to the preferences and to the current situation of the user. Not only the users, but also the industry comes to

* Also affiliated with Vrije Universiteit, Amsterdam, The Netherlands.

** Also affiliated with Vrije Universiteit Brussel, Brussels, Belgium.

the realization that integrated content services and personalized user experience are becoming more important.

Personalization in information retrieval [1] and information presentation [2] has therefore become a key issue. Successful personalization experiments have been done, in e-commerce [3] and news websites [4], where the most common example is the Amazon.com recommendations. Personalization is seen as a key ingredient of the so-called Web 2.0 applications [5] [6]. However, such personalization is still local. As a nice example, on the Web users are increasingly involved in multiple virtual environments (e.g. MySpace, Flickr, YouTube, Amazon, entertainment sites) in each of them with a different identity (e.g. login information, preferences). There is very limited integration between them, and if there does exist some integration, it is not always under the user's control. Moreover, there remains a great lack of transparency in the use of personal data between different applications [7].

The main objective in this research is to provide techniques to improve personalization in the world of multi-device access to interactive Web applications connected via semantic-based integration. This world can be characterized with three key terms: *Semantic-based*, *Interactive* and *Multi-device*. We use the acronym *SIM* to refer to this application setting. In SIM Web applications, people do use multiple devices to interact with multiple distributed Web applications. It results in complex interaction patterns requiring integrated views of distributed data collections, multiple modeling perspectives of content, user and environment data, as well as an increased need for personalized information presentation. To realize this kind of personalization, we need to address challenges concerning the integration of content services, the integration of user modeling and personalized presentation. In our approach we exploit recent developments in the field of *Web services*, *Web 2.0* and *Semantic Web* to meet these challenges.

In this paper we present the core elements of this approach. In Section [2] we explain the rationale behind our focus on the SIM setting. In Section [3] we introduce the SenSee application that we use to illustrate the SIM approach. SenSee is a personalized content recommender for multimedia consumption in an ambient home media-centre and is created within the context of the Passepartout project [8]. We then discuss its architecture and subsequently in Sections [4], [5] and [6] we show how we in SenSee implemented the Web Service, Web 2.0 and Semantic Web solutions for SIM applications. In Section [7] we show how the combination of the techniques can in addition be used to aid in the core functionality of SenSee. We finally summarize our experience for the general class of SIM applications in Section [8].

2 SIM Web Application Requirements

A main concern in modern Web applications is *personalization* and adapting the application in all its facets to the user's current situation. In this research we consider the development of personalized Web applications in the light of three characteristic trends and properties of modern Web applications.

(1) One aspect that has definitely changed in the recent years is that users no longer access the Web via PCs only. The Web has become much more a multi-device environment. Many types of devices can connect to the Web nowadays: think of PDAs, mobile phones, GPS-systems, TVs, game consoles, etc. As a consequence, Web applications can no longer be engineered with only one device, with its inherent properties, in mind. Designing for using an application on different devices is one step, e.g. allowing the user to choose between PC, PDA or mobile phone, but the consequent step and the one that we are now confronted with is to consider the user's interaction with an application via the current wealth of devices in an integrated fashion. So, instead of using one device we are faced with a multi-modal context in which for example the TV-content viewer uses the PC to manage personal preferences and the mobile phone to carry these preferences around to pass them then to a TV-set for viewing the content.

The Web applications that operate in such a *multi-device* setting bring challenges as well as benefits, particularly when we consider personalized access. The main challenge is the integration of the different parts of the application, i.e. to adapt the application aspects to different environments and different capabilities. Think of differences in video, sound and data processing capabilities: when distributing the application functionality over the device-related application parts these capabilities are leading. A major benefit for personalization is that we can learn from the user (assess the user's situation) in a more unobtrusive way, since the user will spend more time with all of his devices combined than with the PC alone. Furthermore, we can utilize online sensing devices like GPS functionality through a mobile phone, RFID tags, and biometric sensors, to improve the quality and quantity of the information we have on the user and therefore improve personalization in a context-sensitive way. If we, for example, know the user is driving in a car we could infer that we don't want to bother him with video messages as his attention is needed elsewhere.

(2) In the integrated context of our applications, the increasing need for *semantic-based* reuse of information and application functionality for personalization is obvious. In terms of content, one reason is that it is time-consuming to reproduce data that is already available. Furthermore, specialist data sources often are created by domain experts or crafted collaboratively, making it hard for non-domain experts to recreate something of the same quality. Also, one might want to use external data that can not be under the direct control of the application itself and therefore has to be fetched on demand. Think, for example, of reusing information from Wikipedia¹ or IMDB². When integrating content in an application we are faced with the typical heterogeneous nature of the content, specially in the different terminology that is used from the different domains involved.

Besides reuse of content, also reuse of functionality becomes an even more relevant issue. Being able to share and combine application logic (components) in the integrated application enables to configure the application and its personalization

¹ <http://wikipedia.org/>

² <http://imdb.com>

on demand. The applications we consider do not just “import” content and functionality, but also “export” aspects to other applications. Most prominently, one can see this for the sharing and exchange of the user model information that is used in the personalization process. Applications running on different devices that are able to share the user’s preferences and situation description can improve their (individual) service to the user.

(3) A third trend in Web applications is the ever growing demand for *richer user interaction*. This interactivity can be used for personalization to offer the user more control, for example by allowing richer and more detailed feedback on metadata, both content-metadata and user model data. Modern Web applications should also no longer suffer from an unresponsive user interface between page loads, e.g. because of complex time-consuming operations. Instead, they should move towards the responsiveness of dynamic desktop applications. If a user triggers a query that gets distributed over different sources on the Web, the user should not need to wait until the slowest source responds and sends all its data. It would be much better to show as quickly as possible the first results, integrating results of slower sources as they arrive.

Enabling personalization in the next generation of these SIM Web applications where *semantics*, *interactivity* and *multi-device* are important properties, various new and more elaborate requirements surface. Tackling these challenges in a natural unobtrusive way is key for the success of such personalized applications. In the next section we introduce SenSee that helps to illustrate our target class of applications.

3 SenSee

SenSee is a personalized Web-based recommender for digital television content and an example of a SIM application that we use to illustrate our case. It is developed in the context of the Passepartout [8] project, which investigates the future of digital television. SenSee collects information about TV programs as well as related data from various sources on the Web. Furthermore, the user can access his personalized TV guide from various devices at any time, so not only on the TV itself. To provide high-quality personalization SenSee collects as much data on the user as possible. It provides the user with direct feedback options to provide his information manually, but also observes the user’s behavior and with help of sensor information like GPS determines the user context.

Many related TV recommender systems can be found in literature, like in [9], [10] and [11]. However, those systems mainly focus on the recommendation part. In AVATAR [10], for example, the authors make use of a combination of TV-Anytime [12] metadata fields and a custom-made genre hierarchy. Their recommendation algorithm is an effective hybrid combination of content-based and collaborative filtering, although there is no inclusion of any kind of user context. To get an overview of the various kinds of recommendation strategies and combinations that exist both for selection, structuring and presentation of content refer to [11]. In this paper, however, we focus on the properties of the

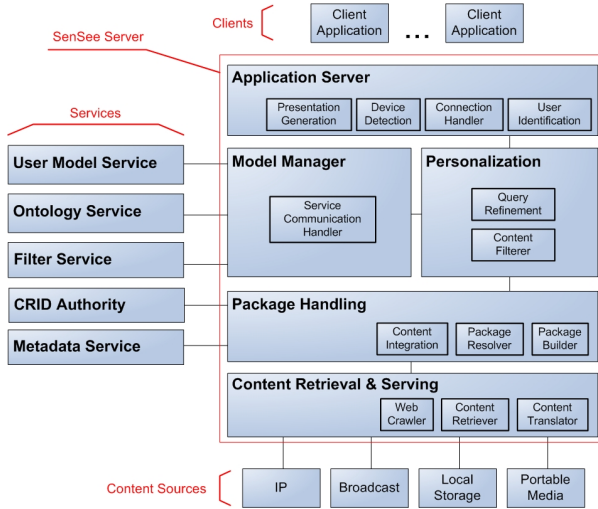


Fig. 1. SenSee Architecture

SIM framework which serves as the backbone of the SenSee recommender and how we can utilize the content metadata together with the user model metadata to solve the issues at hand.

The SenSee architecture, as depicted in Figure 1, is a layered one. At the bottom part of the figure, we find the various heterogeneous content sources. This can include regular TV broadcasts, but also movie clips from the Web, media stored on the local computer or physical media like DVD and Blu-ray discs. Metadata about the content originates from the bottom, propagates up while at each layer gaining semantics and context, and is finally used by an application at the top. In this process first information about the content is collected, and then converted to our internally used vocabulary and aggregated into *packages*. In order to do so we use two external services that are specified by the TV-Anytime specification, namely the *CRID Authority* for uniquely identifying multimedia objects and the *Metadata Service* that maintains metadata on these objects. Since typically there are numerous content packages available, we use personalization and recommendation based on the context and the user model with the aim to prevent that the user gets lost in the information abundance. Furthermore, we offer the user a user-guided search to assist the user in finding what he or she is looking for by refining a user query (keywords) with help of ontological domain knowledge. Both for personalization and user-guided search we use supporting services, each with their own responsibility and functionality. The *User Model Service* (UMS) maintains and retrieves data from the user model (including the current user context), the *Filter Service* (FS) provides filters needed to eliminate content unsuitable for the current user, and the *Ontology Service* (OS) maintains and manages all vocabularies and ontologies defining the

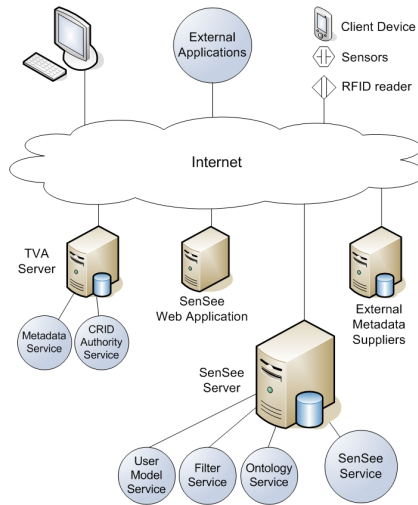


Fig. 2. SenSee Connection Architecture

semantics of concepts and properties. Finally, the metadata is shown to the user in a hierarchically structured way: the user can browse through the data and select the desired multimedia object. The object can then be retrieved via its unique CRID identifier [13] for consumption (viewing).

Figure 2 shows how the different parts of the SenSee framework are connected. The user can access SenSee via a client application. We currently implemented two client applications: The SenSee Web application [3] and a commercial application called iFanzzy [14] that runs on a set-top box and is controlled by the television remote control. In this paper we will concentrate on the SenSee Web application which runs in a Web-browser [4].

The SenSee Web application connects to the SenSee server, which besides the SenSee Service (which contains the main application logic) also contains the three supporting services: UMS, FS and OS. The TV-Anytime services, CRID Authority Service and Metadata Service, as well as the content metadata that is not stored locally, are accessible via the Internet. Sensors and device information are accessed via external applications which are responsible to reading the sensor values and adding or updating them in the user model. Having SenSee running as a service enables these various applications to quickly make a connection and push their readings to the system. SenSee on its turn processes these readings and can immediately take them into account in upcoming actions.

For mass multimedia content consumption, people prefer the television as main target device. Since SenSee targets users from all kinds of social classes and age groups these sensors could make *interactivity* easier. Because of the personal nature of this application, people are required to log in, either by login/password

³ <http://wwwis.win.tue.nl:8888/SenSee/>

⁴ Note: Currently only tested in Firefox and Internet Explorer.

or through an RFID sensor. RFID recognition has been implemented to login people automatically when they for example enter the living room. Knowing who is watching, allows SenSee to optimize recommendations, the look-and-feel, and privileges that are appropriate for the current user or user group.

To personalize access to various online content sources and services, a certain sense of *semantics* is required for interpretation of the metadata. The domain model used by SenSee, to discern between for example ‘actor’ and ‘director’ or a ‘western’- and ‘action’-movie, is specified in the TV-Anytime specification [12]. TV-Anytime was specifically tailored to describe future multimedia content and provides constructs to describe all kinds of content elements as well as hierarchical containers suited to cluster and organize these numerous elements. To be able to exploit external content, metadata is required for interpretation. However, using various heterogeneous content sources implies that available metadata might be complying with different conceptual schemas which in turn might be not compatible with the TV-Anytime specification. In such a case SenSee uses transformations to migrate this content to the TV-Anytime platform. Currently, SenSee retrieves content from online Electronic Program Guides⁵, BBC backstage⁶ and various Web sites like IMDB and Wikipedia.

Since TV-Anytime focuses on describing multimedia content, which gives a rather restricted perspective, other concepts related to, for example, time and geographical locations are less profound. Therefore, in SenSee we fill those gaps by adding external knowledge coming from publicly available ontologies, schemas and thesauri by relating their concepts to existing ones in TV-Anytime, giving the domain more profundity. The most important knowledge structures, which are all maintained in the Ontology Service (OS), are among others:

- The TV-Anytime and MPEG7⁷ schemas to describe content
- The W3C OWL-Time ontology [15] providing time conceptualizations
- A geographic thesaurus to describe locations like e.g. [16]
- WordNet^{8,9} to provide synonyms, hyponyms and other lexical relations
- Topic hierarchies to identify e.g. program genres (TV-Anytime provides several topic classifications among which are content and action classification)

Maintaining the various RDF [17]/OWL [18] repositories we chose for the metadata, such as the user models and the various contexts in the UMS, and the various ontologies and thesauruses in the OS, is done through a Sesame [19] triple repository.

4 Web-Service Architecture

The SenSee application *shares* all of its main parts to encourage interoperability with others. Web services enable the interconnection between connected devices,

⁵ <http://xmltv.org/wiki/>

⁶ <http://backstage.bbc.co.uk/>

⁷ <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>

⁸ <http://wordnet.princeton.edu/>

⁹ <http://www.semanticweb.org/library/>

allow third parties to adapt their existing software to the provided interfaces and enable lightweight sensor devices to provide input.

Communication between all services and clients is established via XML-RPC¹⁰. XML-RPC is a remote procedure call protocol making use of XML to encode its messages and HTTP as transportation layer. XML-RPC was chosen because of its simplicity. It is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned. The implementation we use is version 3 of Apache XML-RPC¹¹. The main advantage of this implementation is its ability to transmit any Java object, taken that it is serializable, and its ability to handle exceptions with ease. SOAP¹² was also considered, but discarded again because of the higher complexity in use, while the functionality was the same. To handle the various incoming XML-RPC requests and have them executed in such a way that different calls do not interfere with each other while maintaining good performance, we chose to use the Jetty Web server¹³. Jetty is a highly customizable standalone Web server completely build in Java, known for its scalability and efficiency.

Via this universal communication protocol we enable multiple devices, understanding the HTTP protocol, to be able to contact the SenSee server. Choosing lightweight XML-RPC messages to build and receive requests enables us to even use devices with limited processor power and bandwidth, like for example mobile phones. Furthermore, working with services allows us to run multiple instances of a specific service at the same time in different locations. The advantage of such a setting is that we enable load balancing, to avoid that one particular server chokes under too much requests. The disadvantage of such an approach is that we then also need a synchronization algorithm to keep all service instances up-to-date.

5 Interface Asynchronicity and Ajax

A SIM application like SenSee uses data and application logic of a fundamentally distributed nature, so we have to solve distribution-related problems. One of the main problems is that we cannot rely on those sources. Data and applications can be taken off-line without notice or might be very badly reachable because of network problems. However, what we want to minimize is that the user has to wait indefinitely because of one unavailable or slow source. This problem in the user interface can be tackled by making use of parallelism. To be able to use parallelism in a Web interface setting we used the asynchronous Ajax technology. In this way the user can see results that arrive quickly immediately, while data that arrives later can be easily blended in within the current user interface, without reloading the whole page. If the user clicks ‘Submit’ with the keyword query ‘bike’, different ontologies are checked for possible conceptualizations for

¹⁰ <http://www.xmlrpc.com/>

¹¹ <http://ws.apache.org/xmlrpc/>

¹² <http://www.w3.org/TR/soap12-part0/>

¹³ <http://jetty.mortbay.org/>

this keyword. Due to the asynchronicity in the interface, we can check various ontologies at the same time while results arrive in a undetermined order.

For the implementation we used the Google Web Toolkit¹⁴ (GWT). GWT provides a Java API which enables programming in Java, while afterward the GWT compiles the classes and dynamically converts them to regular HTML and Javascript that handles both the user interface and the asynchronous XML-RPC calls.

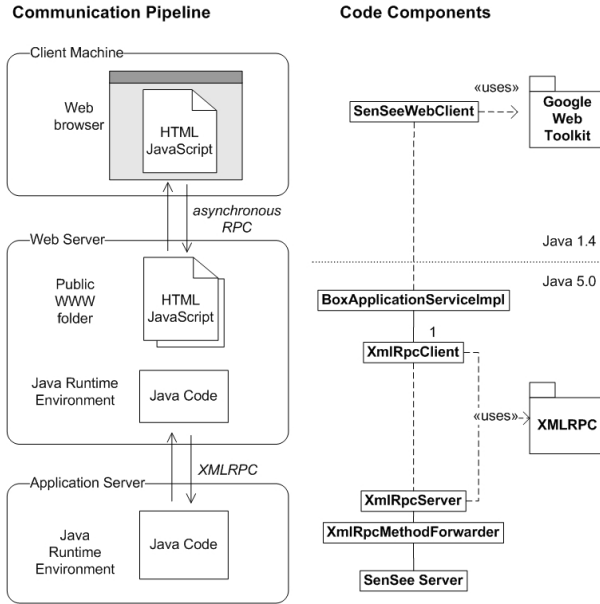


Fig. 3. SenSee communication pipeline

In Figure 3, we see the communication pipeline from a client Web application to the SenSee server. The Web browser initiates the communication by loading a Webpage from the Web server. Via asynchronous RPC calls Java methods are executed on the Java back-end of the Web server. Each one of those method-calls themselves, again forward an XML-RPC call to the appropriate service (the SenSee server, UMS, OS or FS) running on an application server.

6 Semantic-Based Content Integration

When a user fires a request, SenSee responds with an integrated set of content containing pieces from various sources. Like previously mentioned, SenSee retrieves content and metadata descriptions from these various sources which comply with

¹⁴ <http://code.google.com/webtoolkit/>

different data schemes. However, since SenSee is completely built around the TV-Anytime scheme, we chose to transform all incoming content to TV-Anytime. The TV-Anytime specification consists of two parts, “phases” in TV-Anytime speak. The first phase consists of a synchronized set of specification documents for meta-data, content referencing, rights management, and content protection. The second phase defines open standards that build on the foundations of the first phase and includes areas such as targeting, redistribution and new content types. A central concept in the second phase is packaging. A package is a structure which contains a set of unambiguous identifiers referring to content-elements which are somehow related. An example would be a ‘Lord of The Rings’ package containing all the movies, Making-Ofs, the text of the books, images, the soundtracks, etc. To be able to reuse this generated content we add a sense of context to each package from which we can benefit in future requests.

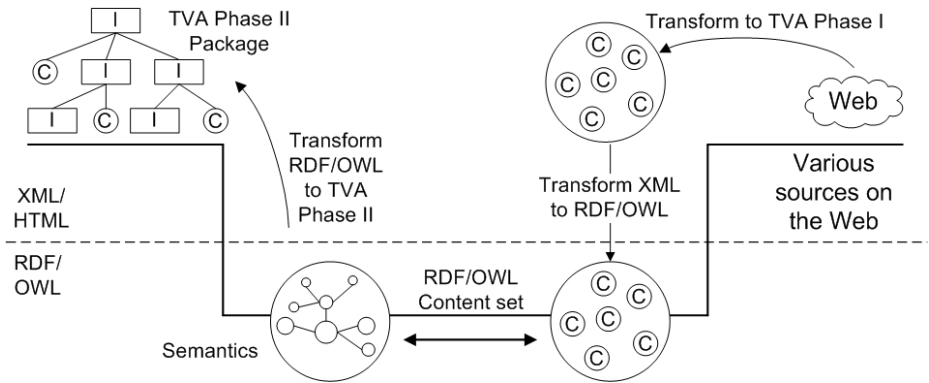


Fig. 4. SenSee data integration scheme

In Figure 4 we see the complete SenSee data integration scheme. In this figure we see that there are two layers visible, namely XML/HTML at the top and RDF/OWL at the bottom of the figure, depicting a clear separation between the two. On the right hand side we see our various sources located on the Web, these sources are typically retrieved in HTML (Wikipedia, IMDB, ...) or XML (XML-TV and BBC Backstage content) formats. This content is then transformed to TV-Anytime Phase I (XML). However, in order to apply our semantics available in the OS which are typically modelled in RDF/OWL, we need to transform the TV-Anytime XML (Phase I). This second transformation crosses the dashed line bringing the TV-Anytime Phase I content set into the RDF/OWL layer. All content here is stored in a Sesame RDF store which serves as a temporal cache to augment and work with the retrieved data. The advantage of our approach of using RDF is that the resulting integrated datamodel can be easily adapted for plugging in and removing data sources and relating those to supporting ontologies because of the open world nature of RDF. In this way we can use

inference techniques (e.g. subsumption) to get more useful data for the user and we can utilize mechanisms like the context feature in Sesame 2.0¹⁵ to track the origin of information.

Once here, we can apply the semantics stored in the OS to enrich the current content set. For example, every TV-Anytime Phase I content set always has a set of keywords to describe this content in its metadata. The TV-Anytime (TVA) field for this is:

```
<element name="Keyword" type="tva:KeywordType" minOccurs="0" maxOccurs="unbounded"/>
```

We can make a simple enrichment by searching for synonyms of these keywords, in the WordNet dictionary, and expand the existing set. Similarly, we can also add semantics to time specifications. In TVA every time-related field is modeled by an MPEG-7 ‘timePointType’, where a time point is expressed with the regular expression:

```
'-'?yyyy'-'?mm'-'?dd'T'hh':'?mm':'?ss(''.')s+)?(zzzzz)?
```

The date ‘2007-01-01T12:30:05’ is an example. By mapping this string to a time description in our Time ontology we can obtain the following time specification (do note the abbreviated syntax):

```
<CalendarClockDescription rdf:ID="example">
  <second>05</second>
  <hour>12</hour>
  <minute>30</minute>
  <year>2007</year>
  <month>01</month>
  <week>01</week>
  <day>01</day>
</CalendarClockDescription>
```

This richer time description enables us for example to easily cluster all content produced in a particular year or all content being broadcasted before or after a certain point in time.

After this enrichment phase, our content is ready to be used to create a TVA Phase II package for a particular user query. The retrieval of the content was triggered by a user query and the package will now be the (rich) answer to this query. A package is constructed hierarchically and consists out of items, nodes in the hierarchy to enable browsing and navigation, and components which contain the effective content like a movie, a piece of text, an audio file or any other multimedia type.

The simplest package we can generate in SenSee is a package with one item that contains one component, which is effectively an arrangement of the TVA Phase I metadata. However, we are also looking into experimenting with more advanced automatic clustering. We try to cluster content based on relationships of its metadata with helper ontologies, e.g. we relate the TVA Phase I metadata to ontology concepts with string matching techniques. Like this we can make use of the fact that ontologies are well-crafted by domain experts, and therefore

¹⁵ <http://www.openrdf.org/doc/sesame2/users/ch04.html#d0e836>

the content grouping will be well-structured. Sorting the grouping structure can be done by applying traditional recommendation techniques, e.g. by taking the user model into account.

7 Application in User-Guided Search Support

In the previous sections we demonstrated solutions for the SIM issues. However, the described techniques could also be used to improve the core personalization functionality of the application. Besides offering ‘passive’ recommendations, the user can also actively search through the available content. Our approach is based on semantic faceted browsing techniques (e.g. refer to [20] or [21]). The idea is to go beyond a pure keyword-based search. Instead the user is assisted to search through different *facets* of the data. To accomplish this, additional metadata fields as defined in TV-Anytime specification were used. Moreover, like previously mentioned, ontological sources to make a further semantical connection between the terms in TV-Anytime play a important role. Consider for instance the time facet, TV-Anytime uses the XML datetime datatype to express the time when a program is broadcasted. To semantically enrich time information we use the W3C OWL-Time ontology, adding for instance that the concept ‘afternoon’ ranges from 1400h-1800h. In this way we can easily search for programs in the afternoon by looking for all programs broadcast within this 1400h-1800h interval.

By incorporating all these results, as is shown in the screenshot in Figure 5, we try to close the gap between the keywords the user types in and what the user intends to search for. As an example, consider a user making the following keyword request: “sports Scotland ‘Friday evening’” when he looks for Scottish sports games to watch during Friday evening when his friends come over. The system reacts by trying to find any ontological matches for these keywords, through the use of the OS. In Figure 5 the results are shown. Under the tab ‘Genres’ we see that the system found some matches for the keyword ‘sport’ in the genre classification and in the Geo ontology a match was found for ‘Scotland’. In the time ontology a match was found for ‘Friday evening’ and automatically translated to an interval from 18pm until 23pm as shown in the calender under the ‘Time’ tab. With this system-assistance the user is able to manually refine the original request. With the genre hierarchy it is possible now to specify a more particular sport type the user might be looking for, or just select the broad term ‘sports’. In the geographical hierarchy either a narrower term, like a region in Scotland, or a broader term like ‘United Kingdom’ can be chosen. Via the calender the user can revise his time interval.

After the system-assistance, the newly refined query is taken as input for the retrieval process which tries to find relevant content made available by the various sources. When the sources start responding with multimedia content, the retrieved content needs to be filtered and personalized by leaving out items deemed unsuited, and ranking important item higher up the results list. This process, which forms the hart of the personalization process, uses mainly the UMS to find the user’s preferences, the FS to provide the appropriate filter, and

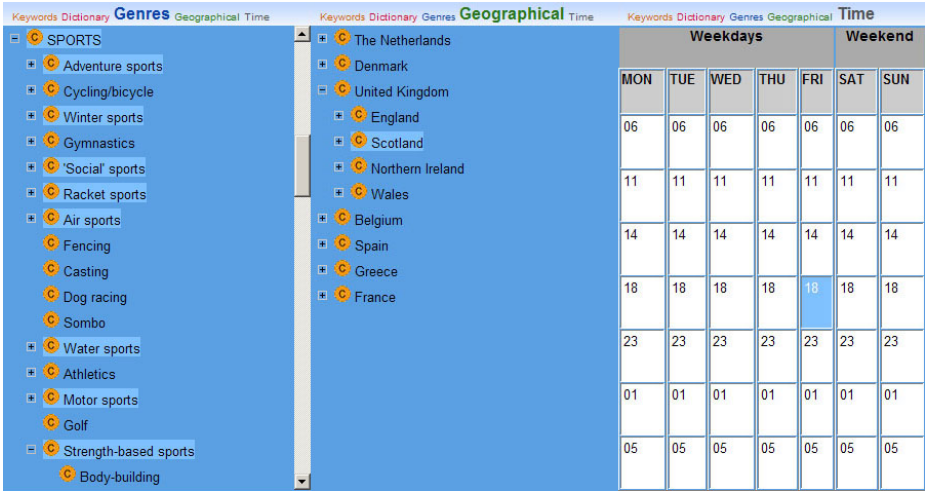


Fig. 5. User-driven concept refinement

the OS to give correct semantics throughout the complete sequence. The filter provided by the FS basically acts as the glue between values in the user model and fields in the content’s metadata. It decides for every content-element whether or not it should be discarded after comparing its metadata to the interests and preferences of the user. Lastly, the resulting content set needs to be presented to the user, by means of the TV-Anytime packaging concept. Packages allow us to create for example a ‘Marlon Brando’ package were everything known from this person can be bundled and browsed. Packages can contain all kinds of content-elements ranging from pure text to full HD (High Definition) movie clips. Like this, the results are clustered and hierarchically structured to obtain an easy-to-browse content-container where the user finds what he or she was looking for or at least is interested in.

8 Conclusion

We have considered a growing class of Web applications that share three characteristic demands that are typical for these emerging applications: semantics for integration, richer interactivity and multi-device adaptation. We have taken an illustrative example of such an application, a TV recommender called SenSee, that tackles these problems with a metadata-driven approach. The main technologies that have been used in the realization of SenSee were techniques from Web Services, Web 2.0 (more specific Ajax technology) and Semantic Web. With Web Services we were able to flexibly connect functional blocks together, e.g. for dynamically choosing the appropriate modality for the appropriate devices. Furthermore, we saw that our approach allowed plugging-in new functionality, but also makes our functionality available to others. Web 2.0 technology like

Ajax was used to improve the user experience. By applying asynchronicity we could accommodate external content sources that may be slow or unavailable, while at the same time the application was staying as responsive as a regular desktop application. Semantic Web technologies were exploited to relate concepts from various heterogeneous sources to the used TV-Anytime specification on which SenSee operates. We also showed that we had an extra benefit as we could use these same semantic techniques to improve the core business of the recommender: by dynamically plugging-in extra knowledge in form of domain ontologies we could give the user more control over the process of finding what they really wanted.

The experience gained in developing SenSee was achieved within the Passepartout project where multiple partners were working closely together. In the project other applications were developed by companies such as Philips Research¹⁶, Stoneroos¹⁷, V2¹⁸, Henri Tudor¹⁹ and CWI²⁰, and we observed that our solution proved to be valuable as framework to use, and that SenSee could be extended with new third-party algorithms and add-ons, improving interoperability. The service-based architecture helped us in many ways facilitating these cooperations. We learned how the growing class of Web applications with SIM characteristics can be built. Although there might also be other ways to accomplish this, the semantic-based, metadata-driven approach we took allows the application of technologies that can get the job effectively done.

References

1. Allan, J., B.C.(eds.): Challenges in information retrieval and language modeling. SIGIR Forum 37(1), pp. 31–47 (2003)
2. Brafman, R.I., Domshlak, C., Shimony, S.E.: Qualitative decision making in adaptive presentation of structured information. ACM Trans. Inf. Syst. 22(4), 503–539 (2004)
3. Ardissono, L., Goy, A.: Tailoring the interaction with users in web stores. User Modeling and User-Adapted Interaction 10(4), 251–303 (2000)
4. Ardissono, L., Console, L., Torre, I.: An adaptive system for the personalized access to news. AI Communications 14(3), 129–147 (2001)
5. Webster, D., Huang, W., Mundy, D., Warren, P.: Context-orientated news filtering for web 2.0 and beyond. In: WWW '06. Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, pp. 1001–1002. ACM Press, New York (2006)
6. Millard, D.E., Ross, M.: Web 2.0: hypertext by any other name? In: HYPERTEXT '06. Proceedings of the seventeenth conference on Hypertext and hypermedia, New York, NY, USA, pp. 27–30. ACM Press, New York (2006)

¹⁶ <http://www.research.philips.com/>

¹⁷ <http://www.stoneroos.nl/>

¹⁸ <http://www.v2.nl/>

¹⁹ <http://www.tudor.lu/>

²⁰ <http://www.cwi.nl/>

7. Jones, M.B.: The identity metasytem: A user-centric, inclusive web authentication solution. In: *Toward a More Secure Web - W3C Workshop on Transparency and Usability of Web Authentication* (2006)
8. Passepartout: Itea passepartout project. (2005-2007) <http://wwwis.win.tue.nl/~ppartout>
9. Ardissono, L., Kobsa, A., Maybury, M.T. (eds.): *Personalized Digital Television. Human-Computer Interaction Series*, vol. 6. Springer, Heidelberg (2004)
10. Blanco Fernández, Y., Pazos Arias, J.J., Gil Solla, A., Ramos Cabrer, M., López Nores, M.: Bringing together content-based methods, collaborative filtering and semantic inference to improve personalized tv. *4th European Conference on Interactive Television (EuroITV 2006)* (May 2006)
11. van Setten, M.: Supporting people in finding information: Hybrid recommender systems and goal-based structuring. *Telematica Instituut Fundamental Research Series*, No.016 (TI/FRS/016). Universal Press (2005)
12. TV-Anytime. The TV-Anytime Forum, Requirement Series: RQ001v2.0, TV140. (April 2003) Available at: <ftp://tva:tva@ftp.bbc.co.uk/pub/Plenary/0-Plenary.html>
13. Earnshaw, N.: The tv-anytime content reference identifier (crid) (2005)
14. Akkermans, P., Aroyo, L., Bellekens, P.: European Semantic Web Conference 2006 (demo presentation) (2006), <http://www.eswc2006.org/demo-papers/FD36-Lora.pdf>
15. Hobbs, J.R., Pan, F.: An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)* 3(1), 66–85 (2004)
16. Chipman, A., Goodell, J., Johnson, R., Ward, J.: Getty thesaurus of geographic names: editorial guidelines (2005) http://www.getty.edu/research/conducting_research/vocabularies/guidelines/tgn_1_contents_intro.pdf
17. Manola, F., Miller, E.: Resource Description Framework (RDF). <http://www.w3.org/TR/rdf-primer/>
18. McGuinness, D.L., van Harmelen, F.: OWL web ontology language. <http://www.w3c.org/TR/owl-features>
19. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema (2002)
20. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: *CHI '03. Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, pp. 401–408. ACM Press, New York (2003)
21. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) *ISWC 2006. LNCS*, vol. 4273, pp. 272–285. Springer, Heidelberg (2006)

Towards Improving Web Search by Utilizing Social Bookmarks

Yusuke Yanbe, Adam Jatowt, Satoshi Nakamura, and Katsumi Tanaka

Department of Social Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, 606-8501
Kyoto, Japan

{yanbe, adam, nakamura, tanaka}@dl.kuis.kyoto-u.ac.jp

Abstract. Social bookmarking services have become recently popular in the Web. Along with the rapid increase in the amount of social bookmarks, future applications could leverage this data for enhancing search in the Web. This paper investigates the possibility and potential benefits of a hybrid page ranking approach that would combine the ranking criteria of PageRank with the one based on social bookmarks in order to improve the search in the Web. We demonstrate and discuss the results of analytical study made in order to compare both popularity estimates. In addition, we propose a simple hybrid search method that combines both ranking metrics and we show some preliminary experiments using this approach. We hope that this study will shed new light on the character of data in social bookmarking systems and foster development of new, effective search applications for the Web.

Keywords: Web search, social bookmarks, PageRank, meta-search.

1 Introduction

In the early years of the Web, directory services were utilized in order to arrange the Web and to make it accessible to users. However, the rapid growth of the Web soon made this approach impractical. Computing page relevance was also insufficient since usually too many pages were relevant to user queries. In order to effectively rank pages the quality of Web documents had to be captured. Thus, came the era of link based algorithms such as PageRank [18] and HITS [10], which estimate quality of pages by measuring their relative popularity in the Web. PageRank is currently the most popular link-based Web page ranking method. It is based on a random surfer model, where the probability of the surfer reaching a given page is calculated as the result of a random selection of links. Consequently, the popularity of the page is determined on the basis of the size of its hypothetical user stream.

Link-based page popularity estimation has, however, several disadvantages. One is related to the difficulty of creating links as it usually requires some effort and knowledge from users. Although, recently we observe the explosion of Weblogs or wikis, which make the link creation relatively easy, yet search engines seem not to trust links on such pages due to spamming threats. In general, links on majority of

pages are still created by a relatively small group of content producers. However, there is an overwhelming group of content consumers whose opinions cannot be captured by standard link-based ranking metrics. Additionally, links often serve many different purposes on Web pages and, hence, should not always be treated as positive votes for pages [14].

Another disadvantage of link based ranking mechanisms is related to their temporal aspect. Web is a very dynamic environment and many new pages are continuously created (see for example, [7,17]). However, pages usually need long time to acquire links and to become popular among Web authors. In result, PageRank algorithm is biased against new pages [2,12,23]. Considering the fact that users usually need fresh information, this bias is a major disadvantage of link-based algorithms making them weak in providing fresh content or detecting new, hot topics and trends in the Web.

In general, the conventional link-based ranking approach is still useful, mostly due to its success in combating spamming, however, we believe that it needs to be complemented by another reliable metric. Along with the advent of Web 2.0, social bookmarking systems seem currently to have a potential for improving the capabilities of existing search engines. Social bookmarking lets users share, classify, and discover interesting Web pages. In social bookmarking systems, the popularity of a Web page is usually calculated by the total number of times it has been bookmarked by users. We call this measure SBRank. As creating bookmarks is relatively easy and does not require much technological knowledge, thus, in contrast to links, any Web user can freely vote for pages. This, together with the high level of social interaction in social bookmarking services, makes SBRank a highly dynamic measure allowing for detecting high-quality, fresh and hot information on the Web.

Although social bookmarks have many advantages, relying on them alone is still not advisable in a general purpose Web search. This is because there is currently not enough data to produce satisfactory results for any arbitrary query¹. Although, recently, we are observing a rapid increase in the number of bookmarked pages, yet we believe that the combination of link structure and social bookmarking based popularity estimates seems to be currently an optimal strategy. Future search applications should have at least the scalability of the existing search engines combined with improved ranking models.

In this paper, we attempt to make a comparative analysis between PageRank and SBRank metrics. The objective of this investigation is to analyze the feasibility and potential of a hybrid search method that would combine both popularity measures. In order to do so, we examine pages in social bookmarking systems and analyze their popularity using SBRank and PageRank measures. We also investigate the dynamics of SBRank metric in order to analyze whether it can improve freshness of search results.

More socially-aware search algorithms that would leverage the content of so-called Web 2.0 are an attractive vision as users often want to find information that is socially accepted (recommended by many users) and also recently popular. However, conventional link-based ranking methods cannot completely fulfill such requirements.

¹ Meta-search applications that would increase the amount of data by collecting evidences from different social bookmarking services have not appeared yet.

This work attempts at laying foundations towards building Web search applications that would exploit social bookmarks. We believe that our analysis and other similar systematic studies are necessary for designing reliable and high-quality Web search applications.

Previous studies of social bookmarking in the Web focused mostly on its social and linguistic aspects [6,15,16,19,20,21,22]. For example, the phenomenon of folksonomy (i.e. community-evolved taxonomy) was analyzed [16,19,21,22], tagging dynamics was examined [6] or a taxonomy of the current social bookmarking services was proposed [15]. The aim of our investigation is, however, different from these works, and has a practical objective, that is, examining the possibility and potential benefits of complementing traditional Web search with social bookmarking data.

The rest of this paper is organized as follows. Section 2 discusses the related research. Section 3 demonstrates the results of the analysis that we made. Next, Section 4 summarizes our findings and discusses the issues involved with building Web search applications that would utilize social bookmarks. Lastly, Section 5 concludes the paper and provides a brief look at our future work.

2 Related Work

The origins of social bookmarking date back to the work of Keller et al. [9] who in 1997 proposed to enhance Web browsers' bookmarking capabilities by using collaborative approach. Later, Bry and Wagner [3] also conducted a similar research. In the end of 2003, Joshua Schachter launched the first social bookmarking service called *del.icio.us*². Later, many kinds of social bookmarking systems have been established and, currently, we are witnessing a rapid increase in their popularity.

Although, already some investigations have been made [6,15,16,19,20,21,22], social bookmarking is still a relatively new phenomenon that has not been studied well. Studies that have been made so far focused mostly on the issues related to folksonomy and social aspects. For example, Zhang et al. [22] introduced a hierarchical concept model of folksonomies using HACM - a hierarchy-clustering model. The authors reported that certain kinds of hierarchical and conceptual relations exist between tags. In another work, Golder and Huberman [6] measured regularities in user activities, tag frequencies, and bursts in popularity of tags used in social bookmarks. The authors discussed also dynamics of tagging exhibited in social bookmarking. In addition, tags were classified into seven categories depending on the functions they perform for bookmarks. More recently, Marlow et al. [15] introduced the taxonomy of tagging systems to illustrate their potential benefits. In another work, Wu et al. proposed a search model for annotated Web resources using social bookmarks as an example [20]. Nevertheless, none of the previous studies made comparative analysis of link- and social bookmark-based page ranking methods for the purpose of their combination.

Recently, several researches have been done on temporal link analysis [1,2,4]. Temporal link analysis focuses on link evolution, discovering link change patterns or on utilizing link timestamps for improving page ranking. For example, Amitay et al.

² <http://del.icio.us/>

proposed a method for finding authority pages in time as well as for detecting trends in the Web by using link timestamps [1]. Baeza-Yates et al. [2] suggested modifying PageRank by incorporating last-modification dates of pages. The objective was to eliminate the bias of PageRank towards old pages [2,12,23]. In another paper, Cho et al. [4] proposed a quality model of pages based on the changes in the amount of in-bound links of pages in time. According to this model, pages with growing popularity trends, measured by large increases of in-bound link numbers, should have highest qualities assigned, especially, if they are still relatively unpopular in the Web. On the other hand, Yu et al [23] proposed an algorithm called Timed PageRank for incorporating link duration into page ranking process by exponentially decaying PageRank scores of linking pages. However, the approaches that use links dynamics are rather impractical as it is usually difficult to determine link creation dates. In contrast, social bookmarks usually contain timestamps indicating dates of their creation. Thus, unlike in the case of link-based ranking, incorporating temporal aspects into the Web search seems to be generally more feasible by using social bookmarks.

Lastly, meta-search engines [5,11,13] are also related to our work. Several meta-search engines have been recently employed on the Web. They provide the advantage of the increased coverage of the Web as well as more up-to-date results due to drawing data from multiple search engines. No approach has been, however, proposed so far to combine the information derived from link structure and social bookmarks for enabling a joint page ranking metric. This is probably due to different characteristics of both information sources and the lack of their comparative analysis. In this paper, we attempt to fill in this gap.

3 Comparative Analysis

3.1 Dataset Characteristics

To analyze characteristics of pages in social bookmarking services we collected two datasets. As a source of the first dataset we selected del.icio.us since it is currently the most popular social bookmarking service³ and it was also used by other researchers for studying social bookmarking [6,22]. Second dataset was created using Hatena Bookmark⁴ – the most popular bookmarking service⁵ in Japan, which was available online since February 2005.

Both datasets were obtained in the following way. We have utilized *popular tags*, which are sets of the most popular and recently used tags. Such tags are continuously published by del.icio.us⁶ and Hatena Bookmark⁷. In total, 140 tags were retrieved on December 6th, 2006 from del.icio.us and 742 tags on February 16th, 2007 from Hatena Bookmark. Next, we collected popular URLs from these tags. Usually less

³ In September 2006 it was reported that the service had 1 million registered users:

<http://blog.del.icio.us/blog/2006/09/million.html>

⁴ <http://b.hatena.ne.jp>

⁵ The service had 60,000 users in October 2006: <http://d.hatena.ne.jp/naoya/20061020>

⁶ <http://del.icio.us/tag>

⁷ <http://b.hatena.ne.jp/t>

than 25 popular pages were listed for each tag in both social bookmarking systems. At this stage, we obtained 2,673 pages for del.icio.us and 18,377 pages for Hatena Bookmark. In the last step, we removed duplicate URLs (i.e. URLs listed under several popular tags). Finally, we obtained 1,290 and 8,029 unique URLs for del.icio.us and for Hatena Bookmark, respectively. Each URL had two attributes: firstDate and SBRank. firstDate indicates the time point when a page was introduced to the social bookmarking system for the first time by being bookmarked by one of its users. SBRank, as mentioned above, is the number of bookmarks of a given page obtained at the date of the dataset creation.

In order to detect PageRank values of the URLs, we used Google Toolbar⁸ which is a browser toolbar that allows viewing PageRank values of visited pages⁹. PageRank values obtained in this way are approximated on the scale from 0 to 10 (0 means the lowest PageRank value of a page).

To sum up, the obtained datasets are snapshots of the collections of popular pages in both social bookmarking systems. Each page has its Pagerank and SBRank values recorded which it had at the time of the dataset creation.

3.2 Distribution of PageRank and SBRank

Figures 1a and 1b show the percentage distribution of PageRank values in both datasets. We found that more than a half of pages (56.1%) have PageRank values equal to 0 in the del.icio.us dataset. There are even more such pages in Hatena Bookmark dataset (81%); probably due to its more local scope. These pages are rather unpopular according to the link-based ranking and are relatively difficult to be found using conventional search engines. However, many social bookmarkers considered them to be of high quality and bookmarked them in the systems. It may imply that the pages were discovered by users from other sources than conventional Web search engines. Possibly this could happen by interacting with the social bookmarking systems, since unlike bookmarks on a personal Web browser, social bookmarks affect users socially. For example, del.icio.us informs users about popular pages that recently obtained relatively many bookmarks¹⁰. Users can also subscribe to “Inbox” - a bookmark activity reporting service. From this feedback, pages attracting much attention can become rapidly known to many users.

In general, we think that there may be two possible reasons that caused the occurrence of many pages with low PageRank values in the datasets, despite their high popularity among social bookmarkers.

- The pages were created recently, thus, on average, they have relatively few inbound links.
- The pages were created long time ago but their quality cannot be reliably estimated using PageRank measure.

In order to determine which of these two reasons is more probable we did temporal analysis of the pages, which we will discuss in Section 3.4.

⁸ <http://toolbar.google.com>

⁹ We had to use Google Toolbar since Google API does not provide any automatic method for acquiring PageRank scores.

¹⁰ <http://del.icio.us/popular/>

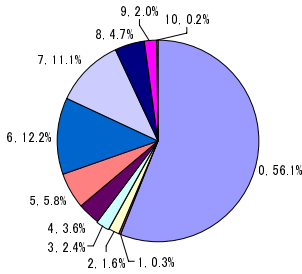


Fig. 1a. Distribution of PageRank values (del.icio.us)

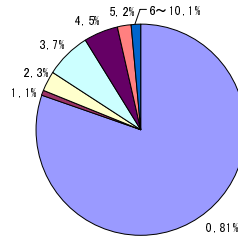


Fig. 1b. Distribution of PageRank values (Hatena Bookmark)

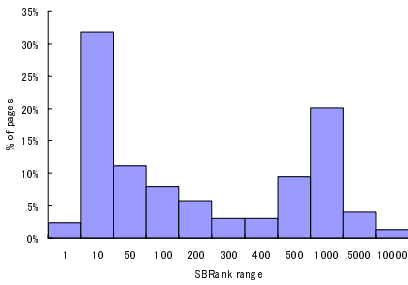


Fig. 2a. Histogram of SBRank (del.icio.us)

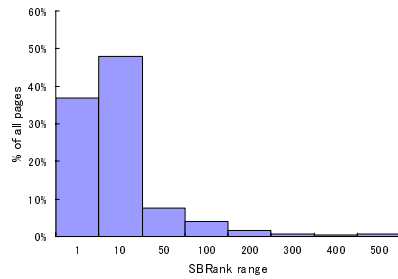


Fig. 2b. Histogram of SBRank (Hatena Bookmark)

Figures 2a and 2b show the distributions of SBRank values. It can be seen that quite few pages are bookmarked by many users, while the rest is bookmarked by a relatively low number of users. This is similar to PageRank metric that features power law distribution of PageRank values.

3.3 Correlation Between PageRank and SBRank

In this section we examine whether there is any correlation between PageRank and SBRank values. Figures 3a and 3b show scatter plots of both measures.

We observed a positive correlation coefficient ($r=0.53$ in del.icio.us and $r=0.10$ in Hatena Bookmark datasets) between SBRank and PageRank values. This is an important result, since, if the correlation coefficient had a very high value, that is, if generally SBRank values followed PageRank values, it would mean that PageRank alone adequately measures page quality. Hence, there would be no reason for its complementation with SBRank. On the other hand, if correlation coefficient between both measures had a very low absolute value, it would suggest that one of the metrics likely provides incorrect results. Since the values of the correlation coefficient were within the acceptable range, we can consider a combination of both rank estimates to be possible.

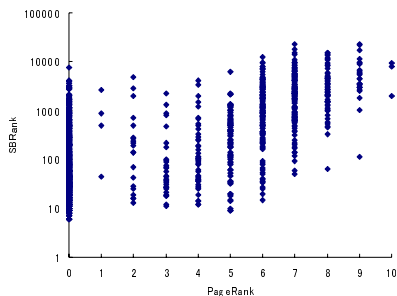


Fig. 3a. Scatter plot of PageRank and SBRank (del.icio.us) (logarithmic scale)

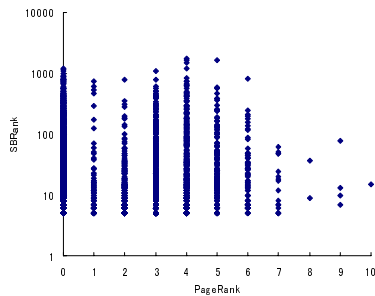


Fig. 3b. Scatter plot of PageRank and SBRank (Hatena Bookmark) (logarithmic scale)

3.4 Temporal Analysis

We turn now our attention to temporal aspects of the datasets. Figures 4a and 4b show plots of pages against the dates of their addition to the social bookmarking systems (firstDate). To correctly interpret these figures we have to remember that the datasets contain pages which were popular in both social bookmarking systems at the dates of the datasets' creation (December 6th, 2006 for del.icio.us and February 16th, 2007 for Hatena Bookmark datasets). firstDate indicates a date when a page had its first bookmark created, hence, when it was added to the social bookmarking system for the first time. It can be seen from Figure 4a that more than a half of the pages were listed among popular URLs in the first three months after being added into del.icio.us. The other half of the pages were bookmarked in the system for the first time more than three months ago. Hatena Bookmark dataset contains even more fresh pages. However, Hatena Bookmark is about one year younger than del.icio.us system. Nevertheless, these figures imply that social bookmarking users often prefer fresh pages. Additionally, almost all pages with PageRank values equal to 0 were posted very recently as it can be seen in Figures 5a and 5b. This last observation suggests that the pages with zero PageRank values are fresh and high-quality pages, which did not have enough time to acquire many inbound links. However, to be completely sure, one would have to know the actual origin dates of these pages¹¹.

These results highlight one of the useful aspects of SBRank comparing to link-based page ranking metrics. The standard link-based page ranking approach is not effective in terms of fresh information retrieval. This is because pages require relatively long time in order to acquire large number of in-bound links. Consequently, PageRank values of pages are highly correlated with their age. Young pages have difficulties in reaching top search results in traditional search engines even if their

¹¹ Internet Archive (<http://www.archive.org>) could possibly provide more constraints on the actual age of the pages. However, we have found that it contains past snapshots of only about 41% of pages from both datasets.

quality is quite high. Figures 6a and 6b show that there are quite low negative values of the correlation coefficients between PageRank and firstDate in our datasets ($r=-0.85$ for del.icio.us and $r=-0.51$ for Hatena Bookmark datasets). The longer the page existed in the social bookmarking systems, the higher is the probability that its PageRank value is high. We did similar experiment for SBRank vs. firstDate (see Figures 7a and 7b). The correlation coefficient, in this case, had the following values: $r=-0.49$ for del.icio.us and $r=-0.08$ for Hatena Bookmark datasets.

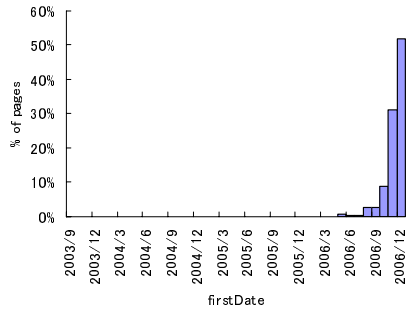
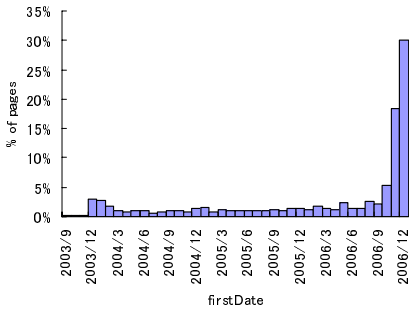


Fig. 4a. Histogram of firstDate of pages (del.icio.us)

Fig. 5a. Histogram of firstDate of pages that have PageRank value equal to 0 (del.icio.us)

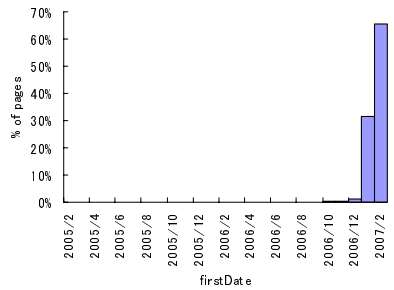
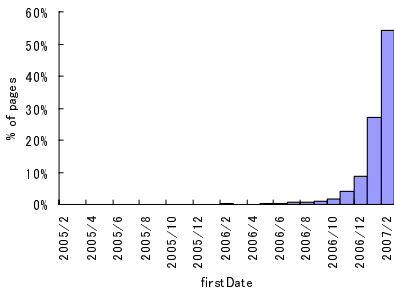


Fig. 4b. Histogram of firstDate of pages (Hatena Bookmark)

Fig. 5b. Histogram of firstDate of pages that have PageRank value equal to 0 (Hatena Bookmark)

To sum up, the results suggest that SBRank has better dynamics than the traditional link-based page ranking metric. This is because social bookmarks allow for a more rapid, and unbiased, popularity estimation of pages. Complementing PageRank using SBRank has thus potential to bring benefits from the viewpoint of the temporal characteristics of both metrics.

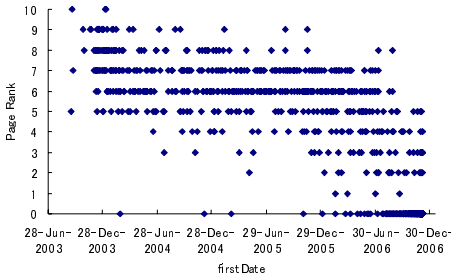


Fig. 6a. Scatter plot of firstDate and PageRank (del.icio.us)

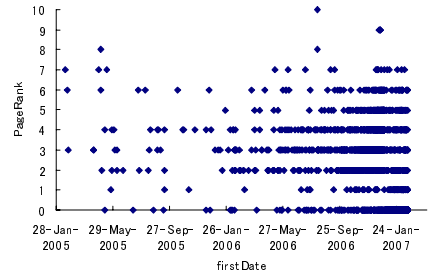


Fig. 6b. Scatter plot of firstDate and PageRank (Hatena Bookmark)

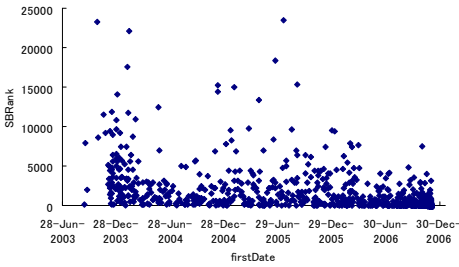


Fig.7a. Scatter plot of firstDate and SBRank (del.icio.us)

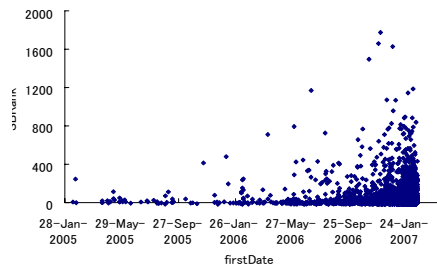


Fig. 7b. Scatter plot of firstDate and SBRank (Hatena Bookmark)

3.5 Hybrid Web Search Proposal

In this section, we demonstrate a simple method for enhancing Web search. Then we show the results of preliminary experiments that were conducted using this method. Such improvement could be simply done by re-ranking top N results returned from conventional search engines using the information about the number of their social bookmarks. First, in order to examine whether such an approach would be feasible, we analyzed how many pages in top search results contain at least one social bookmark. Table 1 shows results obtained using Google search engine and del.icio.us social bookmarking system for several sample queries. We can see that, on average, about 79% of pages returned from Google search engine contain any social bookmarks in del.icio.us and about 23% in Hatena Bookmark.

To implement a joint rank estimation measure we propose a linear combination of both ranking metrics.

$$CombinedRank_j = \alpha * \frac{SBRank_j}{\max_{\forall j:1 \leq j \leq N} (SBRank_j)} + (1 - \alpha) * \frac{PageRank_j}{\max_{\forall j:1 \leq j \leq N} (PageRank_j)} \quad (1)$$

$SBRank_j$ is the number of bookmarks of a page j in del.icio.us, while $PageRank_j$ is a PageRank value of the page acquired using Google Toolbar. We normalize both

SBRank_j and *PageRank_j* values by dividing them by the maximum values found for all *N* pages. α is a mixing parameter with the value ranging from 0 to 1.

In the experiment we have used the following queries: “social network”, “iphone”, “nintendo wii” and “gardening”. For each query, we collected $N=50$ top search results from Google search engine. By accessing the returned results, we evaluated the relevance, quality and freshness of each page. Before the manual analysis, pages were randomly ordered to eliminate the potential bias coming from search engine ranking. Quality measures were decided based on several characteristics of pages such as professional outlook, informativeness, text size, number of unique colors and similar features. These characteristics, among others, are usually common for high quality pages [14]. Freshness was determined by analyzing temporal expressions occurring in page content and a general impression of the page’s age in case no temporal expressions could be found. Next, we calculated the average value of these three evaluation criteria for each returned page. The resulting values were then used for measuring precision and recall of the results produced by our method.

By applying Equation 1 we could plot precision-recall graphs for each query using different values of parameter α (see Figures 8 to 11). Precision and recall were computed analyzing top k ($k=\{10,20,30,40,50\}$) results within 50 pages returned by the search engine.

From Figures 8-11 it can be seen that PageRank measure used alone ($\alpha=0$) produced better results only for the query “social network” for $k=\{10, 20\}$. On the other hand, SBRank measure used alone ($\alpha=1$) produced the highest quality results for the remaining values of k for query “social network” and for $k=\{20,30,40\}$ for query “iphone”. In case of other queries, the hybrid approach was better or at least equally good as PageRank or SBRank measures used alone.

After averaging the precision-recall graphs for all the queries we noticed that the combined approach tends to produce better results for $k=\{10\}$ (Figure 12). On the other hand, there is no improvement of the quality of search results for $k=\{20,30,40\}$ when comparing to PageRank or SBRank used alone.

Choosing the value of the mixing parameter α is a difficult task. As a possible solution, we suggest relating it to one of the two factors, the age of pages or the availability of social bookmarks for pages. Thus, we propose the following three approaches that could be potentially used, in which α_j is a mixing parameter whose value depends on the characteristics of page j :

$$\alpha_j = \frac{1}{t_j^{now} - t_j^{add}} \tag{2}$$

$$\alpha_j = \frac{1}{t_j^{now} - t_j^{cre}} \tag{3}$$

$$\alpha_j = \begin{cases} 0 & \text{if } PageRank_j > SBRank_j \\ 1 & \text{if } PageRank_j \leq SBRank_j \end{cases} \tag{4}$$

Here, t_j^{now} is the time of query issuing, t_j^{add} is the date of the addition of the page j into a social bookmarking system; t_j^{cre} is the creation date of the page j . However, detecting creation dates of pages is rather difficult. As a possible solution, creation

dates could be approximated by choosing the minimum value between t_j^{add} and the earliest timestamp of past snapshots of the page j found in any web archive such as the Internet Archive.

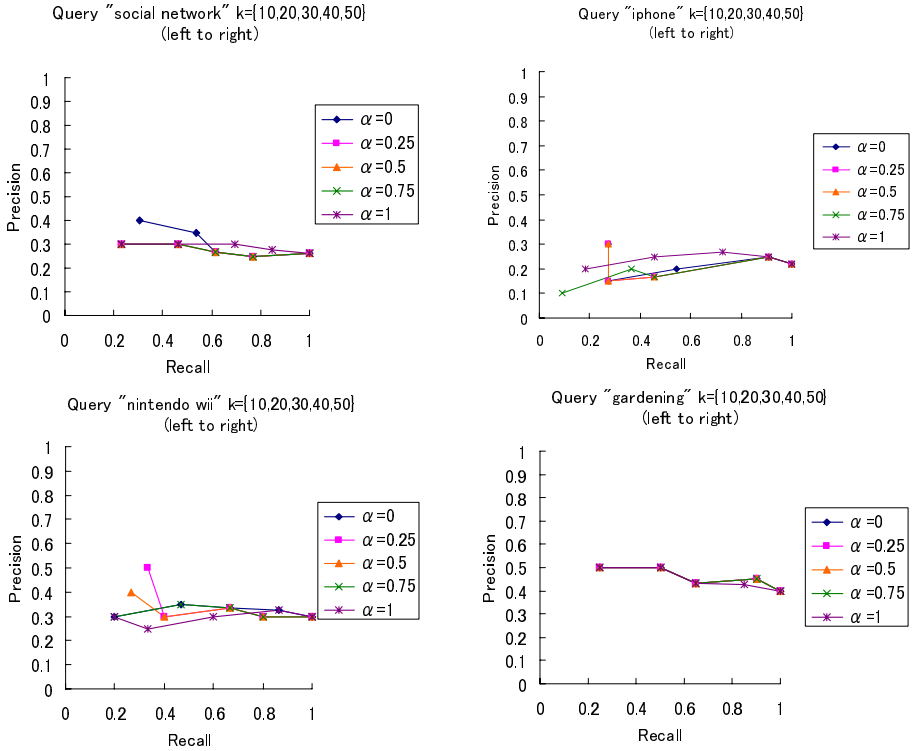


Fig. 8-11. Precision-recall curves for each query: “social network”, “iphone”, “nintendo wii” and “gardening”

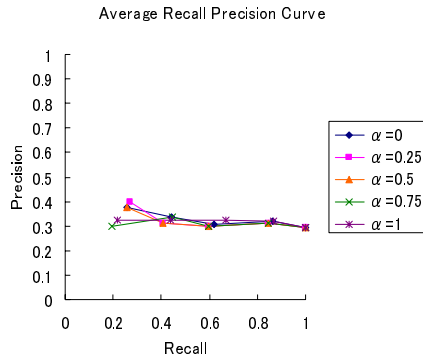


Fig. 12. Average precision-recall curves

Table 1. Number of pages having any social bookmarks for the top 100 results returned from Google search engine for sample queries

query	hatena	del.icio.us	both
graphic design	12	69	69
java	42	92	92
apple	19	83	84
gardening	4	79	79
kyoto	11	61	62
iphone	17	73	74
ipod	37	79	79
steve jobs	26	74	74
ajax	58	86	86
digital library	13	91	91
social network	22	84	84
nintendo wii	14	71	73
<i>Average</i>	<i>22.92</i>	<i>78.5</i>	<i>73</i>

By using Equations 2 and 3, a page would be ranked more by its SBRank measure the younger is its date of insertion into social bookmarking systems or the younger is its actual origin age. This approach would favor the social bookmark-based ranking method in the case of relatively young pages. On the other hand, the approach based on Equation 4 would select the ranking metric that provides a higher value. Thus, pages with few social bookmarks would be ranked more by their PageRank values.

4 Discussion

Web search algorithm that would exploit consumer generated input, which constitutes so-called Web 2.0, is certainly an attractive idea. Several possible directions can be followed to achieve such a “socially-aware” search. For example, swicky¹² is an application that enables building trustful, vertical search engines by communities of users. Our approach is different as we focus on employing social bookmarks made by Web users since they have many advantages over links. Intuitively, there are two main reasons why users create social bookmarks, making pointers to pages for their future reuse or sharing information with other users. It means that either the users expect to revisit bookmarked pages in future or they want to make them known to others. Both objectives allow us to consider social bookmarks as positive votes for pages. Additionally, if we roughly divide Web users into content creators and content consumers, then PageRank can be interpreted as a result of *author-to-author evaluation* of Web resources. On the other hand, SBRank can be considered as a

¹² <http://swicki.eurekster.com/>

result of *reader-to-author evaluation*. Thus, users who are not capable of creating and managing Web documents could also cast votes for pages leading to a more democratic search process. Another advantage of SBRank over PageRank is that it seems to have better temporal characteristics. SBRank is more dynamic than PageRank, and it often takes short time for pages to reach their popularity peaks in social bookmarking systems [6].

Below, we summarize the observations that we made through our analysis:

- More than half of popular pages in the datasets have lowest PageRank values
 - This implies that many pages which have low PageRank values can be incorporated into top search results through a hybrid Web search
 - It also suggests that people likely discover bookmarked pages from other sources rather than from search engines since many pages in our datasets are relatively difficult to be found by traditional search engines
- Few pages have high SBRank while many pages have rather low SBRank
- There is a weak positive correlation between SBRank and PageRank
 - This result suggests the possibility that SBRank can complement PageRank to enhance Web search
- About half of pages listed as popular in the social bookmarking systems have been introduced in recent three months
 - This indicates high dynamics of SBRank measure and in general of social bookmarking systems as they enable pages to become rapidly popular
 - It also suggests that there are many fresh pages in social bookmarking systems
- There is a high negative correlation between firstDate and PageRank values
 - This result is consistent with the previous observations demonstrating the strong positive bias of PageRank metric towards old pages

In our analysis, we have not considered page relevance that could be estimated by using tags assigned to pages. For example, in the context of link structure analysis, Haveliwala [8] introduced topic-sensitive PageRank. It measures page importance in relation to selected topics, thereby improving page ranking. Similar approach could be adapted to social bookmarks. In this paper, however, we focus on popularity estimation of pages rather than on their relevance.

SBRank is based on user bookmarking activities, however, the importance of each bookmark may be different. A possible extension of our approach would be, thus, to incorporate weighting scheme into SBRank calculation that would depend on the characteristics of users bookmarking pages. This could improve the effectiveness of the page ranking and could help combat potential spamming. Spamming is a threat for every Web search algorithm. Although, until now, no significant spamming attacks have been observed in social bookmarking systems, we think that necessary measures must be taken to prevent deliberate manipulations of social bookmarks in the future. Several measures could be undertaken here as possible lines of defense. For example, user popularity and the history of her or his interactions with the system could be analyzed or users could report suspected inputs themselves.

Lastly, scalability is another problem related to social bookmarking-based Web search. We believe that more data will soon become available along with the

increasing popularity of social bookmarking. Also we hope that efficient meta-search approaches will appear in the near future. In the current situation, we think that the combination of link-based ranking metric and social bookmark-based one is an optimal strategy.

5 Conclusions

Social bookmarks have potential to complement and improve the traditional search in the Web as bookmarked pages are manually checked by multiple Web users, who express their preferences towards pages. Besides improving the quality estimation of pages, social bookmarks can enhance freshness of search results, which is the quality that many search engines currently lack.

In this paper, we investigated the possibility of merging the ranking methods based on the link analysis with the one based on social bookmarks. We have done quantitative studies aiming at comparing both popularity measures and their temporal characteristics. In result of the comparative analysis, we were able to make several observations which allow us to conclude that a hybrid Web search is feasible and useful. We believe that such an analysis is important for the creation of novel search applications considering the weakness of link-based ranking algorithms and the increasing popularity of social collaboration systems in the Web.

In future, we would like to continue the experiments in order to test the proposed approaches. We plan also to work on designing meta-search approaches for improving the search scalability as well as on spam-resistant ranking algorithms.

Acknowledgements

This research was supported by the MEXT Grant-in-Aid for Scientific Research in Priority Areas entitled: Content Fusion and Seamless Search for Information Explosion (#18049041, Representative Katsumi Tanaka), and by the Informatics Research Center for Development of Knowledge Society Infrastructure (COE program by MEXT) as well as by the MEXT Grant-in-Aid for Young Scientists B entitled: Information Retrieval and Mining in Web Archives (Grant#: 18700111), and by “Design and Development of Advanced IT Research Platform for Information” (Project Leader: Jun Adachi, Y00-01, Grant#: 18049073).

References

1. Amitay, E., Carmel, D., Herscovici, M., Lempel, R., Soffer, A.: Trend Detection Through Temporal Link Analysis. *Journal of The American Society for Information Science and Technology* 55, 1–12 (2004)
2. Baeza-Yates, R., Castillo, C., Saint-Jean, F.: Web Dynamics, Structure and Page Quality. In: Levene, M., Pouloussis, A. (eds.) *Web Dynamics*, pp. 93–109. Springer, Heidelberg (2004)
3. Bry, F., Wagner, H.: Collaborative Categorization on the Web: Approach, Prototype, and Experience Report. *Forschungsbericht/research report* (2003)

4. Cho, J., Roy, S., Adams, R.: Page Quality. Search of an Unbiased Web Ranking. In: Proceedings of SIGMOD Conference, pp. 551–562 (2005)
5. Dwork, C., Kumar, R., Naor, N., Sivakumar, D.: Rank Aggregation Methods for the Web. In: 10th World Wide Web Conference, pp. 613–622 (2001)
6. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems, *Journal of Information Science*, 198–208 (2006)
7. Gomes, D., Silva, M.J.: Modeling information persistence on the Web. In: Proceedings of the 6th International Conference on Web Engineering, Palo Alto, CA, USA, pp. 193–200 (2006)
8. Haveliwala, T.H.: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Transactions on Knowledge and Data Engineering*, 784–796 (2003)
9. Keller, R.M., Wolfe, R.R., Chen, J.R., Labinowitz, J.L., Mathe, N.: A Bookmarking Service for Organizing and Sharing URLs. In: Proceedings of the 6th International World Wide Web Conference, Santa Clara, CA, pp. 1103–1114 (1997)
10. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 604–632 (1999)
11. Lawrence, S., Giles, C.L.: Inquirus, the NECI meta search engine. In: Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, pp. 95–105 (1998)
12. Liu, B.: *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*. Springer, Heidelberg (2007)
13. Lu, Y., Meng, W., Shu, L., Yu, C., Liu, K.: Evaluation of Result Merging Strategies for Metasearch Engines. In: Proceedings of Web Information Systems Engineering conference, pp. 53–66 (2005)
14. Mandl, T.: Implementation and evaluation of a quality-based search engine. *Hypertext2006*, pp. 73–84 (2006)
15. Marlow, C., Naaman, M., Boyd, D., Davis, M.: HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, To Read. *Hypertext2006*, pp. 31–40 (2006)
16. Mathes, A.: Folksonomies - Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, LIS590CMC* (2004)
17. Ntoulas, A., Cho, J., and Olston, C.: What's new on the Web? The evolution of the Web from a search engine perspective. In: Proceedings of the 13th International World Wide Web Conference, New York, USA, pp. 1–12 (2004)
18. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
19. Strutz, D.N.: Communal Categorization: The Folksonomy. *INFO622: Content Representation* (December 2004)
20. Wu, X., Zhang, L., Yu, Y.: Exploring Social Annotations for the Semantic Web. In: World Wide Web Conference, pp. 417–426 (2006)
21. Wu, H., Zubair, M., Maly, K.: Harvesting Social Knowledge from Folksonomies. *Hypertext2006*, pp. 111–114 (August 2006)
22. Zhang, L., Wu, X., Yu, Y.: Emergent Semantics from Folksonomies: A Quantitative Study. In: Spaccapietra, S., Aberer, K., Cudré-Mauroux, P. (eds.) *Journal on Data Semantics VI. LNCS*, vol. 4090, pp. 168–186. Springer, Heidelberg (2006)
23. Yu, P.S., Li, X., Liu, B.: On the temporal dimension of search. In: Proceedings of the 13th International World Wide Web Conference, New York, USA, pp. 448–449 (2004)

Designing Interaction Spaces for Rich Internet Applications with UML

Peter Dolog and Jan Stage

Aalborg University, Department of Computer Science,
Fredrik Bajers Vej 7, DK-9220 Aalborg East, Denmark
{dolog, jans}@cs.aau.dk

Abstract. In this paper, we propose a new method for designing rich internet applications. The design process uses results from an object-oriented analysis and employs interaction spaces as the basic abstraction mechanism. State diagrams are employed as refinements of interaction spaces and task models to specify synchronization events and follow up actions on the client and server side. The notation is based on UML.

Keywords: Rich Internet Applications, software design, interaction spaces, state diagrams.

1 Introduction

Rich internet applications have been introduced as a response to the limitations in richness that web users experience compared to desktop applications [7, 8]. They employ technologies such as Flash from Adobe, ActiveX, or recently AJAX technology [11]. The term Rich Internet Applications has been launched first by Macromedia. Later, this technology has also been adopted by others, for example Google and Flickr.

Such applications introduce additional complexity connected with asynchronous communication and synchronization problems as some data are being held and processed at the client side. This influences software engineering methods to build such applications.

In this paper we propose a method for rich internet application design with UML. The method combines interaction spaces and task models [10] with UML based design for adaptive web applications [6]. This combination provides the following advantages:

- Interaction spaces and task models are natural metaphors for designing user interface fragments that a web user will interact with and move beyond traditional web site and web content abstractions.
- The UML statechart diagrams provide means to define user interaction events, synchronization events between client and server as well as synchronization events between fragments of the user interface.

The rest of the paper is structured as follows. Section 2 presents the method for designing rich internet applications with illustrative examples. Section 3 relates the method to the other work in the area. Finally, section 4 provides conclusions and proposals for further work.

2 Design Process and Techniques

Figure 1 is an activity model of our method where:

- **Data Model** describes the problem domain classes needed for application and that are used in use cases and tasks;
- **Use Case Model** provides the context for which the software system is going to be used;
- **Task Model** is a refinement of each use case. It describes the activities that are performed during each use case in terms of a UML statechart diagram.
- **Interaction Space Model** is a refinement of each task model. It describes structural details of corresponding task flows where a user interaction is needed.
- **Guide Model** is a refinement of the task model. It provides navigation and synchronization details on user interaction from software behaviour point of view.
- **Mapping to Implementation** maps the design abstractions to the appropriate implementation according to the UML Guide principles [5] employing tagged values, side effect actions and transformations to the running code.

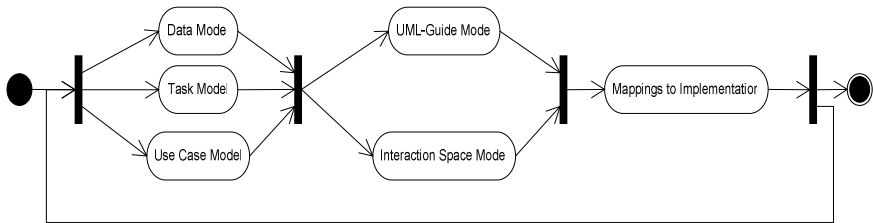


Fig. 1. A design process for rich internet applications

Scenario. The discussion in this section is based on an example of a rich internet application for furniture configuration. The case envisions a website for buying furniture (like IKEA). Consider for example that a user has found a piece of furniture that he considers buying for his living room. He would like to see if it fits into the room with his other furniture and find the model and colour that looks best. He chooses the model page to accomplish this. In that page, he first draws the living room by giving sizes of floor and wall. He also gives the walls and floor colours. Then he puts in furniture sketches of his existing furniture. This is simply done by selecting types and colours from a palette. Now he puts in the new piece of furniture. He determines the various properties of this item, e.g. colour and model, by choosing from the online catalogue. He can change his point of view to see the furniture from different angles. He is able to change the room by moving the existing furniture

around. When he is finished, he saves the furniture configuration under his own profile. Once the room is finished, it can be posted to the application server in order to allow the shop to determine whether the requested items are in stock and if necessary recommend alternative options.

Data and Use Case Model. Data and Use Case modelling follows traditional object oriented principles [9] when describing application domain classes, concepts, associations, aggregations and generalization/specialization.

A number of use cases can be identified from the scenario in above such as *Draw a room with walls and floor*, *Make a furniture configuration with existing furniture (Room items)*, *Put in new furniture (Store items)*, *Edit a furniture configuration (change colour etc.)*, and so on. Each use case is described in a textual version and can be depicted in a graphical form on a use case model showing the relationships to the other use cases.

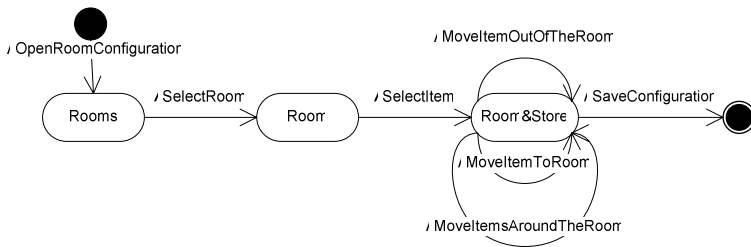


Fig. 2. An excerpt of a task model for furniture configuration

Task and Interaction Space Model. Each use case can also be expressed as a statechart diagram [9]. An excerpt of a task model for our scenario is shown in Figure 3. The transitions are user actions, and the states represent the results of the user actions displayed at a user interface. In accordance with [10], the task models are enriched with *interaction spaces*, forming the elements of user interaction design for each use case. Interaction spaces are conceptual elements which prescribe how particular tasks will be supported by a user interface.

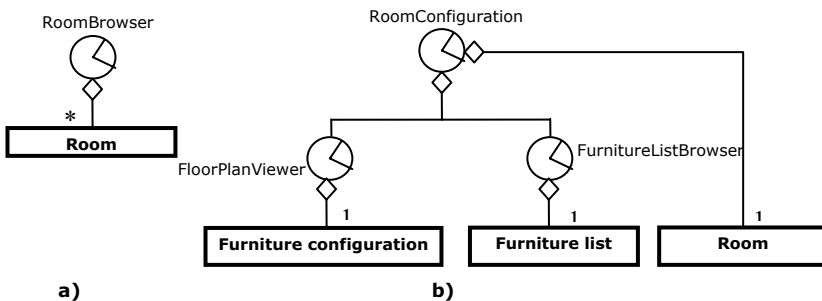


Fig. 3. Interaction spaces for room furniture configuration with data model classes: a) SelectRoom b) SelectItem and move it

Figure 4 depicts a typical example of interaction spaces with a browser and a view for a problem domain class. Furthermore, each interaction space is further connected to data classes that it is using. For each transition and state in the task model, we consider relevant interaction spaces. For example, we need a *Room browser* that shows all the user’s rooms. Selection is made by pointing out one object in the list. We need a *Floor plan viewer* and a *Furniture list browser* to show the contents of the room. In this list, an object can be selected by pointing it out. Moving around should be a drag and drop type of function. These two interaction spaces are parts of an overall interaction space that we call *RoomConfiguration*. Finally, we need an interaction space for saving the furniture configuration.

Guide Model. *Statechart diagrams* are used in the UML-Guide [6] for modelling user navigation in a hypertext; each *state* represents the production of a given information chunk on the device observed by a user, and each *state transition* represents an event caused by user interaction that leads to the production of a new chunk of information. State diagrams therefore provide an abstraction of *hypertext trails*, where each trail can be adapted by taking into account the user background, level of knowledge, preferences and so on [5, 6]. Atomic states, super states, history states, fork and join are additional symbols to describe composition, concurrent execution, remembering, and so on. *Events, guards, and side effect actions* are used to specify constraints, triggers, operations for example for synchronizing user and display data as well as adaptation rules.

As each interaction space has already data context through a link to data classes and to the tasks to be performed, it is sufficient to map set of tasks from task model to UML states. Functional dependencies between interaction spaces and their states are modelled as state transitions with parameter passing. These transitions ensure that the user interface will be in consistent and synchronized state. In case of highly interactive activities at the user interface the transitions also ensure that the data is updated according to the user activity.

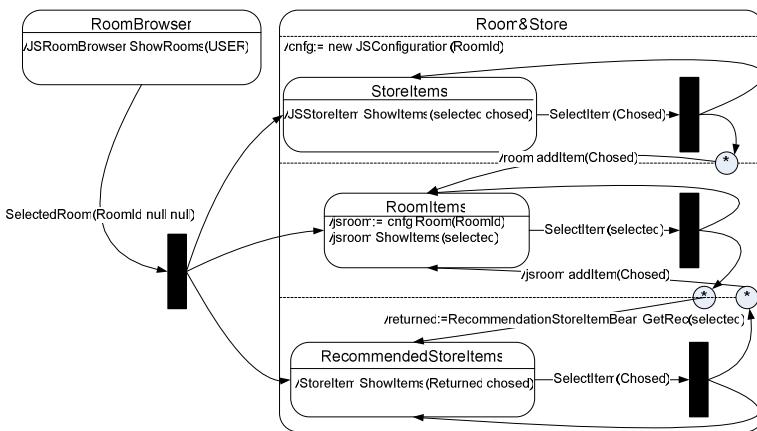


Fig. 4. An excerpt of the UML guide model for furniture configuration

Figure 4 depicts an excerpt of a navigation model according to UML-Guide. The model is a refinement of the task model presented in the beginning of this section. There are two dialog steps: *RoomBrowser* and *Room&Store*. The *Room&Store* incorporates 3 concurrent regions representing three concurrently presented dialog steps: *RoomItems*, *StoreItems* and *RecommendedStoreItems*. Whenever a user selects an item from store items (*SelectItem* event on the transition from *StoreItems*), the item is placed into the room by synchronization invoked *Room.addItem* function. The model at the client side is reinitialized and client side functions such as *jsRoom.ShowItems()* re-renders the displays considering the added item as part of the scene. More interesting situation happens whenever user selects an item in a room: an event occurs which triggers a synchronizing action with a server of furniture shop and recommended items based on similarity functions and user preferences (*GetRec(selected)* function). Additional parameters can be specified such as booking history and so on.

3 Related Work

A WebML extension for rich internet applications has been proposed in [2] with new units for the client side operation of a web application. In our approach, we propose an alternative UML-based design technique focusing user interaction and behavioral characteristics of navigation with client site business logic and asynchronous communication between server and client.

Task models have been already employed as a means to model complex processes for web applications in WSDM [4]. Similarly to our approach, the tasks are used to describe control flow between user activities and their decompositions to web application internal operations. In our work we use the tasks model together with the interaction spaces. Furthermore, we use decomposition techniques to UML Guide for adaptive navigation design to specify synchronization of different web page fragments and request the data needed for them.

Interactive Dialog Model (IDM) [1] relates to our approach through its focus on user interaction. It is based on a map based technique for user dialog specification on the web connected to data and content.

SHDM [3] uses semantic web conceptual model to specify structural features of the user interface widgets. Widgets are also a central part of the interaction space specifications. They are used to specify parts of the interaction space in terms of data and interaction facilities.

In [12], authors make use of sequence diagrams to synchronize different devices. The technique might be applicable for the rich internet applications as well in combination with the techniques proposed in this paper and in our technique.

4 Conclusion

We have proposed a new UML-based design method for rich internet applications. It is based on the task models, interactions spaces, and web interaction and navigation specification in the UML-Guide. It provides direct means to analyze user interaction

as well as asynchronous communication between web client and web server. It reflects the need to have a part of the business logic at the client side. It provides a flexibility with respect to the web page design as this decisions are made later in the design stage according to the UML-Guide principles.

In our further work we plan to conduct larger set of studies about features of this technique. We would like to also see how to further support a designer with some tools which would ease the design refinements.

References

1. Bolchini, D., Paolini, P.: Interactive Dialogue Model: A Design Technique for Multichannel Applications. *IEEE Transactions Multimedia* 8(3) (2006)
2. Bozzon, A., Comai, S., Fraternali, P., Carughi, G., T.: Conceptual Modeling and Code Generation for Rich Internet Applications. In: ICWE2006. International Conference on Web Engineering, Palo Alto, California USA, ACM Press, New York, NY, USA (2006)
3. De Moura, S., Schwabe, D.S.: Interface Development for Hypermedia Applications in the Semantic Web. In: La-Web 2004 Proceedings. IEEE Press, Orlando, Florida (2004)
4. De Troyer, O., Casteleyn, S.: Modeling Complex Processes for Web Applications using WSDM. In: Lovelle, J.M.C., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) ICWE 2003. LNCS, vol. 2722, Springer, Heidelberg (2003)
5. Dolog, P.: Engineering Adaptive Web Applications. Doctoral dissertation. University of Hannover (March 2006)
6. Dolog, P., Nejdil, W.: Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems. In: Stevens, P., Whittle, J., Booch, G. (eds.) «UML» 2003 - The Unified Modeling Language. Modeling Languages and Applications. LNCS, vol. 2863, Springer, Heidelberg (2003)
7. Driver, M., Valdes, R., Phifer, G.: Rich Internet Applications Are the Next Evolution of the Web. Technical report, Gartner (May 2005)
8. Duhl, J.: Rich Internet Applications. White Paper, IDC (November 2003)
9. Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., Stage, J.: Object-Oriented Analysis & Design. Aalborg: Marko (2000)
10. Nielsen, C.M., Overgaard, M., Pedersen, M.B., Stage, J., Stenild, S.: Exploring Interaction Space as Abstraction Mechanism for Task-Based User Interface Design. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) TAMODIA 2006. LNCS, vol. 4385, Springer, Heidelberg (2007)
11. Paulson, L.D.: Building Rich Web Applications with Ajax. *Computer* 38(10), 14–17 (2005)
12. Vandervelpen, C., Vanderhulst, G., Luyten, K., Coninx, K.: Light-Weight Distributed Web Interfaces: Preparing the Web for Heterogeneous Environments. In: Lowe, D., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 197–202. Springer, Heidelberg (2005)

A Behavioral Model for Rich Internet Applications

Sara Comai and Giovanni Toffetti Carughi

Dipartimento di Elettronica, Politecnico di Milano
P.zza L. da Vinci, 32 - I20133 Milano
comai@elet.polimi.it, toffetti@elet.polimi.it

Abstract. Rich Internet Applications (RIAs) are reshaping the way in which the Web works. They change not only the appearance of the Web interfaces, but also the behavior of applications, permitting novel operations, like data distribution, partial page computation, and disconnected work. In this paper we try to understand the differences between the behavior of traditional dynamic Web applications and RIAs, considering the WebML modeling language and its actual implementation.

1 Introduction

In a “traditional” dynamic HTML-based application the interface is an HTML document, computed by the server at each user’s request. When the user interacts with the page, by following an hyperlink or submitting a form, the server is invoked and the destination page is computed from scratch and sent back to the client. The role of the client is to intercept the user’s action, deliver the request to the server, and display the response.

In a Rich Internet Application (RIA), the client is assigned a fraction of the data and of the computation, so that the user can perform a complex interaction with the interface without invoking the server, unless data needs to be exchanged. Furthermore, if the user’s interaction requires a server round-trip for refreshing some data, the client can selectively retrieve from the server only the information that need to be changed, update its internal status, and redisplay the modified content.

This paper investigates RIAs by analyzing their behavior in order to understand if they constitute only a technological improvement over the architecture of conventional dynamic Web applications or if they can alter significantly the way in which Web applications behave, and therefore pose novel challenges to their design, implementation, and evaluation. As a reference model we consider WebML [2].

2 Traditional Web Application Model

A traditional dynamic Web application is described by its structure and behavior, as defined by the following models.

The **structural model** comprises a *data model*, which specifies the content objects underlying the applications (e.g., represented by an Entity-Relationship or an UML class diagram), and an *interface model* (or hypertext model), which describes the front-end exposed to the user.

The key ingredients of the interface model are: 1) content components¹, 2) interface containers, 3) parameter passing, and 4) interaction mechanisms.

1) A *content component* is used to extract content stored in the data layer and is characterized by: a (possibly empty) set of *input parameters*, a *query*, to retrieve the desired objects, exploiting the values of the input parameters; a *population*, storing the result of the query, and a (possibly empty) set of *output parameters*.

2) *Interface containers* are structured collections of content components. In the Web context, they typically represent Web *pages*, or sub-modules of pages and can be organized hierarchically.

3) *Parameter passing* is represented by *parameter passing rules*. A parameter passing rule expresses a dependency between a pair of components; it consists of a *source component*, a *destination component*, and a *mapping* between the output parameters of the source and the input parameters of the destination.

4) *Web interaction mechanisms* are specified by means of *links*, representing hyperlinks and input submit commands, typically rendered as anchors or form buttons. A link connects a source and a destination component and is associated with one parameter passing rule.

The **dynamic model** of a traditional Web application explains what happens when the user accesses a page or navigates its internal links. The behavior of a traditional Web application can be automatically inferred from the structural model presented above. In this section we present a sketch of the the page computation algorithm adopted by WebML and highlight the main characteristics of the behavior of traditional Web applications. A formal version of the WebML semantics defined by means of Statecharts can be found in [3].

In traditional Web applications page computation occurs entirely at the server: a page request causes the page to be computed from scratch as described by the following algorithm.

```

INPUT: page to be computed, initial assignment of input parameters IP
OUTPUT: computed page
CURRENTINPUT=IP; COMPONENTS=FindComputable(CURRENTINPUT);
WHILE (COMPONENTS is not empty) DO {
  C=Choose(COMPONENTS);
  I=ChooseInput(C);
  OutP=Execute(C, I);
  FORALL Rule in ParameterPassingRules(C)
    CURRENTINPUT += AssignInput(Rule.destination,OutP);
  COMPONENTS=FindComputable(CURRENTINPUT);
}

```

¹ For brevity we don't consider business components (i.e. WebML operations) here.

The page computation process starts from the requested page and from an initial assignment of values to the input parameters of the content components (e.g., the OID of the object selected from an index, the values input in a form). The maximal set of content components is then computed, starting from the received parameters and respecting a partial order imposed by the parameter passing rules: the set of computable units is determined (procedure *FindComputable*); one of such units is selected for execution (procedure *Choose*); the input parameters to be used are chosen (*ChooseInput*) and the query/method of the unit is executed and output parameters are computed (*Execute*). The output parameters are then passed through the parameter passing rules to the destination components (*AssignInput*).

As an example, consider the hypertext in Figure 1 showing the list of folders, a selected folder, the list of messages contained in this folder and a selected message. L1 is a navigation link recording the last selected value (it is an history link, denoted with h), L2 is a parameter passing rule not associated with a navigable link, L3 is a navigation link with no history. When the user selects a new message in the current folder by navigating link L3, the page must be recomputed. The initial assignment of input parameters contains the id of the newly selected message, while the history contains the id of the current folder. In the first iteration, it can evaluate the *Folders* component, which does not require input parameters, and the *Message* component, whose input parameter comes from the initial assignment. When the *Folders* component is evaluated, it provides the value of the input parameter of the *Folder* component (taken from the history). After the *Folder* component is evaluated, the input parameter of the *Messages* component becomes available and also this component can be evaluated, which terminates the page computation process.

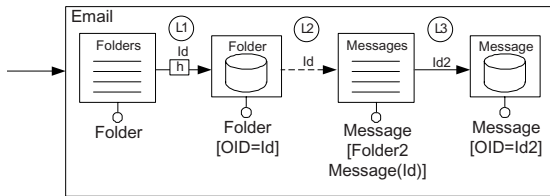


Fig. 1. An hypertext with four components, three parameter passing rules, and two navigational links

3 Rich Internet Application Dynamic Model

Rich Internet applications require the extension of the structural model with a distribution of the data and of the functionalities between the client and the server (see [1] for details) and a more flexible run-time behavior w.r.t. traditional applications. Upon the interaction of the user, the client can selectively request from the server only a portion of the data and maintain unchanged all the pieces of information that are not affected. Furthermore, the interaction of the user may

cause some pieces of content, which were previously displayed, to be invalidated because they are no longer consistent with the rest of the page.

A novel dynamic model where each link is associated with the notion of *computation sequence* (or *sequence*, for short) is needed; when a link is navigated, the dynamic model dictates explicitly the effects on all the components of the page. In a sequence, the following operators are used:

- *Evaluate*: causes the computation of the input parameters of a component and of its query. It is denoted by the term $c_i ++$, where c_i is a component.
- *Refresh*: causes the computation of the query. It is denoted by the term $c_i +$.
- *Invalidate*: discards the input parameters of a component and its population. It is denoted by the term $c_i -$.
- *Empty*: discards the population of a component. It is denoted by $c_i -$.

Given a link, a sequence associated with it is *legal* if it satisfies the following constraints: 1) the operators affect components belonging to the page of the destination component; 2) each component appears at most once in the sequence associated with an *Evaluate* or a *Refresh* operator, thus avoiding cyclic computations; 3) the order of evaluation of the components is such that all the components that may provide input parameters to a given component are evaluated before the dependent component.

The proposed dynamic model subsumes the traditional one, and is able to express also more complex behaviors.

Examples. Consider the interface model of Figure [11](#).

The sequence on link L1: Folder++, Messages++, Message– specifies a behavior where only the components directly depending on the navigated link are updated, whereas the remaining components are no longer displayed. In other words, only the affected information is requested again to the server, as customary in AJAX and FLASH interfaces.

The sequence on link L1: Folder++, Messages++ specifies instead that when link L1 is navigated, the *Message* component is kept fixed and continues to show the same information irrespective of the (possibly repeated) change of folder. Such a behavior may be chosen to spare a data request to the server, and defer the re-evaluation of the displayed message to the time when the user selects a new message.

Extended page computation algorithm. The page computation process for the extended model is represented by the following algorithm:

```

INPUT: page, computation sequence Seq,
navigated link L, initial parameter assignment IP;
OUTPUT: computed page
AssignInput(L.destination,IP);
FOREACH (op in Seq) do {
  SWITCH (op.type){
    case Evaluate:
      (I,AffectedQuery)=ChooseInput(op.component);
      OutP=Execute(op.component, I, AffectedQuery); break;
  }
}

```

```

case Refresh:
    OutP=Execute(op.component); break;
case Invalidate:
    op.component.Input=op.component.Population=null;
    op.component.Output=null; break;
case Empty:
    op.component.Population=null;
    op.component.Output=null;
}
FORALL Rule in ParameterPassingRules(op.component)
    AssignInput(Rule.destination,null);
}

```

4 Discussion

The study and the implementation of the page computation algorithms of the traditional and of the Rich Internet Applications models allowed us to better evaluate their main differences.

The behavior of the whole application becomes more difficult to understand, since it is given by the combinations that can be obtained through the application of the computation sequences associated with the enabled navigational links. The specified behavior should always guarantee properties such as computation termination and determinism, triggerability of links and computability of components, and so on. The traditional model presented in Section 2 guaranteed most of such properties, while the new extended model introduces also new issues. Here we overview only some properties of the proposed models, summarized in Table 1.

Table 1. Traditional vs RIAs guaranteed properties

Property	Traditional	Rich Internet Application
Component computability	NO (cycles)	NO (due to cycles and/or partial computation)
Reachability	YES	NO (due to partial computation)
Freshness	YES	NO (due to partial computation)
Consistency	YES	NO (due to partial computation)

Non-computability of a component can arise in both applications. It occurs when a component may never receive the needed input parameters: this may happen in particular configurations with cycles among components. In RIAs, non-computability may also originate from computation sequences in which explicit invalidation actions prevent a component from receiving all its needed parameters. For example, in Figure 1 the computation sequence (L1: Folder++, Messages-; Message++) associated to the navigation of link (L1) cannot produce the content of the Message component.

The proposed RIA model is then affected by further issues due to possibility of specifying partial computations. First of all, partial computations may cause non-reachability of components and triggerability of links: the set of sequences

associated with the links of a page may not allow to compute components and therefore trigger their outgoing navigational links. The model for the traditional Web applications, computing the maximal set of content components and using the parameter passing rules at each user's interaction, can instead guarantee such properties.

Moreover, stale data may be present in the interface. In Figure 1, if the computation sequence associated with link L3 is (Message++), only the *Message* component is recomputed. The list of messages is not recomputed and, therefore, new messages available at the server are not shown. *Data freshness* problems do not arise in traditional dynamic Web applications, because pages are always entirely recomputed from the data currently stored in the data layer.

Finally, *consistency* problems may arise. By consistency we mean that if a component receives inputs from another component, then whenever the population of the first component is evaluated and new output parameters are produced, also the input parameters and the population of the dependent component are evaluated. This problem does not arise with the proposed algorithm for traditional Web applications, since the computation always starts from scratch by evaluating the maximal set of components and parameter passing rules apply; it may instead occur in RIAs when partial computations are specified. Consider Figure 1 and the computation sequence (L1: Folder++, Messages++) associated to the navigation of link L1; consistency is not guaranteed between the *Messages* component and the *Message* component that is not recomputed: indeed, if the user selects a new folder, a message belonging to a different folder is shown.

5 Conclusions

In this paper we have presented a dynamic model for Web applications, distinguishing between traditional Web applications and Rich Internet Applications. In case of traditional Web applications a default behavior may be associated with the structural model as it has been done for the WebML language. RIAs can exhibit richer and more flexible behaviors that cannot be specified through a standard behavioral model. In particular, the computation of RIAs typically involves partial fragments of the interface: this may introduce several new issues that need to be considered in the design, implementation, and evaluation of RIAs.

References

1. Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G.: Conceptual modeling and code generation for rich internet applications. In: Wolber, D., Calder, N., Brooks, C., Ginige, A. (eds.) ICWE, pp. 353–360. ACM Press, New York (2006)
2. Ceri, S., Fraternali, P., Brambilla, M., Bongio, A., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, Seattle, Washington (2002)
3. Comai, S., Fraternali, P.: A semantic model for specifying data-intensive web applications using webml. In: Semantic Web Workshop, Stanford, USA, July (2001)

Considering Web Accessibility in Information Retrieval Systems

Myriam Arrue and Markel Vigo

University of the Basque Country, Informatika Fakultatea, Manuel Lardizabal 1, E-20018,
Donostia, Spain
{myriam,markel}@si.ehu.es

Abstract. Search engines are the most common gateway for information searching in the WWW. Since Information Retrieval systems do not take web accessibility into account, results displayed are not useful for users with disabilities. We present a framework that includes the requirements to overcome this situation. It is composed of three modules: Content Analysis Module, Accessibility Analysis Module and Results Collector Module. This framework facilitates the implementation of search engines which return results ranked according to accessibility level as well as content relevance. Since criteria to sort results by their accessibility are necessary, we define accurate quantitative accessibility metrics which can be automatically calculated exploiting results yielded by any automatic evaluation tool. A prototype based on these requirements has been implemented to show the validity of the proposal.

1 Introduction

The WWW has a great potential to make life easier for disabled people and make them less dependant on their relatives or friends since users can perform tasks they hardly could accomplish by themselves (i.e.: do shopping, buy tickets, etc.). However, as most websites are not accessible, these users come up against design barriers which do not let them access the information. In order to tackle this situation Web Content Accessibility Guidelines [7] were proposed by the Web Accessibility Initiative¹ (WAI). WCAG 1.0 guidelines define specific testing techniques or checkpoints which refer to accessibility issues in a more accurate way. Depending on the way a checkpoint impacts on the accessibility of a web page, each checkpoint has a priority assigned (1, 2 or 3 from more to less impact). In addition, based on these priorities three conformance levels are defined:

- Conformance level A: all priority 1 checkpoints are satisfied.
- Conformance level AA: all priority 1 and 2 checkpoints are satisfied.
- Conformance level AAA: all priority 1, 2 and 3 checkpoints are satisfied.

New versions of these guidelines are being currently developed. The last draft of WCAG second version was released in April 2006 [6] and proposes a new guideline

¹ <http://www.w3.org/wai>

concept. This set of guidelines incorporate a new accessibility description as it defines the properties an accessible website has to accomplish. Similarly to the previous version of WCAG, each checkpoint defines three priority and analogous conformance levels. According to this description, an accessible website should fulfil these four guidelines:

- Make content **PERCEIVABLE** for any user.
- Make content and controls **UNDERSTANDABLE** to as many users as possible.
- Use **ROBUST** web technologies that maximize the ability of the content to work with current and future accessibility technologies and user agents.
- Ensure that interface elements in the content are **OPERABLE** by any user.

Searching is a significant activity when accessing the WWW. Kobayashi and Takeda [13] state that 85% of users use search engines when seeking for information on the Web. However, results are not tailored to the need of users with disabilities and they may find barriers when trying to access to the websites in results. According to a study carried out with visually impaired users by Andronico et al. [2] only 38% of them find search engines results useful while 90% of sighted users do not have any problem. This may be the reason why only 23% of visually impaired users versus the 70% of sighted users frequently use search engines. Ivory et al. [12] suggest that providing additional page features and re-ranking according to users visual abilities would be a way to improve their search experience. In this sense, this paper aims at exploring web accessibility issues on traditional Information Retrieval mechanisms such as search engines. It proposes a conceptual framework for including web accessibility measures in information retrieval processes. In addition, it presents a prototype implemented based on the proposed framework.

2 Web Accessibility as a Quality Measure

Some research aim at incorporating quality metrics in informational retrieval systems can be found in the literature such as the one presented in [20]. However, they do not consider web accessibility as a quality measure of web applications even if they take into account some usability related properties.

Many authors consider accessibility closely related to usability as they both enhance user satisfaction, effectiveness and efficiency. According to some of them, accessibility can be understood as a subset of usability. In fact, the concept of accessibility is related to the absence of physical or cognitive barriers to using the functionality implemented in a website, such as navigation, information searching, etc. Although diverse methods and tools for web usability evaluation exist [11], accessibility assessment has not been sufficiently developed, even though accessibility measurement, rating and assessment is essential in determining website quality. The lack in such accurate measures and tools for automatically calculate them may be the reason why accessibility is sometimes forgotten.

As far as standards related to quality are concerned, the ISO 9126-1 standard [10] defines six software product quality characteristics: functionality, reliability, efficiency, usability, maintainability and portability. For evaluation purposes, it also defines a quality model for software product quality and it should be used in conjunction with the ISO/IEC 14598-1 [9] providing methods for the measurement, assessment and evaluation of software product quality. When specifically refereeing to websites, specific models such as 2QCV3Q by Mich et al. [15] have been proposed. Even if they include several aspects related to both usability and accessibility, web accessibility is not considered as an important property of websites.

All the approaches for measuring the quality of software products coincide in the importance of creating adequate metrics in order to efficiently perform the quality evaluation process. The most accepted and used web accessibility metrics are qualitative ones proposed by the WAI in the WCAG 1.0 document. As previously mentioned, these metrics assign 0, A, AA or AAA value depending on the fulfilment of the WCAG 1.0 guidelines. They are not accurate enough in order to rate and classify websites according to their accessibility level. A website fulfilling only all priority 1 checkpoints would obtain the same accessibility value as another website fulfilling all priority 1 checkpoints and almost all priority 2 checkpoints: both of them would get the A level conformance. These criteria seem to be based in the assumption that if a webpage fails to accomplish one of the guidelines in a level, it is so unaccessible as if it fails to fulfil all of them. That is true for some users, but in general it is essential to have not only a reject/accept validation, but a more accurate graduation of the accessibility. Thus, as stated by Olsina and Rossi [16], defining quantitative accessibility metrics is necessary in order to overcome this situation. Moreover, they are essential in order to perform an adequate rating of websites and consequently for including web accessibility measures in informational retrieval systems. Development of adequate and accurate metrics will encourage developers to consider accessibility property as a website quality measure. Consequently, accessibility could be included in several processes such as information retrieval.

3 Related Work

There are two main components to be included in such a framework for including web accessibility metrics into informational retrieval systems. As stated in section 2, development of accurate accessibility metrics is essential as the websites should be rated and ranked based on them. These metrics will be calculated based on the results obtained after performing a comprehensive accessibility evaluation according to existing sets of guidelines. Both components should be automatized processes as the objective is to include them into other automatic processes which are the information retrieval systems. The following sections present some of the existing research works related to both components: web accessibility metrics and automatic accessibility evaluation.

3.1 Web Accessibility Metrics

Sullivan and Matson [18] evaluate eight checkpoints from WCAG 1.0. As a result, the so-called "failure-rate" is a proportion between potential errors and real errors. Therefore, the result range goes from 0 to 1. It is a naive approximation since other factors such as error impact, error nature (whether checkpoints are *errors*, *warnings* or *general warnings*) and other requirements explained in the following section are not taken into account.

$$failure_rate = \frac{real_errors}{potential_errors}$$

Hackett et al. [8] proposed the WAB formula (Web Accessibility Barrier). This formula uses as input parameters the total pages of a website, total accessibility errors as well as potential errors in a web page and error priority. However, the returned marks are not restricted to a limited range of values. Therefore, it can be useful only for ranking web pages according to their accessibility level. The drawback of this metric is that considering the result for a unique web page, it is not possible to have an accessibility reference since there are no boundaries for good or bad accessibility levels. The formula for a single web page is calculated for all WCAG checkpoints found in the page:

$$WAB_score = \sum \frac{real_errors}{potential_errors \times priority}$$

Bühler et al. [5] propose aggregation models order to adapt measurement to different disabilities groups. A simplification of that model is the following:

$$A(u) = 1 - \prod (1 - R_b S_{ub})$$

Where R is the evaluation report and S is a severity value from 0 to 1 (for each barrier type b and user group u). However, these metrics are still in a developing stage until better results are obtained. These metrics are supposed to be integrated in the web accessibility benchmarking framework defined in [17]. To our best knowledge, there is not any implemented process for automatically calculating web accessibility metrics even if there are several proposed in the literature.

3.2 Automatic Accessibility Evaluation

In recent years, a great deal of tools for automatic accessibility evaluation has been developed². Even if most of them evaluate predefined sets of general purpose accessibility guidelines such as WCAG 1.0 or Section 508, they vary in the number and type of test cases implemented. As stated in [4] evaluation results returned by different tools for the same website may vary.

In 2004, Abascal et al. [1] proposed the novel approach for automatic accessibility evaluation: separation of guidelines from the evaluation engine. The usefulness of this

² <http://www.w3.org/WAI/ER/tools/>

approach relies on its flexibility and updating efficiency. Adaptation to new guideline versions does not imply re-designing the evaluation engine but guidelines editing. The guidelines specification language is based on XML. Following this approach, in 2005, Vanderdonckt and Bereikdar proposed the Guidelines Definition Language, GDL [19] and recently Leporini et al. the Guidelines Abstraction Language, GAL [14].

4 Proposed Framework for Information Retrieval Systems

One of the objectives of this paper is to present an architecture proposal where information retrieval systems are enriched with web accessibility analysis. We propose a framework which produces results with the most suitable websites according to their content and end-user specific characteristics. In this sense, not only will be relevant the suitability of the web content returned but its accessibility level also will be taken into account. The implementation should adequately combine ranking regarding website relevance accessibility analysis. Research has been carried out on content analysis in order to rate websites but web accessibility has not been much studied in this sense. Thus, the proposed framework is as modular as possible in order to guarantee that further add-ons can be easily integrated. The architecture can be observed in Figure 1.

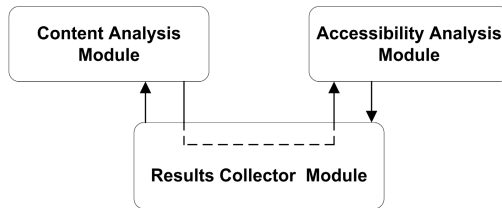


Fig. 1. Model of the proposed architecture

As can be appreciated, the architecture is composed of three independent modules:

Content Analysis Module (CAM) performs the content analysis based on information retrieval methods, techniques. It produces a list of websites rated according to their suitability for a specific query.

Accessibility Analysis Module (AAM) performs web accessibility evaluation. The previously mentioned automatic accessibility evaluation tools could be adequate for implementing this module. Integrating quantitative accessibility metrics into this module is essential in order to obtain accurately ordered lists of websites.

Results Collector Module (RCM) ensures that the information provided by the other two modules is adequately combined. Then, results according to their content and accessibility level will be produced and returned to the end-user.

Due to the modularity of the proposed framework, existing tools such as automatic accessibility tools, search engines, etc. can easily interoperate. In addition, this modular architecture will guarantee a correct independent testing for each module in order to obtain reliable systems.

5 Implementation of a Prototype

A prototype system has been implemented based on the proposed framework in order to test its usefulness. In this sense, three main tasks have been performed:

- Define accurate quantitative metrics.
- Automatic calculation of metrics.
- Integration of accessibility evaluation, metrics calculation and content analysis.

The following sections describe these tasks.

5.1 Quantitative Metrics for Web Accessibility

Accessibility problems in evaluation reports are classified by evaluation tools are classified in three main groups:

- Automatic tests (errors): these problems should not require human judgment to check their validity.
- Manual or semi-automatic tests (warnings): human judgment is necessary to check potential problems associated to particular fragments of code implementing the page.
- Generic problems: human judgment is necessary to check potential problems that cannot be associated to any code fragments; these problems arise in every web page. E.g. WCAG 1.0 14.1 checkpoint: "Use the clearest and simplest language".

The principal objectives when designing the metric have been the following:

- The value obtained by the metric should be meaningful in terms of accessibility level prediction.
- The metric should be useful for ranking web pages according to their accessibility level.

5.1.1 Requirements, Assumptions and Facts

This section defines the requirements, assumptions and facts considered when developing accessibility metrics.

Requirement 1: *The result of the metric should be normalized.*

In order to classify websites according to their accessibility a limited ratio scale from 0 to 100 is chosen so that results of the final quantitative accessibility value are expressed in a percentage scale. The closer the result of the metric is to 0 the less accessible the website is and the closer to 100 is the more accessible it is. This leads us to classify web pages according to their accessibility guidelines conformance percentage.

Requirement 2: *The metric should give one value for each accessibility attribute, as well as an overall value for each page.*

Although automatic accessibility evaluation reports returned by EvalAccess refer to WCAG 1.0 guidelines, these are mapped into WCAG 2.0 guidelines: Perceivable, Operable, Understandable and Robust³.

Apart from an accessibility quantitative value for each guideline, an overall accessibility value based on POUR guidelines is also calculated. However, metric calculation according to these guidelines is useful to get a general idea of how accessible a page is.

Assumption 1: *Besides total number of errors for each checkpoint in the web page, the metric should also take into account the total number of times each checkpoint has been tested.*

The metric should not be based on the absolute number of found errors but in the relative number of found errors in relation to the number of tested cases [18]. That is, the ratio of errors and number of tested cases. For instance, if we analyze a web page that contains 5 images without text equivalent and another one containing 10 where 5 of them have a text equivalent, the second web page should obtain better accessibility score, since the failure percentage is 100% (5 of 5) and 50% (5 of 10) respectively.

Assumption 2: *The priority of an unfulfilled checkpoint should be reflected in the final result [8].*

Priority is an ordinal-scale qualitative variable of three levels: priority 1, priority 2 and priority 3. It is stated by the WAI that priority 1 checkpoints have more impact on the accessibility level of a web page than priority 2 checkpoints and so on. Consequently, their weight in the value obtained by the metric should be different. In order to empirically tune the weights, different values are assigned to the weights in some test files with different accessibility level. The unique restriction when selecting these weights is that $1 > \text{priority1_weight} > \text{priority2_weight} > \text{priority3_weight} > 0$.

These test files have a determined failure rate. In addition, they are simple enough to manually calculate a quantitative metric. Different values were given to weights to calculate the quantitative accessibility value of each file using the metric defined next. The criterion for selecting the most appropriate weights was the similarity of the accessibility value to the failure rate on test files. The test files used are the following:

- **Low Accessibility level web page (LA):** This test file contains images without text equivalent, tables without summary, some links which open pop-up windows, auto-refreshing and wrong document language definition.
- **Accessible web page (A):** This test file contains the same potential errors but such that they do not cause any accessibility error: images have text equivalent, tables have summary, links do not open new windows, there is no auto-refresh and language is well defined.
- **Medium Accessibility level web page (MA):** Elements in this test file are the same than in Low Accessibility file but half of potential errors are actual errors.

³ <http://www.w3.org/TR/WCAG20/appendixD.html>

- **Worse than MA:** 3/4 of the previously mentioned potential errors are actual errors.
- **Better than MA:** This test file is composed of the same elements but 1/4 of them have an actual error.
- **Empty web page (E):** This test file only contains the necessary structural HTML tags without any content element.

Assumption 3: *Generic problems should not have influence on the final metric.*

When performing an automatic evaluation, all web pages get the same report of *generic problems* in order to manually check the referred checkpoints. Thus, a metric based on automatic evaluation should not take into account these checkpoints.

Fact 1: *The interval where the metric results for lowest ratios of errors and tested cases are situated has to be spread.*

We have empirically tested that in each POUR guideline, the ratio of errors over potential errors, the failure rate, tends to be very low. Thus, it is difficult to discriminate among different pages since they all get similar accessibility values. The function in Figure 3 would be an approach to the ideal hyperbole in Figure 2. In this hyperbole, the closer to 0 it is the error and tested cases ratio (E/T), the higher it will be discriminated. The advantage of this approach is that the value of x' can be empirically assigned, in order to easily control the height allocated to the failure rate E/T . This feature makes possible to increase or decrease the variability in any interval depending on the experimental results obtained modifying a and b variables. For this paper, we used $a=20$ and $b=0.3$ following an empirical approach similar to the one carried out in *Assumption 2*.

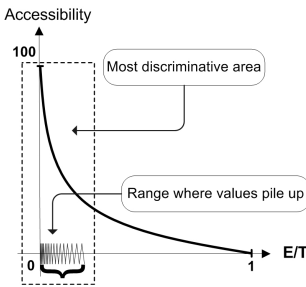


Fig. 2. Ideal hyperbole

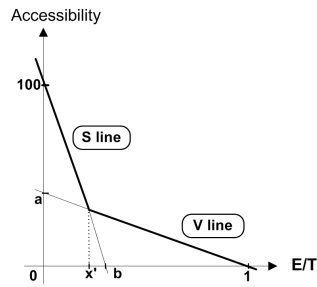


Fig. 3. An approach to the hyperbole

According to the hyperbole approach, if E/T ratio is less than the intersection point x' the accessibility will be calculated using S line. Otherwise, V line is used. x' value depends on variables a, b and tested cases.

$$x' = \frac{a - 100}{\frac{a}{T} - \frac{100}{b}}$$

x' point calculation

$$A = E \times \left(\frac{-100}{b} \right) + 100$$

S line formula

$$A = \left(\frac{-a}{T} \times E \right) + a$$

V line formula

Fact 2: *Manual tests (warnings) should be taken into account in the same way than errors.*

Our research concluded that the failure rate is highly correlated for errors and warnings when checkpoints were grouped by their guideline (POUR) and by their priority (1, 2, 3). Therefore, tested cases in warning checkpoints will fulfil the accessibility guidelines with the same ratio than their equivalent errors subgroup.

5.1.2 Variables, Constant Values and Final Metric

Table 1 contains a description of variables, constants and final values of the metric according to the requirements, assumptions and facts. Some constants are tool dependent while others are guideline-set dependent.

This metric proved to correlate positively with a research carried out by experts on Spanish universities’ websites classification according to their accessibility level as presented in [3].

Table 1. Variables, constants and metric for accessibility quantitative measurement

Variables	Description	range
E	number of accessibility errors in each checkpoint	$0-\infty$
T	number of tested cases in each checkpoint	$0-\infty$
A	variable for hyperbole approach customization (y axis)	0-100
B	variable for hyperbole approach customization (x axis)	0-1
Constants		value
N	Total number of checkpoints (EvalAccess)	44
N_{xy}	Number of checkpoints in guideline $x \in \{P, O, U, R\}$, where P stands for Perceivable, O for Operable, U for Understandable and R for Robust, and type $y \in \{error, warning\}$	
N_x	number of checkpoints in guideline $x \in \{P, O, U, R\}$	
$N_{x,error}$	total number of automatic tests	18
$N_{x,warning}$	total number of manual tests	25
$N_{P,error}$	error checkpoints in Perceivable	4
$N_{O,error}$	error checkpoints in Operable	3
$N_{U,error}$	error checkpoints in Understandable	3
$N_{R,error}$	error checkpoints in Robust	7
$N_{P,warning}$	warning checkpoints in Perceivable	11
$N_{O,warning}$	warning checkpoints in Operable	1
$N_{U,warning}$	warning checkpoints in Understandable	6
$N_{R,warning}$	warning checkpoints in Robust	7
Weights		value
k_1	priority 1 items	0.80
k_2	priority 2 items	0.16
k_3	priority 3 items	0.04
Metric		range
A_{xyz}	Accessibility of priority $z \in \{1,2,3\}$ in $x \in \{P, O, U, R\}$ guidelines and in $y \in \{error, warning\}$ type of checkpoints.	0-100
A_{xy}	Accessibility of $x \in \{P, O, U, R\}$ guidelines in $y \in \{error, warning\}$ type of checkpoints.	0-100
A_x	Accessibility of $x \in \{P, O, U, R\}$ guidelines	0-100
A	Mean accessibility value	0-100

5.2 Automatic Metric Calculation

We selected EvalAccess accessibility evaluation tool for integrating the automatic metric calculation due to its flexible architecture. In addition, the evaluation reports returned by EvalAccess are formatted based on a specific XML-Schema. Consequently, gathering of all the necessary data for metric calculation such as checkpoint type (*error* or *warning*), the times a checkpoint is tested (*T* variable), the times each test fails to be conformant with the guidelines definition (*E* variable), and its priority is straightforward. All these parameters are grouped in 2 groups (errors and warnings). Each group contains 12 subgroups classified by their priority in WCAG 1.0 (3 priorities) and their membership in the WCAG 2.0 four POUR guidelines according to the previously mentioned mapping.

Therefore, the quantitative accessibility metric takes into account the previously mentioned facts, assumptions and requirements. The quantitative accessibility metric is calculated by the following algorithm:

```

for x in each checkpoint in a guideline {P,O,U,R} loop
  for y in each type of checkpoint {error, warning} loop
    for z in each priority{1,2,3} loop
      x'=calculate_x'_point(a,b,T)
      if (E/T < x') then
        Axyz=calculate_S_line(b, E)
      else
        Axyz=calculate_V_line(a, E, T)
      end if
    end loop
  end loop
  Axy = ∑z=13 kz × Axyz ← Step a
end loop
Ax =  $\frac{\sum_y N_{xy} \times A_{xy}}{N_x}$  ← Step b
end loop
A =  $\frac{\sum_x N_x \times A_x}{N}$  ← Step c

```

In **Step a** we get all the A_{xy} values such as $A_{P,error}$. This means that we get values for *error* checkpoints in Perceivable guideline. In **Step b** an average value for each POUR guideline is calculated by weighting A_{xy} value with the number of *errors* and *warnings* in *x* guideline. Finally, we get an overall accessibility value in **Step c** weighting each POUR guideline with the number of checkpoints they contain. The last two steps take into account the number of guidelines in each category (guidelines and type) in order to distribute the weights in a well-balanced way.

5.3 Integrating Web Accessibility Evaluation and Content Relevance Analysis

In order to verify that the proposed architecture fits in a real world, a customized search engine has been developed following the described framework. In the

implemented prototype, the previously mentioned automatic evaluation tool, EvalAccess, and a conventional search engine interoperate in order to return the results of the searching tasks ranked by accessibility level and content relevance. Both services can be easily accessed and used since they are implemented as Web Services (WS). Therefore, the implemented prototype is based on the interoperation of different Web Services. The development of each module in this framework is explained below:

Content Analysis Module: Nowadays, it is unmanageable for a small company to create an outstanding information retrieval system. Large amounts of infrastructure (hardware), reliable operative systems, applications to run over servers and prepared staff is required, which implies a significant economical investment. Thus, we take advantage of the services offered by one of the most used and known search engines, Google. Google provides developers with an API⁴ which facilitates making requests to its search engine. Therefore, the function of this module is fulfilled by the techniques and methods implemented in Google since it generates an ordered list of websites based on the specified query. Thanks to the abstraction layer provided by this API, the interaction with Google Web Service is performed in a transparent way. For this reason, there is no need to directly use the SOAP protocol.

Accessibility Analysis Module: EvalAccess [1], the previously mentioned automatic accessibility evaluation tool has been integrated into the framework. It evaluates web pages according to the WCAG 1.0 guidelines and returns a machine understandable accessibility errors report formatted in XML. Web accessibility quantitative metrics can be easily applied to the information returned in this report. Since it is implemented as a Web Service, clients using SOAP protocol could get the XML report and exploit its results.

Results Collector Module: This module coordinates user requests with the different Web Services. This means coding and decoding data gathered from results of Google WS query and the XML accessibility evaluation report returned by EvalAccess WS. Then, each accessibility report is exploited in order to get all the necessary information and calculate the accessibility value based on the quantitative metrics explained in the previous section. Finally, it returns the results ordered by its accessibility.

A user web interface has been integrated into the Results Collector Module. The system shows a common search engine web interface which communicates with the core of the modules. This interface provides different options related to accessibility guidelines defined in WCAG 2.0: results re-ranking according to a value for a guideline in POUR or their average value. The results page is a list of items ordered by its accessibility as can be see in the Figure 4. Each item contains a title, a URL, its accessibility value as well as a snippet where the search keyword is contextualized.

The overall latency when evaluating web accessibility and calculating the metric is higher than in traditional search. However, the proposed model is still valid since it is useful in order to remove existing barriers.

⁴ <http://www.google.com/apis/>

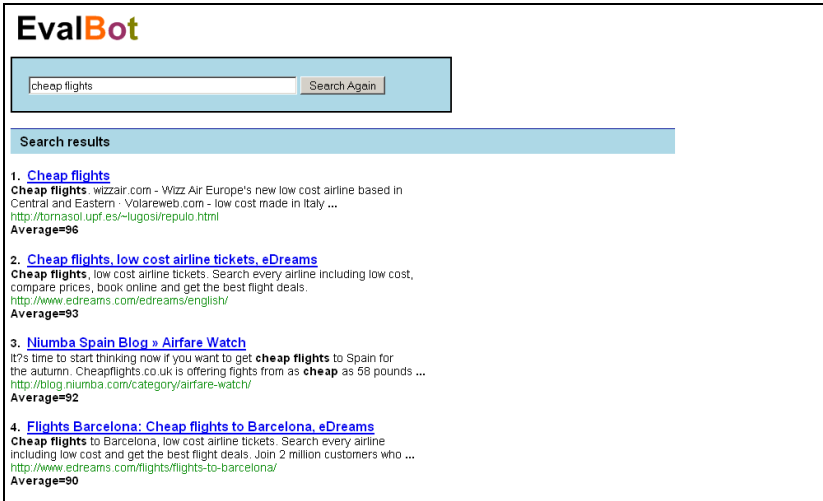


Fig. 4. Search result page

6 Results and Discussion

Several tests have been carried out in order to evaluate the performance of this framework. One testing case is presented in this section for discussing the results obtained by the implemented system. Firstly, a searching query was defined. In this case, this query was: "cheap flights". Then, this query has been introduced to Google search engine as well as to our Customized Search Engine (CSE). Table 2 shows the first 10 results returned by the system and their order in the results list. In addition, errors yielded by automatic evaluation are grouped by their priority.

Table 2. Results obtained by Google and CSE. Automatic evaluation results of EvalAccess tool grouped by their priorities (P1 - Priority 1 errors, P2 - Priority 1 errors, P3 - Priority 3 errors) are also provided.

URL	Google	CSE	P1	P2	P3
www.cheapflights.co.uk	1	5	0	15	265
www.easyjet.com	2	2	0	29	91
www.ryanair.com/site/EN	3	6	45	230	995
www.skyscanner.net	4	8	0	9	69
www.cheapflights.com	5	3	0	7	186
www.flightline.co.uk	6	4	0	6	128
www.flybmi.com/bmi	7	1	0	2	60
www.bmibaby.com	8	10	2	75	169
www.aerlingus.com	9	-	0	5	100
www.openjet.com	10	-	0	1	3
www.bargainholidays.com	-	7	29	67	262
www.cheapestflights.co.uk	-	9	8	143	198

It can be appreciated in Table 2 that the results obtained by both search engines differ, as two websites included in Google result (www.aerlingus.com and www.openjet.com) are not in CSE results. Other two websites have been included in CSE results instead of these two websites (www.aerlingus.com and www.openjet.com). There are discrepancies between Google API and Google.com results due to the fact that apparently queries are made against different indexes. It could be discussed whether the trade-off of content ranking versus the accessibility ranking is really worthy. This is an approach to the conceptual model explained in Section 4 and the evidence that this model works. However, query results could be listed according to Google's results and each result could be marked with its accessibility value without re-ranking it so the user would decide to browse it or not as proposed in [12].

Web accessibility analysis of these websites has been performed by using the service provided by one of the automatic accessibility evaluation tool, EvalAccess. Columns P1, P2 and P3 show the number of automatically detected accessibility errors in the returned websites. The first website returned by Google (www.cheapflights.co.uk/) is also included in CSE result but in the fifth position. Evalbot returns as the first result the website of www.flybmi.com/bmi/. According to the data, the result obtained by the CSE has less accessibility problems than the one obtained by Google, since www.cheapflights.co.uk website errors are P1-0, P2-15, P3-265 whereas www.flybmi.com/bmi website errors are P1-0, P2-2, P3-60. Then, end-users would easier access to the result obtained by the CSE. However, some contradictory data can be found in these results. For example, www.cheapflights.com website accessibility errors are P1-0, P2-7, P3-186 and www.flightline.co.uk website accessibility errors are P1-0, P2-6, P3-128. According to this data, the www.flightline.co.uk website would have less accessibility barriers than the other one. But the CSE results show that it is placed behind in the results list. As explained in **Assumption 1**, this is due to the accessibility quantitative metric is based in the percentage of found errors divided by the number of tested cases (potential errors).

7 Conclusions and Future Work

This paper proposes a framework for incorporating web accessibility information into information retrieval systems. This significantly improves user's satisfaction when searching for information in the WWW, as they could obtain search results ordered by content relevance as well as accessibility level.

The necessity of a quantitative metric for web accessibility assessment has been demonstrated in this paper. If accurate discrimination among web pages wants to be done, these measures are key factors. The proposed metric aims at being a general approach to accessibility awareness in information retrieval processes. The metric does not take into account specific users grouped by disabilities (hearing, visually or physically impaired). However, besides an average value, POUR values are given because the system performs a mapping between WCAG 1.0 and WCAG 2.0 draft.

A prototype based on the proposed framework has also been developed. It enables users with disabilities making search queries which return websites listed according to both Google information retrieval criteria and accessibility criteria. The measure of

web accessibility is based on EvalAccess, an automatic accessibility evaluation Web Service which returns accessibility reports formatted in XML. This feature facilitates applying quantitative accessibility metrics to the evaluation results.

The most significant disadvantage when using the prototype is the increase in the response time comparing with the original search engine. This latency could discourage users to performing search tasks with this prototype. We are currently working on the next version of the framework which will improve the response time.

Acknowledgements

Markel Vigo's work is funded by the Department of Education, Universities and Research of Basque Government.

References

1. Abascal, J., Arrue, M., Fajardo, I., Garay, N., Tomás, J.: Use of Guidelines to automatically verify web accessibility. *International Journal of Universal Access in the Information Society* 3(1), 71–79 (2004)
2. Andronico, P., Buzzi, M., Castillo, C., Leporini, B.: Improving search engine interfaces for blind users: a case study. *International Journal of Universal Access in the Information Society* 5(1), 23–40 (2006)
3. Arrue, M., Vigo, M., Abascal, J.: Quantitative Metrics for Web Accessibility Evaluation. In: Lowe, D.G., Gaedke, M. (eds.) *ICWE 2005*. LNCS, vol. 3579, Springer, Heidelberg (2005)
4. Brajnik, G.: Comparing accessibility evaluation tools: a method for tool effectiveness. *International Journal of Universal Access in the Information Society* 3(3-4), 252–263 (2004)
5. Bühler, C., Heck, H., Perlick, O., Nietzjo, A., Ullveit-Moe, N.: Interpreting Results from Large Scale Automatic Evaluation of Web Accessibility. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A.I. (eds.) *ICCHP 2006*. LNCS, vol. 4061, pp. 184–191. Springer, Heidelberg (2006)
6. Caldwell, B., Chisholm, W., Slatin, J., Vanderheiden, G. (eds.). (2006, April 27) *Web Content Accessibility Guidelines 2.0*. (Working Draft). <http://www.w3.org/TR/WCAG20/>
7. Chisholm, W., Vanderheiden, G., Jacobs, I. (eds.) *Web Content Accessibility Guidelines 1.0*. (May 5, 1999) <http://www.w3.org/TR/WAI-WEBCONTENT/>
8. Hackett, S., Parmanto, B., Zeng, X.: Accessibility of Internet websites through time. In: *Proceedings of 6th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 32–39 (2004)
9. International Organization of Standardization (ISO), *Information Technology - Software Product Evaluation (ISO 14598)*. Geneva, Switzerland (1999)
10. International Organization of Standardization (ISO), *Software Engineering - Product Quality - Part1: Quality Model (ISO 9126-1)*. Geneva, Switzerland (2001)
11. Ivory, M.Y., Hearst, M.A.: The state of art in automating usability evaluations of user interfaces. *ACM Computing Surveys* 33(4), 470–516 (2001)
12. Ivory, M.Y., Yu, S., Gronemyer, K.: Search result exploration: a preliminary study of blind and sighted users' decision making and performance. In: *CHI Extended Abstracts*, pp. 1453–1456 (2004)

13. Kobayashi, M., Takeda, K.: Information Retrieval on the Web. *ACM Computing Surveys* 32(2), 144–173 (2000)
14. Leporini, B., Paternò, F., Scordia, A.: Flexible tool support for accessibility evaluation. *Interacting with Computers* 18(5), 869–890 (2006)
15. Mich, L., Franch, M., Gaio, L.: Evaluating and Designing Web Site Quality. *IEEE Multimedia* 10(1), 34–43 (2003)
16. Olsina, L., Rossi, G.: Measuring Web Application Quality with WebQEM. *IEEE Multimedia* 9(4), 20–29 (2002)
17. Snaprud, M.H, Ulltveit-Moe, N., Pillai, A.B., Olsen, M.G.: A Proposed Architecture for Large Scale Web Accessibility Assessment. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A.I. (eds.) *ICCHP 2006*. LNCS, vol. 4061, pp. 234–241. Springer, Heidelberg (2006)
18. Sullivan, T., Matson, R.: Barriers to use: usability and content accessibility on the Web's most popular sites. In: *Proceedings of the ACM Conference on Universal Usability 2000*, pp. 139–144 (2000)
19. Vanderdonckt, J., Bereikdar, A.: Automated Web Evaluation by Guideline Review. *Journal of Web Engineering* 4(2), 102–117 (2005)
20. Zhu, X., Gauch, S.: Incorporating quality metrics in centralized/distributed information retrieval on the World Wide Web. In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 288–295 (2000)

Fixing Weakly Annotated Web Data Using Relational Models

Fatih Gelgi, Srinivas Vadrevu, and Hasan Davulcu

Department of Computer Science and Engineering,
Arizona State University,
Tempe, AZ, 85287, USA
{fagelgi,svadrevu,hdavulcu}@asu.edu

Abstract. In this paper, we present a fast and scalable Bayesian model for improving *weakly annotated data* – which is typically generated by a (semi) automated information extraction (IE) system from Web documents. Weakly annotated data suffers from two major problems: they (i) might contain incorrect ontological role assignments, and (ii) might have many missing attributes. Our experimental evaluations with the TAP and RoadRunner data sets, and a collection of 20,000 home pages from university, shopping and sports Web sites, indicate that the model described here can improve the accuracy of role assignments from 40% to 85% for template driven sites, from 68% to 87% for non-template driven sites. The Bayesian model is also shown to be useful for improving the performance of IE systems by informing them with additional domain information.

Keywords: Weakly annotated data, information extraction, classification, Bayesian models.

1 Introduction

Recent years have witnessed a huge data explosion on the Web. All kinds of commercial, government and scientific organizations have been publishing their data to enable better information sharing. However, heterogeneity at the presentation, schema and instance levels makes it extremely difficult to find and relate information from different sources. Recently, there has been some ground breaking work on (meta) data extraction from template driven Web sites using semi-automated techniques [1] and ontology-driven automated data extraction techniques [2] from text. And also, some work exists on fully automated techniques for extracting (meta) data from data rich segments of Web pages [3,4,5].

In this paper, our focus will be on improving *weakly annotated data* (WAD) which is typically generated by a (semi) automated information extraction (IE) system from the Web documents. In WAD, *annotations* correspond to ontological role assignments such as *Concept*, *Attribute*, *Value* or *Noise*. WAD has two

major problems; (i) might contain incorrect role assignments, and (ii) have many missing attribute labels between its various entities.

We will use the Web pages in Figure 1 to illustrate WAD that might be extracted using an IE algorithm such as [31,5]. Each of these pages presents a single instance of the ‘Digital Camera’ concept. In Figure 1(a), attributes such as ‘storage media’, and values such as ‘sd memory card’ have uniform and distinct presentation. However, for an automated system it would be extremely difficult to differentiate the ‘storage media type’ label as an attribute and ‘sd secure digital’ as its value due to their uniform presentation in Figure 1(b). On the other hand, in Figure 1(c) the attribute ‘storage media’ does not even exist, but only its value ‘sd memory card’ has been reported.

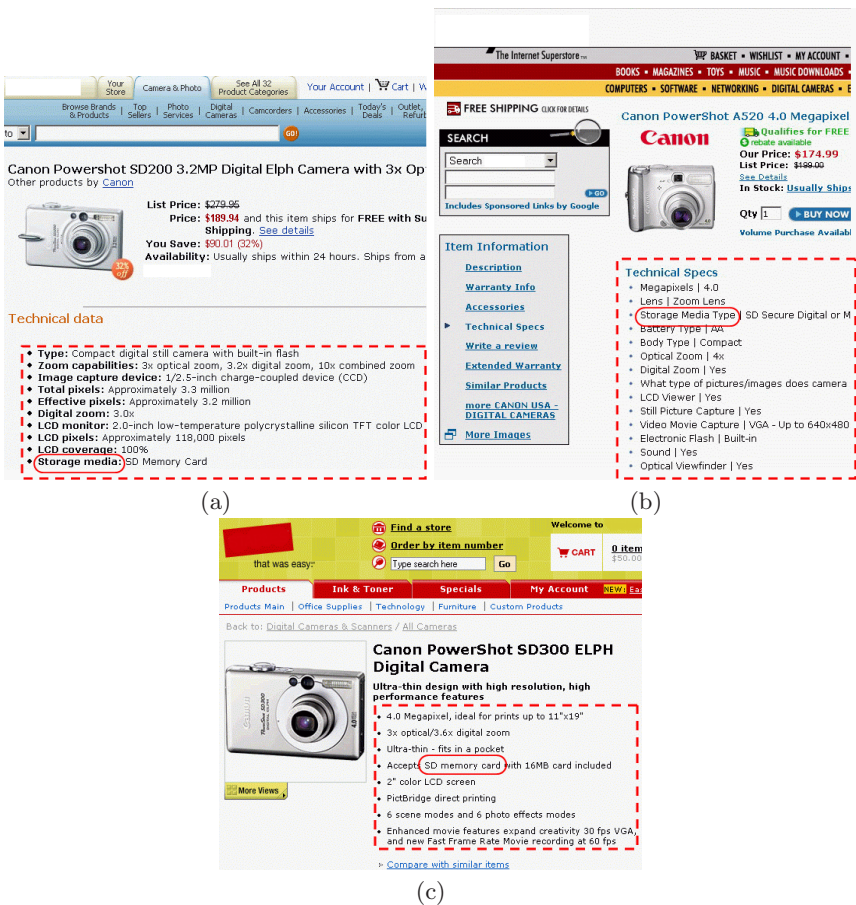


Fig. 1. The instance pages for Canon digital camera from three different Web sites. Digital camera specifications are marked by dashed boxes in each page. In page (a), attributes are explicitly given as bold whereas in (b) they are not obvious. On the other hand, the attributes are altogether missing in (c).

Florescu [6] indicates that schemas in unstructured data can be very rich and they might be difficult to model using DTDs or ER models. Schemas can be derived from the data instead of driving the data generation, or schemas can be a posteriori overlaid existing data. Motivated by the idea, we first extract a *relational graph* from WAD to capture the association strength between labels in the given document collection. This method is based on the identification of the correlation between labels as co-occurrence constraints emphasized in [6]. Then, we build a domain specific probabilistic model that utilizes the extracted relational graph to improve WAD by automatically **correcting the role assignments of the labels** and **discovering their missing attributes**. We also show that the boost up in the accuracy of the WAD provides additional information for the IE systems to improve their performance. Note that since we are working on a huge data set such as the Web, we exploit linear-time algorithms to ensure the speed and the scalability of our methods while not losing much from the quality.

We formulate the role assignment problem as a classification problem. A Bayesian model is used due to the robustness of Bayesian models on classification tasks [7] with large number of features. Since discovery of the Bayesian network dependencies on data is a hard problem [8], we stick to the independence assumption which also ensures the scalability of the proposed model for the Web data.

The distinguishing feature of our model from the standard Bayesian models is the preservation of the structure of the *relational graph* (see Section 2 for details) by incorporating the edge probabilities. Furthermore, the number of features (i.e. tagged labels) are not fixed, as required by conventional classification techniques.

As stated in [9], naive Bayes classifiers perform well on strongly annotated data, i.e., correctly tagged training data, but they are very sensitive to redundant and irrelevant features. Therefore, the excessive amount of irrelevant features and incorrect role assignments inherent in WAD would render a naive Bayes classifier to be ineffective. A subsection is also included to discuss the weaknesses of naive Bayes.

Probabilistic Relational Models (PRMs) [10] are powerful methods to learn the underlying structure of relational data. However, PRMs and other classification approaches on relational data such as [11] assume strongly annotated data, and their scalability is a problem which makes it inappropriate for the Web data.

2 System Overview

In our framework, we assume each label is tagged with one of the four ontological roles listed below;

- *Concept* (C): A concept defines a category or a class of similar items. E.g., ‘books’ and ‘digital camera’ are some of the concepts in the shopping domain.
- *Attribute* (A): An attribute is a property of an instance or a concept. E.g., ‘storage media’ is an attribute of the ‘canon powershot sd200’ instance and the ‘digital camera’ concept.

- *Value (V)*: A value is a label that provides the value information for an attribute of a certain concept or an instance of a concept. E.g., ‘storage media’ attribute of the ‘canon powershot sd200’ instance has the value ‘sd memory card’.
- *Noise (N)*: Any label that does not have any of the above ontological roles are assigned to be noise. For example, some of the labels in headers, footers or navigation aids, such as ‘back to’ shown in Figure II(c) could be annotated as noise.

We assume that we can gather “sufficient statistics” for WAD through a collection of domain specific Web sites. From the automatically extracted data we generate a *relational graph* of the domain where nodes correspond to the labels with assigned roles, and the edges correspond to association strengths between nodes. These annotated labels and relations between them are assumed to be the output of automated IE systems such as RDF files. Such a graph would capture the global occurrence statistics of the labels and their associations within a domain.

In the next phase, for each label in a given Web page, we run a Bayesian classifier that utilizes all the labels in that Web page as its *context* to identify the best role for that label. The advantage of our probabilistic model over a naive Bayes classifier will be discussed in the next section. We refer to our probabilistic model as the *Bayesian classifier* in the rest of the paper.

We will briefly explain how the system operates on the example given in Figure

II A fragment of the corresponding relational graph is depicted in Figure II. Consider the label ‘storage media’ marked in Figure II. Based on our assumption of “sufficient statistics”, a collection of Web pages such as those in Figure II(a), would yield a strong association between ‘canon sd200’ as an object and ‘storage media’ as an attribute. Whereas, the incorrect annotation, extracted from Figure II(b) would yield a weak association between ‘canon a520’ as an object and the ‘storage media’ as a value. Hence, the Bayesian classifier presented here would be able to re-assign the attribute role to the ‘storage media’ label by using the statistics in the relational graph within its context. Similarly, the ‘storage media’ attribute of the ‘sd card’ value which is missing in Figure II(c), could be inferred by utilizing its context and the model.

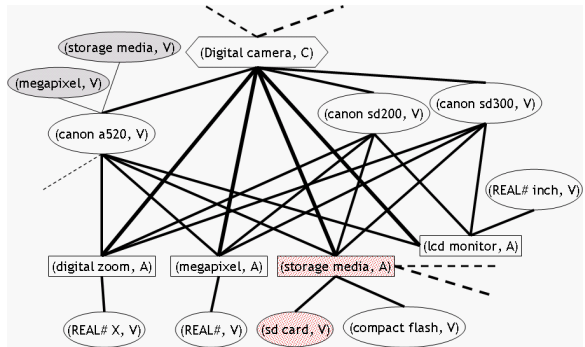


Fig. 2. A fragment of the relational graph for the Shopping domain is shown. Each node is composed of a $(label, role)$ pair. The thickness of an edge is proportional to the *association strength* between its nodes.

3 Probabilistic Model

In this section, we will first present the notations used for the formal description of the model. We will formally define the problem of re-annotating the labels of a Web page and present the probabilistic algorithm. Next, we will define the missing attribute inference problem and propose a solution. We will also explain some of the implementation issues and complexity analysis.

The notation used for formalization is given as follows:

- The set of all labels in the domain is denoted as \mathcal{L} .
- The **ontological roles** \mathcal{R} is the set of *Concept*, *Attribute*, *Values* or *Noise*. Formally, $\mathcal{R} = \{C, A, V, N\}$.
- A **term** is a pair $\langle l, r \rangle$ composed of a label l and a role $r \in \mathcal{R}$. In other words, terms are tagged labels in the Web pages. Each label in a Web page is assumed to be tagged with only one of the given ontological roles above.
- In this setting, we consider all the labels in each Web page are tagged with roles, hence we define a **Web page** to be a vector of its terms. Formally, assuming m labels in the Web page \mathcal{W} ; $\mathcal{W} = \{\langle l_1, r_1 \rangle, \langle l_2, r_2 \rangle, \dots, \langle l_m, r_m \rangle\}$.
- The **relational graph** \mathcal{G} is a weighted undirected graph where the nodes are the terms in the domain, and the weights on the edges represent the *association strength* between the terms.
- In our framework, the **context** of a label $l \in \mathcal{L}$ in a Web page \mathcal{W} is the Web page \mathcal{W} itself.

The nodes in \mathcal{G} denote the labels with their ontological roles and the edges denote the association strengths between the annotated labels. Node weights are initialized as the counts of the corresponding terms and the edge weights are the counts of the corresponding edges in the document collection. Formally, w_{ij} which is the weight between the terms i and j is initialized as the number of times the edge (i, j) appeared in the entire domain. Similarly, w_i represents the weight of the node i and initialized as the occurrence of the corresponding term in the domain, i.e., term count. Note that the edges are undirected since association strength between labels is a bi-directional measure.

3.1 Label Role Inference

The role of a label depends on its *context*. This context of a label is intuitively defined to be its own Web page. The problem of role assignment for each label can now be formally defined as follows;

Definition 1. *Given a Web page \mathcal{W} , the probability of a term $\langle l, r \rangle$ where $l \in \mathcal{L}$ and $r \in \mathcal{R}$ is $P(\langle l, r \rangle | \mathcal{W})$.*

This corresponds to the probability of the classification of l as r to be correct. Then, the role with the maximum probability will be the role assignment for the particular label l that is,

$$\arg \max_r P(\langle l, r \rangle | \mathcal{W}). \tag{1}$$

For simplicity we use the *naive assumption* which states that,

Assumption 1. *All the terms in \mathcal{G} are independent from each other but the given term $\langle l, r \rangle$.*

Furthermore, we only utilize the first order relationships of a term in its context, i.e, neighbors of the term in \mathcal{G} . One can easily extend the model for higher order relationships however the trade-off is the higher complexity which is undesired for Web data.

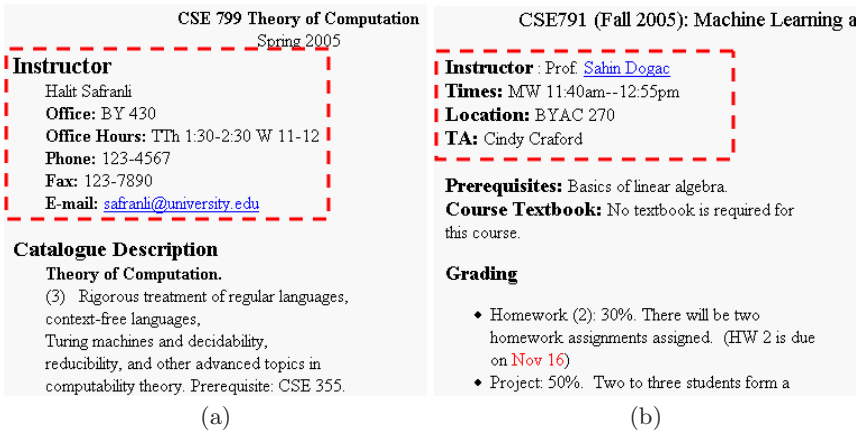


Fig. 3. Ambiguity of the role of the label ‘Instructor’ in the Courses domain. (a) is its rare occurrence as a concept, and (b) shows its common one as an attribute.

During the role assignment probability calculation of a term, since we would like to utilize only the label’s context we also assume,

Assumption 2. *The prior probabilities of all the roles of a label l are uniform.*

Note that, the priors of the roles of the labels other than l in the Web page are their support values as determined by their frequencies. To motivate the idea, consider the label ‘Instructor’ in Figure 3. ‘Instructor’ rarely occurs as a concept in the *Courses* domain. Its attributes such as ‘Phone’, ‘Fax’, ‘E-mail’ are also presented in Figure 3(a). However, ‘Instructor’ usually appears as an attribute of a course in other documents as shown in Figure 3(b). Thus, the prior probability, i.e., $P(\langle Instructor, A \rangle) \gg P(\langle Instructor, C \rangle)$, might strongly bias the role assignment towards its more common role. This would yield an incorrect tagging for the label as an attribute in Figure 3(a).

Now, with the above assumptions, we can state the following theorem.

Theorem 1. Let $\mathcal{W} = \{t_1, t_2, \dots, t_m\}$. Then,

$$\arg \max_r P(\langle l, r \rangle | \mathcal{W}) = \arg \max_r \prod_{i=1}^m P(\langle l, r \rangle | t_i). \quad (2)$$

Proof. Let $t = \langle l, r \rangle$. By Bayes's rule,

$$P(t | \mathcal{W}) = P(t | t_1, t_2, \dots, t_m) = \frac{P(t_1, t_2, \dots, t_m | t) P(t)}{P(t_1, t_2, \dots, t_m)}.$$

Using the independence assumption,

$$= \frac{\prod_{i=1}^m P(t_i | t) P(t)}{\prod_{i=1}^m P(t_i)}$$

Again using Bayes's rule,

$$= \frac{\prod_{i=1}^m P(t | t_i) P(t_i)}{P(t)^m} \cdot \frac{P(t)}{\prod_{i=1}^m P(t_i)} = \frac{\prod_{i=1}^m P(t | t_i)}{P(t)^{m-1}}$$

By Assumption 2, $P(t)^{m-1}$ will be constant. That is,

$$\arg \max_r P(t | \mathcal{W}) = \arg \max_r \prod_{i=1}^m P(t | t_i).$$

□

As shown in Figure 2, a conditional probability such as $P(t | t_i)$ depends on the association strength between the terms t and t_i in the relational graph \mathcal{G} . That is, $P(t | t_i) = \frac{P(t, t_i)}{P(t_i)} = \frac{w_{tt_i}}{w_{t_i}}$ by Bayes's rule where w_{tt_i} is the weight of the edge (t, t_i) and w_{t_i} is the weight of the node t_i . Our probability model is based on the methodology of association rules [12]. Hence, the initialization for the above conditional probabilities is defined analogous to $P(t | t_i) \equiv \text{Confidence}(t_i \rightarrow t)$ [13]. This formulation is consistent with Assumption 2 since it is independent from the prior, $P(t)$.

3.2 Missing Attribute Inference

In WAD, most of the attribute labels are missing, especially in the non-technical domains and non-template driven Web sites. Discovering missing relations is one of the crucial tasks during automated meta-data extraction. Our probabilistic model can also be tailored to infer some of the missing attributes.

Suppose two related entities have a missing attribute in a Web page. The first entity may be either a concept or an instance of a concept whereas the second one may be a value or a set of values.

Definition 2. Given two related entities e_1 and e_2 in a Web page, the probability of a label $l \in \mathcal{L}$ to be the attribute between them is $P(\langle l, A \rangle | \mathcal{S})$ where $\mathcal{S} = e_1 \cup e_2$.

Thus, the missing attribute can be inferred by the following formula,

$$\arg \max_{l \in \mathcal{L}} P(\langle l, A \rangle | \mathcal{S}). \quad (3)$$

And, with the same assumptions described above,

Theorem 2. *Let e_1 and e_2 be two entities and $\mathcal{S} = e_1 \cup e_2$. Then,*

$$\arg \max_{l \in \mathcal{L}} P(\langle l, A \rangle | \mathcal{S}) = \arg \max_{l \in \mathcal{L}} \prod_{t \in \mathcal{S}} P(\langle l, A \rangle | t). \quad (4)$$

Proof. Follows from the same methodology in the proof of Theorem [1](#).

□

3.3 Complexity Analysis

Assuming there are n terms and m associations between them, the initialization phase requires only $O(n + m)$ time by utilizing an adjacency list for the relational graph. It is also assumed that it takes $O(1)$ time to map a term to its corresponding node utilizing a hash table. For the rest of the analysis we will assume that we are working on a very large n and $m = O(n)$, i.e., the average number of relations for each label is constant, which is expected from the Web data. Hence, this phase has $O(n)$ time and memory complexity.

For a label with a particular role in a Web page, all conditional probabilities are calculated depending on the other labels in its Web page. That amounts to $O(|\mathcal{W}|)$. Considering all the labels and all roles the total probability calculations for each Web page will be $O(|\mathcal{R}||\mathcal{W}|^2)$. We only need the memory to store the role probabilities for each label in the Web page, that is $O(|\mathcal{W}|)$. Supposing we have p Web pages in a large data set, $O(|\mathcal{R}||\mathcal{W}|^2)$ is also considered to be constant since the size of a Web page is limited and independent from how large p is. In other words, classification phase is $O(p)$ time and memory. Finally, the overall system has $O(n + p)$ time and memory complexity.

While inferring missing attributes, for each $\langle \text{entity}, \text{attribute}, \text{entity} \rangle$ triple, it is not necessary to explore all the attribute labels. Instead, we only check the labels which are related to both entities which takes only constant time (due to the assumption of constant number of relations for each label indicated above). That makes the complexity of attribute inference $O(|\mathcal{W}|)$ for each page \mathcal{W} . Similarly, the constant bound on $O(|\mathcal{W}|)$ will yield execution time for entire data set to be $O(p)$. This phase requires only constant memory.

For the entire system, log-probabilities generate a slight overhead that does not change the complexity. In conclusion, the presented probabilistic framework is linear in terms of both the number of terms and the number of web pages thus yielding a fast and scalable model.

3.4 Discussion on Naive Bayes

For the WAD, naive Bayes has entirely different characteristics from our Bayesian classifier. The formulation for the naive Bayes classifier is;

$$\arg \max_r P(\langle l, r \rangle | \mathcal{W}) = \arg \max_r \left[\prod_{i=1}^m P(t_i | \langle l, r \rangle) \right] P(\langle l, r \rangle). \quad (5)$$

Naive Bayes uses the reverse conditional probability;

$$P(t_i | \langle l, r \rangle) = \frac{P(t_i, \langle l, r \rangle)}{P(\langle l, r \rangle)}. \quad (6)$$

This violates Assumption 2 since $P(t_i | \langle l, r \rangle)$ is conditioned on $\langle l, r \rangle$. Hence, it would not be able to reason with the context of the label alone – instead relies on the prior probabilities $P(\langle l, r \rangle)$ which yields substantially lower performance as illustrated in our experimental results.

4 Experiments

The descriptions of the three data sets used in the experiments are as follows:

1. **TAP Dataset:** Stanford *TAP Knowledge Base 2* [1] data set. The selected categories alone comprise 9,068 individual Web pages as shown in details in Table 1.
2. **CIPS Dataset:** We prepared a data set which is composed of *faculty*, *course* home pages, *shopping* and *sports* Web pages consisting of 225 Web sites and more than 20,000 individual pages. The computer science department Web sites are *meta-data-driven*, i.e., they present similar meta-data information across different departments. Shopping and sports are some popular attribute rich domains.

Table 1. TAP data set used in experiments

Web sites	# of Web pages	Average # of labels per page
AirportCodes	3829	34
CIA	28	1417
FasMilitary	362	89
GreatBuildings	799	37
IMDB	1750	47
MissileThreat	137	40
RCDB	1637	49
TowerRecords	401	63
WHO	125	21
Overall	9068	200

3. RoadRunner Dataset: [3](#)'s data set comprised of 200 pages in 10 categories.

To overcome the lack of statistics for continuous values, we preprocessed the common data types of values such as percentage, dates, numbers etc. using simple regular expressions. In each experiment, we use the entire data set as training set to exploit global information of the domain, i.e., relational graph of the domain. Context based role inference, i.e., the Bayesian model is based on that relational graph.

The reported accuracies are measured according to the following formula:

$$Accuracy = \frac{\# \text{ of correct annotations}}{\# \text{ of total annotations}}$$

4.1 Experiments with the TAP Data Set

To test our probabilistic method, we converted RDF files in the TAP knowledge base into triples, then we applied distortions to obtain the inputs. For synthetic data, we considered real world situations and tried to prepare the input data as similar as possible to the data on the Web. There are two types of distortion: *deletion* and *role change*. Setting the distortion percentages for both deletion and role change first, we used the percentages as distortion probabilities for each tagged label in the Web page in our random distortion program.

Over TAP data set, we prepared test cases for three kinds of distortions. In the first one, we only applied deletion with different percentages. In the second, similarly we only applied role changing. And the last one is the mixture of the previous two; we applied the same amount of deletions and role changes.

The evaluations of TAP data set are done in automated way assuming the original TAP data set has the correct annotations. We present our results in two categories: (1) individual Web sites and (2) mixture Web sites. For both categories, the Bayesian classifier performed with 100% accuracy for distorted data with only deletions. The reason is, deletion does not generate ambiguity since the initial data is unambiguous. Thus, we found unnecessary to include them in the tables and figures. As a baseline method, we used a naive Bayes classifier.

Experiments with the individual Web sites provided us encouraging results to start experiments with mixture of Web sites as shown in Figure [4](#). The figure displays the final accuracies of Web sites which are initially distorted with {5, 20, 40, 60} percent role changes. Overall results show that there is a huge gap between the Bayesian and the naive Bayes classifiers. Even for 60% role changes the Bayesian classifier performed with more than 85% accuracy. The performance is usually better with the Web sites containing large number of Web pages due to the high consistency and regularity among the Web pages. Another factor is the size of the tagged label set in the Web pages. The larger the set, the more difficult to keep the context concentrated on the related roles in ambiguous

data. That played the most important role for the low performance with *CIA* and *FasMilitary* Web sites and, high performance with *WHO* and *AirportCodes*. On the other hand, naive Bayes slightly improved the initial accuracies as shown in Figure 4(b).

In the experiments with mixture of Web sites, we tested the Bayesian classifier with $\{5, 20, 40, 60, 80\}$ percent role changes and $\{(5,5), (15,15), (25,25), (35,35), (45,45)\}$ percent (deletion, role change) distortions. Figure 5 shows the performance of the Bayesian classifier in terms of concept, attribute and value accuracy, and the final Web site accuracies respectively. The overall performance for the mixture Web sites is slightly lower than the individual Web sites due to the fact that mixture Web sites initially have some ambiguities. Similar to the results in the individual Web sites, naive Bayes has not been successful to increase the accuracies of annotations in the mixture ones. The comparison charts in Figure 5(a) and (b), clearly presents the significant difference between the naive Bayes and our Bayesian classifier.

In conclusion, the overall results show that our Bayesian model can recover the TAP data up to 80% even with 60% and (35%,35%) distortion. The results are not surprising since the data set is simple template driven and also the relations are not complicated. In addition, the experimental results strongly support our claims about the unreliability of prior probabilities and weakness of naive Bayes given in Section 3. Verifying the robustness of the system with template driven Web sites, next we will give the experimental results with a non-template driven data set.

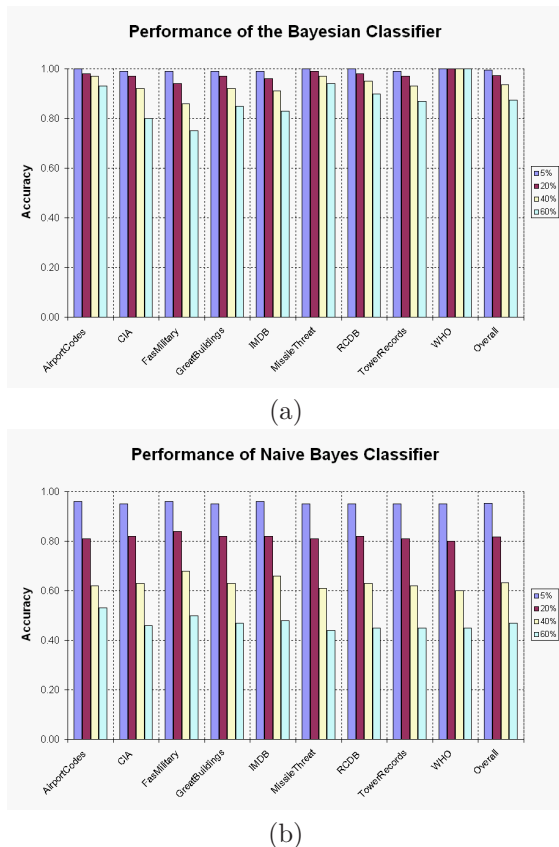


Fig. 4. Performance of the Bayesian and the naive Bayes classifiers for label accuracies in individual Web sites are shown

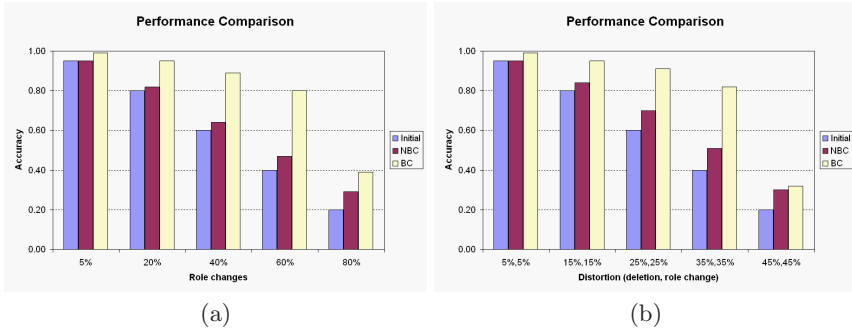


Fig. 5. The comparison of the Bayesian (BC) and the naive Bayes (NBC) classifiers

4.2 Experiments with the CIPS Dataset

Recalling the motivating examples, one can observe that labels are highly ambiguous at both the syntactic and the semantic levels. Many labels have different roles in different Web pages depending on the context.

In this experiment, we used the semantic partitioner [5] to obtain initial annotations of the labels from Web pages. The semantic partitioner system transforms a given Web page into an XML-like structure by separating and extracting its meta-data and associated data. For the evaluations, we created a smaller data set containing randomly selected 160 Web pages from each domain. We divided samples into 4 groups with 40 pages from each category, and each group is evaluated by a non-project member computer science graduate student. The overall accuracy of each category provided in Figure 6 is based on the total accuracy of these sample documents.

The accuracies of initial bootstrapping by the semantic partitioner and of the corrected data by the Bayesian classifier is presented in Figure 6. Since the faculty and course categories are the sub-categories of computer science departments, they presented very similar characteristics as shown in Figure 6(a) and (b). The overall accuracies have been increased roughly from 71% to 89% – a 18% boost. In the shopping domain, although the increment of the value accuracy is similar to the previous two, the overall accuracy jumped up from 69% to 93% – a

Table 2. The results of missing attribute inference over CIPS data set

Category	CIPS data set			Sampled data set			
	# of sites	# of pages	# of infer.	# of pages	# of infer.	# of crr.	%
Faculty	60	7617	1232	160	38	26	0.68
Course	60	4228	1009	160	49	35	0.71
Shopping	47	3361	4523	160	198	153	0.77
Sports	58	5805	2877	160	82	51	0.62
Total	225	21011	9621	640	367	265	0.72

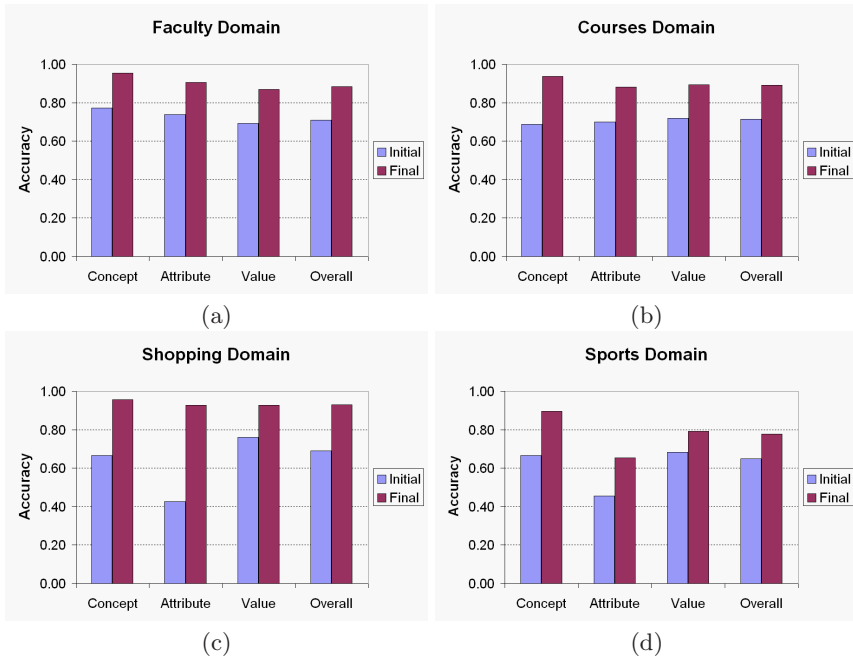


Fig. 6. Performance of the Bayesian classifier in CIPS data set

24% boost. Mistagged concepts and attributes have been corrected with very high accuracy. Initial taggings for meta-data (concepts + attributes) has the lowest accuracy among those four categories since the labels are presented more uniformly than in the other categories as illustrated in Figure 11(b). Fortunately, the labels used for the meta-data in the domain are more common yielding strong statistic thus high recovery accuracy. Conversely, the overall accuracy of the sports category is the lowest among four categories since the jargon of the domain varies much more than the other domains.

The result of the missing attribute inferences in CIPS dataset is presented in Table 2. For the overall data set 9641 attributes have been inferred, 72% of which are correct according to the statistics of the sample data set. The accuracy of the missing attribute inference in the shopping category is slightly better than in the other categories due to the usage of more common meta-data labels as mentioned above. As an example for missing attribute inference, the attributes ‘megapixel’, ‘zoom’, ‘storage media’ and ‘lcd’ are correctly inferred in the web page in Figure 11(c).

4.3 A Case Study with IE Systems

This experiment was conducted to illustrate the utility of the probabilistic model for improving the performance of an IE system. With the permission of the

authors, we modified the semantic partitioner code. The original semantic partitioner operates by first identifying approximate tandem repeats of presentation information corresponding to the labels within a Web page. Then, it groups and annotates the labels into XML-like hierarchical structures.

The annotations by the semantic partitioner were used as input for the Bayesian classifier. Next, the semantic partitioner was modified to utilize the inferred role probability distributions, in addition to the presentation information so that it can identify more tandem repeats and extract data even in the presence of irregularities. For example in Figure 1(b), the original semantic partitioner fails to distinguish the attributes and values in the ‘Technical Specs’ area since they are presented similarly. However, the modified semantic partitioner recognizes the roles of these labels as attributes and values, and correctly identifies the repeating sequence of attribute-value pairs.

Table 3. Comparison of the performance the RoadRunner algorithm with semantic partitioner system, before and after utilizing the probabilistic domain model

Classes				Comparative Results		
site	description	#pages	metadata	RoadRunner	Sempart (before)	Sempart (after)
amazon.com	cars by brand	21	yes	21	-	21
amazon.com	music bestsellers by style	20	no	-	-	-
buy.com	product information	10	yes	10	10	10
buy.com	product subcategories	20	yes	20	20	20
rpmfind.net	packages by name	30	no	10	10	10
rpmfind.net	packages by distribution	20	no	20	20	20
rpmfind.net	single package	18	no	18	18	18
rpmfind.net	package directory	20	no	-	20	20
uefa.com	clubs by country	20	no	20	-	20
uefa.com	players in the national team	20	no	20	-	-

Table 3 shows the performance of the original and modified semantic partitioner using the public RoadRunner data set. Of the ten categories, the modified semantic partitioner was able to extract information from three additional categories. The modified system was also able to extract information in the two categories (package directory and music bestsellers by style) where RoadRunner system fails since these pages do not follow a regular grammar. The ‘uefa’ data is organized in terms of complex tables, RoadRunner was able to infer the template by using two sample pages whereas the semantic partitioner (both initial and modified) was unable to extract from such tables using a single page. Overall, the performance of the modified semantic partitioner is better than the original one and it is comparable to the RoadRunner system.

5 Future Work and Conclusion

In this paper, we proposed a fast and scalable probabilistic model to improve the Web data annotations that are generated through (semi) automated IE systems. Our method can be distinguished by its capability of reasoning with contextual

information. Although the initial data contains many incorrect annotations and missing attributes, the Bayesian model presented here was shown to substantially improve the Web data annotations for both template driven and non-template driven Web site collections. We conjecture that the model can be incorporated into IE systems to improve their performance.

The future work includes the formulation of a generic expectation - maximization (EM) framework between an IE system and the Bayesian classifier described here which iteratively improves the annotations.

References

1. Guha, R., McCool, R.: TAP: A semantic web toolkit. *Semantic Web Journal* (2003)
2. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., McCurley, K.S., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: A case for automated large-scale semantic annotation. *Journal of Web Semantics* 1(1), 115–132 (2003)
3. Crescenzi, V., Mecca, G.: Automatic information extraction from large web sites. *Journal of ACM* 51(5), 731–779 (2004)
4. Lerman, K., Getoor, L., Minton, S., Knoblock, C.: Using the structure of web sites for automatic segmentation of tables. In: *ACM SIGMOD*, Paris, France, pp. 119–130. ACM Press, New York (2004)
5. Vadrevu, S., Gelgi, F., Davulcu, H.: Semantic partitioning of web pages. In: *WISE*, New York, NY, USA, pp. 107–118 (2005)
6. Florescu, D.: Managing semi-structured data. *Queue* 3(8), 18–24 (2005)
7. Murphy, K.: A brief introduction to graphical models and bayesian networks. Available online at: <http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html> (1998)
8. Chickering, D.M.: Learning bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics V* (1996)
9. Gama, J.: Iterative bayes. *Theoretical Computer Science* 292(2), 417–430 (2003)
10. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: *IJCAI*, pp. 1300–1309 (1999)
11. Neville, J., Jensen, D.: Iterative classification in relational data. In: *AAAI Workshop on Learning Statistical Models from Relational Data*, pp. 13–20 (2000)
12. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: *ACM SIGMOD*, Washington, D.C, pp. 207–216 (1993)
13. Alpaydin, E.: 3. In: *Introduction to Machine Learning*, pp. 39–59. MIT Press, Cambridge (2004)

Creating Personal Histories from the Web Using Namesake Disambiguation and Event Extraction

Rui Kimura¹, Satoshi Oyama², Hiroyuki Toda³, and Katsumi Tanaka²

¹ KDDI Corporation

3-10-10 Iidabashi, Chiyoda-ku, Tokyo 102-8460, Japan
ui-kimura@kddi.com

² Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
{oyama,tanaka}@dl.kuis.kyoto-u.ac.jp

³ NTT Cyber Solutions Laboratories, NTT Corporation
1-1 Hikari-no-Oka, Yokosuka, Kanagawa 239-0847, Japan
toda.hiroyuki@lab.ntt.co.jp

Abstract. We have developed a system for gathering information from the Web, using it to create a personal history, and presenting it as a chronological table. It simplifies the task of sorting out the information for various namesakes and dealing with information in widely scattered sources. The system comprises five components: namesake disambiguation, date expression extraction, date expression normalization and completion, relevant information extraction, and chronological table generation.

Keywords: Web search, namesake disambiguation, event extraction, clustering, machine learning.

1 Introduction

It is said that queries containing a person's name account for 5-10% of all Web searches [1]. Many users consider information about people as search target. Many Internet users search for information about people, and the number will increase as Web relationships become more and more common, as evidenced by the explosion of social networking services such as MySpace [4]. Users are searching not only for the profiles of people, but also for pictures of them, information about events in which they participated, gossip and rumors about them, and anything else related to their history. Gathering such a wide variety of information about a person is troublesome.

Efforts to improve this process have led to several interesting alternatives. The system proposed by Al-Kamha et al. [2] clusters Web pages returned by a search engine for person-name queries by using three independent measures: attributes like phone number, e-mail address, and zip code, similarity between Web pages, and link relationships between Web pages. WebHawk [3], a system developed by

¹ <http://www.myspace.com/>

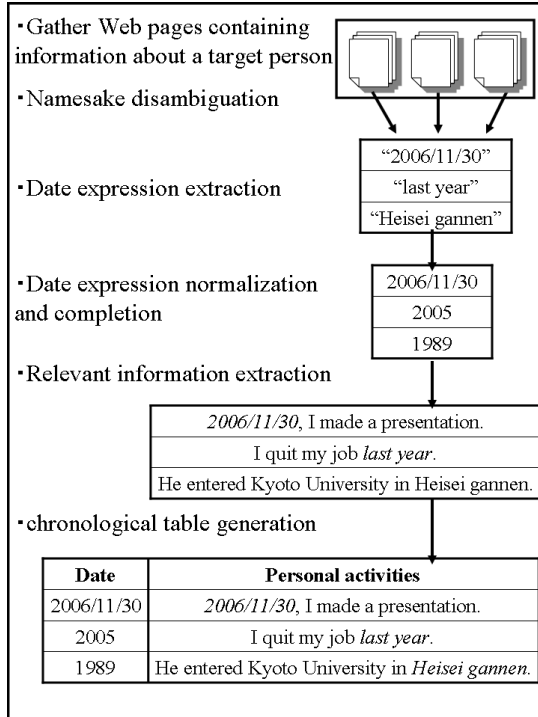


Fig. 1. System overview

Wan et al., extracts personal information, organizations, e-mail addresses, etc. from Web pages and clusters the pages on the basis of the extracted information. It then generates an informative description for each cluster so that the user can quickly identify the target person.

In contrast to these earlier efforts, which extract only basic information, we have developed a system that generates a more comprehensive summary for the target person by extracting information about his or her past activities as well as the basic information. This system enables users to easily obtain a more complete picture of the target person. Moreover, it presents the temporal information obtained about the person in a timeline format, as might be found in a biography or a chronological table. An overview of this automatic generation of a chronological table is shown in Figure 1.

The system uses the name of the target person to search for and gather Web pages containing information about the person. It does this using an existing search engine. If pages are found for different people having the same name, the system clusters the pages by person. The user then specifies the cluster corresponding to the target person. The system then collects the activities for the specified target person from the pages in the corresponding cluster. For the

system to be able to generate a chronological table, a date must be assigned to each activity. The system does this by identifying the expressions in each page that represents a date and then extracting the expressions and corresponding activities. Finally, using the extracted pairs of date expressions and activities, the system places the activities in chronological order and generates a chronological table.

2 Namesake Disambiguation

While the previous systems analyze the original Web pages found by the search engine, our system analyzes only the terms in the snippets shown on the search results pages. It then clusters the snippets to identify the pages belonging to each person. Compared with analyzing the original Web pages, our approach considerably reduces the time taken for downloading the pages and analyzing them.

Our system takes into account the kinds of terms in the snippets and page titles to distinguish between namesakes. Related objects and occupation terms are extracted from the page titles and snippets on the search results pages, and the search results are clustered using the extracted terms to distinguish the namesakes.

The names of people and organizations, i.e., “related objects,” are named entities. According to the definition used by IREX (Information Retrieval and Extraction Exercise) [4], a named entity consists of eight kinds of terms, ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY, and PERCENT. Many efforts have been made to extract named entities from text over the last decade or two. We use the Named Entity Extraction Tool (NExT)² developed by Watanabe et al. [5] to extract related objects.

We define occupation terms as terms that can be classified as job, role, or position in Nihongo Goi Taikai - a Japanese lexicon [6], and that consist of more than two letters. We add these terms to a dictionary of ChaSen [7], a Japanese morphological analysis system, and extract them. Example occupation terms are listed in Table 1.

As a baseline method, we extract nouns from page titles and snippets by using ChaSen.

We create a document vector based on the vector space model. We use terms extracted and weight them using term frequency (TF) and inverse document frequency (IDF). For document d and term m_x , the weight is given by

$$w(d, m_x) = tf(d, m_x) \frac{1}{1 + \log(df(m_x))} \quad (1)$$

where $tf(d, m_x)$ is the number of occurrences of term m_x in document d and $df(m_x)$ is the number of documents including term m_x .

² http://www.ai.info.mie-u.ac.jp/~next/docs/20040116_NExT.pdf

Table 1. Example occupation terms

Category	Examples
Job	medical director, analyst, interpreter, graduate student, foreign student, golfer, goal keeper, Shinto priest, Member of Parliament
Role	king, chief cabinet secretary, store manager, auditor, assistant, doctor, Member of Parliament, assistant general manager, executive director
Position	staff, instructor, CEO, screenwriter, closer, starting pitcher, home-plate umpire, autocrat, puppet

We make document vectors

$$\mathbf{v}_{noun}(d) = (w(d, l_1), w(d, l_2), \dots, w(d, l_m)) \quad (2)$$

where $w(d, l_x)$ is the weight of term l_x in document d and m is the number of different terms appeared in the document set. We cluster documents using these document vectors and the single linkage clustering method, which is a hierarchical clustering method.

3 Date Expression Extraction

To collect timeline information, we have to extract date expressions indicating when each event happened. Mizobuchi et al. [8] divided terms representing time into time expressions (representing hours, minutes, and seconds) and date expressions (representing years, months, and days). Since we want to create a historical timeline for the target person, we define the smallest time period as a day.

Date expressions are named entities. We could extract them using the NExT named entity extraction tool, which was mentioned above, but this tool cannot extract date expressions written in Japanese hiragana. Moreover, it extracts expressions that are of no use in constructing a timeline, such as “the first year” in “the first year of high school.” Therefore, we constructed specialized rules.

We constructed the extraction rules using regular expressions, like those shown in Table 3. Examples of target date expressions are shown in Table 2. As shown in Table 3, the date expressions are categorized as either complete or incomplete. A complete expression has the full complement of year, month, and day, while an incomplete expression lacks one or two of them, indicates a relative time, such as “two years ago” or “four months later,” or indicates an ambiguous time, such as “at the end of the year.” Other incomplete expressions include the seasons, like “summer,” and ambiguous expressions like “early part of October” or “some years later.” While “at the end of the year” probably means December, “summer” could mean one of several months. The day is ambiguous in “early part of October,” and the number of years in “some years” in “some years later” is unclear. Though these expressions are often used in ordinary texts, we do not extract them due to the ambiguities.

Using regular expressions, we extract date expressions from the clustered Web pages (after removing their HTML tags). Terms that indicate a time transition,

Table 2. Example date expressions to be collected

Completeness	Type	Example
Complete	numeric	2005/4/13
	Western	May 3, 1999
	Japanese	Heisei 18 Nen 12 Gatsu 12 Nichi
	proper noun	New Year's Day of 2003
Incomplete	relative	two years ago
	ambiguous expression	at the end of the year
	year and month abbreviated	on Wednesday, 29th
	year abbreviated	May 3
	month and day abbreviated	Heisei 18
	day abbreviated	May 2003

Table 3. Examples of regular expressions used to extract date expressions

Regular Expressions
<code>([0-9]+)[/\.\-]([0-9]+)[/\.\-]([0-9]+)</code>
<code>(Heisei Syowa Taisyō Meiji)\s*([0-9]+ </code> <code>(ju hyaku sen ichi ni san shi go roku shichi hachi ku)+ gan)(endo nen)</code>
<code>([0-9]+ (ju hyaku sen ichi ni san shi go roku shichi hachi ku)+)</code> <code>\s*(endo nen getsu nichi)</code>
<code>([0-9]+ (ju hyaku sen ichi ni san shi go roku shichi hachi ku)+)</code> <code>\s*(nen kagetsu getsu nichi)(mae ato)</code>

such as “next year” and “two days ago,” are useful, but terms that indicate a time period, such as “for five days,” are not, so these latter terms are not targeted.

Terms indicating year, month, or day appearing adjacently are combined, and the combined expression is considered to be one date expression. For example, if “in October” appears next to “on the last day” or, in Japanese, these two terms are connected with the Japanese particle “no,” they are combined into “on the last day in October,” which is then taken as one date expression.

4 Date Expression Normalization and Completion

After the date expressions, like the examples in Table 2, are extracted, the years, months, and days are normalized. A year is normalized into a four-digit number, a month into a two-digit number, and a day into a two-digit number. A complete date expression is thus normalized into an eight-digit number. For example, “1982/5/3” is converted into 19820503 and “Heisei 18 nen 6 gatsu” is converted into 20060600.

The time transition expressions are incomplete and cannot be normalized. Instead, their meaning is completed by using the context to estimate the date represented. More specifically, the meaning of an incomplete expression is com-

Table 4. Examples of completed date expressions

Incomplete expression problem	Preceding expression information used	Completed expression
May 23 (year missing)	April 13, 2005 (year)	20050523
the fifth day (year and month missing)	May 3, 1999 (year and month)	19990305
in the same month (no time transition)	May 3, 1999 (year and month)	19990500
this year (no time transition)	Heisei 18 Nen 12 Gatsu (year)	20060000
yesterday (time transition)	2005/4/13 (year, month, and date)	20050412
a half year later (time transition)	2005/4/13 (year and month)	20051000

pleted by identifying time expressions appearing prior to the incomplete expression and using the information they contain to complete the expression. Examples of completed date expressions are shown in Table 4.

Three rules are used for completion. The first rule covers expressions in which the year or the year and month are abbreviated. Such expressions are completed by using the corresponding data from a preceding date expression. The second rule covers expressions without a time transition, for example, “at the same month.” Such expressions are completed by using a date with the same date in a preceding date expression. The third rule covers expressions with a time transition. If the expression represents l years and m months and n days later, the expression is completed using a date that is l years and m months and n days later than the date represented by the preceding date expression. Similarly, if the expression represents l years and m months and n days ago, the expression is completed using a date that is l years and m months and n previous to the date represented by the preceding date expression. In these third rule calculations, the significant values of both the incomplete date expression and the preceding date expression are considered. Consider the examples shown in Table 4. The incomplete expression “yesterday” is completed using the preceding date expression “2005/4/13.” In both expressions, year, month, and day are significant, so the completion calculation is “subtract one day from 2005/4/13.” In contrast, only year and month are significant in the incomplete expression “a half year later,” while year, month, and day are significant in the preceding date expression “2005/4/13.” The completion calculation is thus “add six months to 2005/4.”

5 Relevant Information Extraction

Since the Web pages often have information for more than one person with the same name, the system has to identify the information that is relevant to the

target person. Moreover, it has to determine whether the paragraph is information is relevant to the person's history. It does this by using the HTML tree structure of each page and machine learning.

Previous approaches to exacting information from Web pages are generally usable only for Web pages with similar structures, design, or contents or for a few pages with many instances on each page. Our system is designed to extract information from all Web pages containing relevant information. Therefore, it must be able to extract information from various types of Web pages. Moreover, different queries should return different personal histories, so the extracted results must be query dependent. However, since it is virtually impossible to create an extraction rule for each query, i.e., each target person name, the extraction rule must be query independent. We thus use a robust approach to extracting relevant information—the use of a rough context-content model. Using the HTML tree structure of the retrieved pages, the system can guess the context in which each paragraph is used.

While XML tags describe the architecture of a page, HTML tags describe the design. The HTML tag names are not the name of a class, as are XML tags, so the tags around text do not indicate the meaning of the text. Nevertheless, many documents, newspaper stories, magazine articles, and so on have a heading representing the content, so the content can be roughly identify by simply looking at the heading. Moreover, the heading is usually easily distinguishable from the text. For example, the heading is generally located above the text. The same is true of Web pages containing HTML tags. Therefore, estimating whether some text contains information about the target person can be done by checking the heading.

The W3C's HTML Website³ explains that HTML tags are either for block-level elements or for in-line elements. Block-level elements create larger structures and begin on a new line. They are more important for the HTML paragraph structure than the in-line elements, so we use the tags for the block-level elements and remove those for the in-line elements before extracting relevant information.

The system extracts context C_A of text T_A , which is enclosed by a pair of tags, using the following algorithm. The HTML node to which T_A belongs is N_T , and the HTML node to which C_A belongs is N_C . If N_X is an HTML node and N_{root} is the HTML root node, $\text{PreviousNode}(N_X)$ is defined as follows.

$\text{PreviousNode}(N_X) =$

- (i) $\text{PreviousSiblingNode}(N_X)$
if $\text{PreviousSiblingNode}(N_X)$ is not null,
 - (ii) $\text{ParentNode}(N_X)$
if $\text{PreviousSiblingNode}(N_X)$ is null
AND if $\text{ParentNode}(N_X)$ is not null
 - (iii) N_{root}
- (3)

³ <http://www.w3.org/TR/html401/struct/global>

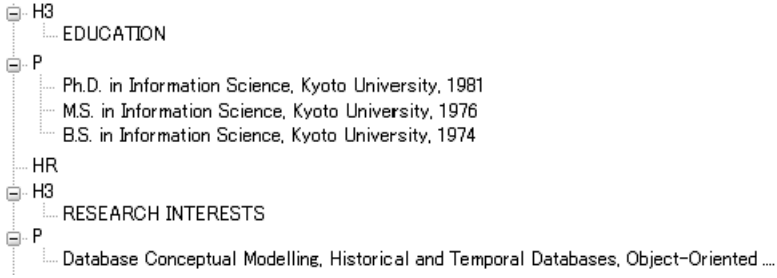


Fig. 2. HTML tree

The node for which the system is searching defined as N_S , and at first N_S is the parent node of N_T . Using the definition of $\text{PreviousNode}(N_X)$, N_C is recursively searched as follows.

$$\begin{aligned}
 N_C = & \\
 & \text{(i) RootNode} \\
 & \quad \text{if PreviousNode}(N_S)\text{is RootNode,} \\
 & \text{(ii) PreviousNode}(N_S) \\
 & \quad \text{if PreviousNode}(N_S)\text{is a text node,} \\
 & \text{(iii) the child node of PreviousNode}(N_S) \\
 & \quad \text{if PreviousNode}(N_S)\text{is a node of a heading tag.}
 \end{aligned}
 \tag{4}$$

If N_C has not been determined, N_S is set to $\text{PreviousNode}(N_S)$ and Formula 4 is repeatedly applied until N_C is determined. If N_C is determined by Formula 4, the context, C_A , is defined as the inner-text of N_C .

Using these algorithms, the system can find a context for each text item on a Web page. Consider the example HTML tree in Figure 2 and a search for the context of the text “Database Conceptual Modelling,” First, N_S is set to the previous node in the text, the P tag node. Next, N_S is searched for and the H3 tag node is found. Using Formula 4, the system determines that N_S is a node of a heading tag, meaning that N_C is a child node of N_S , the node for “RESEARCH INTERESTS.” In this way, the context of “Database Conceptual Modelling, ...” is determined to be “RESEARCH INTERESTS.”

A feature vector is constructed for each identified context-text pair on a page. The nouns on the page are extracted using Chasen, and each one is defined as a feature. The feature value is the number of occurrences of the noun on the page. Although numbers representing dates would be useful for identifying information useful for generating the chronological table, if all the numbers on a page were defined as features, the number of features would be burdensome. Moreover, number features rarely occur more than once in the data set. This causes feature sparseness and low classification accuracy. Therefore, the numbers are defined as one of five features: “one-digit number,” “two-digit number,” “three-digit number,” “four-digit number,” and “more-than-four-digit number,” as shown in Table 5. All date expressions are defined as “date expression.”

Table 5. Aggregated features used to train binary classifier

Type	Feature	Example
numbers	one digit number	3
	two digit number	32
	three digit number	524
	four digit number	1999
	more-than-five digit number	150,000,000
names of people	queried name	Hanako Sato
	part of queried name	Sato
	other person's name	Tanaka
date expressions	date expression	1999/12/1

Since we want to extract information only for the target person, the query name, part of the query name, and the names for other people are good features for extracting relevant information. However, if the names were directly used as features, the rules would greatly depend on the query names used to collect the training data of the machine learning; therefore, when the rules were applied to different data, the accuracy would be low. To make the rules independent of the queries, we abstract the features. The names of people are defined as one of three features, “queried name,” “part of queried name,” and “other person’s name.” This aggregation of features also helps prevent overfitting to the training data and results in good generalization to other data. Using these features, we train a binary classifier to classify the segments on a page into relevant and irrelevant ones.

6 Chronological Table Generation

To generate a chronological table for the target person, it is necessary to collect pairs of a date and an activity for the person. The sentences extracted likely include activities for the person. The style of the sentences may vary widely; for example, some sentences may have no date expressions and some may have more than one pair of a date and an activity. If a sentence has no date expression, it is necessary to search among the preceding sentences for a date expression related to the activity. If a sentence has more than one pair of a date and an activity, it is necessary to match the dates with the information. We have created four rules for doing this.

- The system first extracts sentences or parts of sentences that include date expressions. These expressions are normalized and completed, and the system makes a list of candidate texts which are the extracted sentences or incomplete sentences.
- If an extracted date-expression sentence does not end with a noun that is linked to a *sa-hen* verb, the sentence is eliminated from the candidate list.

- If the first word following a date expression is not “ni,” “yori,” “kara,” or a space, the sentence is eliminated from the candidate list.
- If a sentence does not have Japanese words, it is eliminated from the candidate list.

The candidate texts remaining following the application of these rules are used to generate a chronological table.

7 Evaluation

7.1 Namesake Disambiguation

As sample queries, we chose 20 Japanese names from a list of names that has at least three famous people in an article “same last name and first name” of Wikipedia⁴. We did a Google search on each name and used the first 100 results for each one. We regarded a pair of a page title and a snippet search result as one document. For each name, we made an answer set by manually clustering these documents so that each cluster corresponded to one person. The average number of clusters in the answer set was 22.85. The maximum number of clusters in correct data was 48, the same as the number of clusters for “Yutaka Kobayashi,” and 12.25% of the clusters had only one document.

The clustering results were evaluated by comparing them to the answer set (correct clusters), which was prepared manually, using precision, recall, and F-measure.

$$F\text{-measure} = \frac{\textit{precision} * \textit{recall} * 2}{\textit{precision} + \textit{recall}}. \quad (5)$$

We used an F-score measure [9] as the metric clustering accuracy.

$$F\text{-score} = \sum_{c \in C} \frac{|m_c|}{|m|} \max_{r \in R} \frac{2|m_r \cap m_c|}{|m_r| + |m_c|}, \quad (6)$$

where C is the set of clusters in the answer set, c is one cluster in the answer set, m_c is the set of documents belonging to cluster c , R is the set of clusters in the clustering result, r is one person in a classified result, m_r is the set of documents belong to cluster r , and m is a set of documents in the answer set.

We varied the number of clusters, which was used as the stopping condition for single-linkage clustering, between 1 and 99. The precision, recall, F-measure, and F-score for each name were averaged for each number of clusters. We created document vectors consisting of nouns and related objects and/or occupation terms and used them for our evaluation.

Figure 3 shows precision-recall curves for the classification results using our system and the different types of feature vectors as well as for a baseline method

⁴ <http://ja.wikipedia.org/w/index.php?title=%E5%90%8C%E5%A7%93%E5%90%8C%E5%90%8D&oldid=7825577>; dated 08:34, 22 September 2006

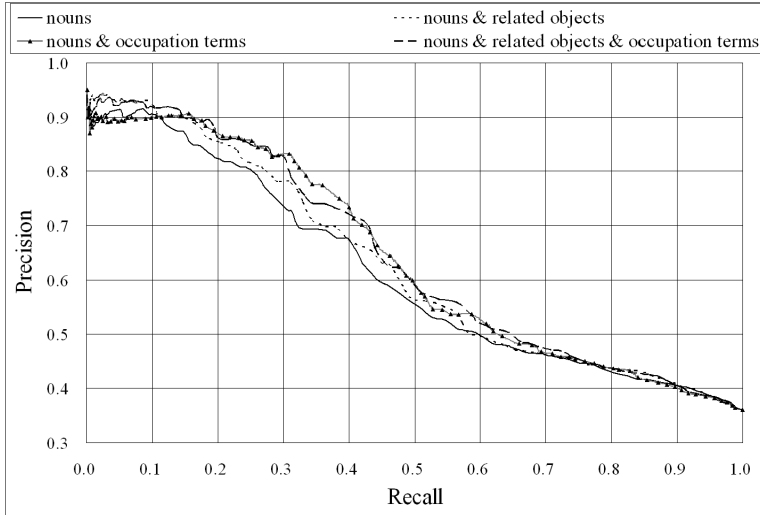


Fig. 3. Clustering accuracy

using only nouns as features. For all three types of feature vectors, our system performed better than the baseline method.

Table 6 shows the highest F-measure and F-score for all cluster numbers (denoted by subscript “max”) and those for the actual cluster number in the answer set (“answer”). All the maximum scores were scored using the feature vectors with nouns and related objects and occupation terms. Using the vectors with nouns and occupation terms and the vectors with nouns and related objects and occupation terms produced significantly better results than the baseline method.

7.2 Date Expression Extraction

We did a Google search on five Japanese names (Katsumi Tanaka, Satoshi Oyama, Yuri Ebihara, Junichiro Koizumi, and Hidetoshi Nakata) and used the first 20 results for each one. Then, using the 100 web pages, we evaluated the accuracy of our extraction component. Table 7 shows the number of automatically extracted date expressions, the number that were correct, and the actual number manually found. The recall was 71.3%, and the precision was 93.3%.

While the precision was sufficiently high, it can be improved by eliminating some mistake patterns. One pattern in particular (exemplified by “two days in Cairns”) accounted for more than half the mistakes. There were also misrecognition problems. For example, the “6.1.1” in “section 6.1.1” was recognized as “6/1/1.” Problems also occurred because the same Chinese (“kanji”) character can have more than one meaning. In Japanese, both “moon” and “month” use the same Chinese character, as do “day” and “sun.” This multiple usage of characters complicates the extraction of date expressions. Other misrecognized terms are part of a proper noun, which often raise in Japanese text.

Table 6. Clustering accuracy

Method	$F\text{-measure}_{max}$	$F\text{-score}_{max}$	$F\text{-measure}_{answer}$	$F\text{-score}_{answer}$
nouns	0.509	0.637	0.557	0.629
nouns & related objects	0.514	0.631	0.563	0.631
nouns & occupation terms	0.513	0.651	0.572	0.632
nouns & related objects & occupation terms	0.516	0.651	0.575	0.639

Table 7. Precision and recall of date expression extraction

Query	Extracted	Correct	Actual	Precision	Recall
Katsumi Tanaka	473	468	559	0.989	0.837
Satoshi Oyama	223	205	514	0.919	0.399
Yuri Ebihara	323	317	352	0.981	0.901
Junichiro Koizumi	505	435	557	0.861	0.781
Hidetoshi Nakata	293	271	397	0.925	0.683
Total	1817	1696	2379	0.933	0.713

It is difficult to avoid all extraction mistakes when using regular expressions. It is not practical to write rules for all patterns of the various proper nouns that appear in Web pages. We may need to use a morpheme analyzer to determine whether a term actually represents a date.

The recall was lower than precision because we emphasized precision in order to get a precise date for each piece of information. We did not extract uncertain terms, such as “2/3,” which can be either a fraction or a date, and “18,” which cannot be identified as a year but sometimes means “Heisei 18.” Such date expressions could possibly be extracted by using words appearing around them. For example, date expressions tend to be collocated with certain prepositions like “in” (“in May”) and “on” (“on May 5th”). They also tend to occur with specific words, like a day of the week, such as “Wednesday.”

7.3 Date Expression Normalization and Completion

We did a Google search on each name in Table 7 and used the first 100 results for each one. In the obtained data, there were 5264 date expressions, and 1159 of them were incomplete. On average, each of the 500 pages had about 10.5 date expressions, and about 2.3 of them needed to be completed. This meant that our objective was to complete one-fourth of the date expressions by using the other three-fourths. If the first date expression in a text did not have the year, the three rules defined for completion could not be applied. They could also not

Table 8. Precision of date expression completion

	Rule 1		Rule 2		Rule 3	
	Applied	Precision	Applied	Precision	Applied	Precision
Katsumi Tanaka	103	0.922	6	0.333	8	0.250
Satoshi Oyama	142	0.951	4	1.000	5	0.200
Yuri Ebihara	141	0.702	11	0.727	8	0.625
Junichiro Koizumi	139	0.626	17	0.353	24	0.583
Hidetoshi Nakata	195	0.610	15	0.867	16	0.250
Total	720	0.743	53	0.623	61	0.426

be applied to subsequent date expressions in that text lacking the year. There were 325 of these date expressions.

The date expressions to which the rules could not be applied were classified into “people can complete” (about 61%) and “people cannot complete” (about 39%) depending on whether a person could estimate the date by using the URL, title, and/or text on the Web page. For example, if the URL for the page was “http://blog.example.com/20050602/index.html,” a person could determine that it was written on 2 June 2005. If the title of the page was “EURO2004,” a person could determine that the events described on the page happened in 2004. The remaining 834 incomplete date expressions were completed by applying the rules. About 71.2% of them were completed correctly, so using only simple completion rules produced relatively good precision.

Table 7 shows the precision of each rule. The precision of the third rule, which was applied to date expressions with a time transition such as “three months ago,” was lower than those of the other rules. When constructing our three completion rules, we assumed that a time transition as one from a date of adjacent date expressions; however, in blog posts and news articles, most time transitions are from dates when the post or article was written. This explains why most of the incorrect completions were for blog posts and news articles. Since the names of famous people, which are searched for more often than other names, are frequently used in blog posts and news articles, this is a significant problem.

Since about 61% of the date expressions to which the rules could not be applied were classified into “people can complete,” performance could be improved by using more complex rules and additional information, like the URL.

7.4 Relevant Information Extraction

We conducted a preliminary experiment to compare the relevant information extraction performance of three machine learning methods: Naive Bayes [10], C4.5 decision tree learner [11], and SVM [12]. The C4.5 learner had the best classification accuracy. We thus used it to evaluate the classification accuracy of our system.

Table 9. Precision and recall of relevant information extraction

Use context-text model	true	true	false	false
Use aggregation of features	true	false	true	false
Precision	0.593	0.073	0.546	0.103
Recall	0.143	0.045	0.119	0.007
F-measure	0.231	0.055	0.195	0.013

For experimentation, we use 20 Japanese names used for experimentations in disambiguation of namesakes. We downloaded ten random web pages from the first 100 search results for each name. The text elements for each pages were classified manually to produce an answer set for 200 Web pages.

In this evaluation, we cross-validated the results by dividing the answer set into 20 subsets. One name out of 20 names was selected, and the 190 pages corresponding to the other names were used as training data. The ten pages corresponding to the name were used as test data. We repeated this evaluation 20 times, once for each name and calculated micro averages for precision, recall, and F-measure.

To evaluate our system, which uses both a context-text model and feature aggregation, we compared the accuracy for four data sets: one using both the context-text model and feature aggregation, one using only the context-text model, one using only feature aggregations, and one using neither. Due to the computational cost, we counted the number of occurrences in a data set for each feature and used the top 100 features. The results are shown in Table 9.

The scores for our system using both the context-text model and feature aggregation were better than those of the other methods. Especially, feature aggregation was very effective for improving both precision and recall. Even for the best method, however, its recall was not very high. This was probably due to the great difference between the number of positive examples and negative examples. The data used contained only 1.55% positive examples. We plan to incorporate an existing technique for handling imbalanced data, such as artificially generated virtual positive data [13].

8 Conclusion

Our proposed system gathers information from the Web, uses it to create a personal history, and presents the history as a chronological table. It simplifies the task of sorting out the information for various namesakes and dealing with information in widely scattered sources. The system has five components: namesake disambiguation, date expression extraction, date expression normalization and completion, relevant information extraction, and chronological table generation. This system will thus enable efficient creation of personal histories.

Acknowledgments

This work was supported in part by Grants-in-Aid for Scientific Research (Nos. 18049041 and 19700091) from MEXT of Japan and by a MEXT project entitled “Software Technologies for Search and Integration across Heterogeneous-Media Archives.”

References

1. Guha, R., Garg, A.: Disambiguating people in search. Stanford University (2004)
2. Al-Kamha, R., Embley, D.W.: Grouping search-engine returned citations for person-name queries. In: Proc. ACM WIDM 2004, pp. 96–103. ACM Press, New York (2004)
3. Wan, X., Gao, J., Li, M., Ding, B.: Person resolution in person search results: Webhawk. In: Proc. ACM CIKM 2005, pp. 163–170. ACM Press, New York (2005)
4. IREX Committee. In: Proceedings of the IREX Workshop, IREX Committee (1999)
5. Watanabe, I., Masui, F., Fukumoto, J.: Improvement of next performance: Elavolating precision and userbility of the named entity extraction tool. In: Proc. NLP 2004, pp. 413–415 (in Japanese) (2004)
6. Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Ooyama, Y., Hayashi, Y.: Nihongo Goi Taikei - A Japanese Lexicon (CD-ROM). Iwanami Syoten (in Japanese) (1999)
7. Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Matsuda, H., Takaoka, K., Asahara, M.: Japanese Morphological Analysis System ChaSen version 2.2.1 (2000)
8. Mizobuchi, S., Sumitomo, T., Fuketa, M., Aoe, J.: A method for understanding time expressions. In: Proc. IEEE SMC 1998, pp. 1151–1155. IEEE Computer Society Press, Los Alamitos (1998)
9. Zhao, Y., Karypis, G.: Evaluation of hierarchical clustering algorithms for document datasets. In: Proc. ACM CIKM 2002, pp. 515–524. ACM Press, New York (2002)
10. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. UAI 1995, pp. 338–345 (1995)
11. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
12. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
13. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. JAIR 16, 321–357 (2002)

Comparing Clustering Algorithms for the Identification of Similar Pages in Web Applications

Andrea De Lucia¹, Michele Risi¹, Giuseppe Scanniello², and Genoveffa Tortora¹

¹ Dipartimento di Matematica e Informatica, Università di Salerno, Via Ponte Don Melillo, 84084, Fisciano (SA), Italy

² Dipartimento di Matematica e Informatica, Università della Basilicata, Viale Dell'Ateneo, Macchia Romana, 85100, Potenza, Italy
{adelucia,mrisi}@unisa.it, giuseppe.scanniello@unibas.it, tortora@unisa.it

Abstract. In this paper, we analyze some widely employed clustering algorithms to identify duplicated or cloned pages in web applications. Indeed, we consider an agglomerative hierarchical clustering algorithm, a divisive clustering algorithm, k-means partitional clustering algorithm, and a partitional competitive clustering algorithm, namely Winner Takes All (WTA). All the clustering algorithms take as input a matrix of the distances between the structures of the web pages. The distance of two pages is computed applying the Levenshtein edit distance to the strings that encode the sequences of HTML tags of the web pages.

Keywords: Clone detection, clustering algorithms, reverse engineering.

1 Introduction

Code cloning in web applications is one of the factors that make their maintenance more difficult and time consuming [1][11]. A code clone is a code portion in source files that is identical or similar to another. Developers introduce clones for various reasons such as lack of a good design, fuzzy requirements, undisciplined maintenance and evolution, and reusing code by copy-and-paste.

Recently, researchers have extensively studied clone detection for static web pages, written in HTML, or dynamic ones [3][11]. Clone detection approaches for web applications are usually based on similarity measures: two pages will be considered clones if they are characterized by the same values of the defined measure [4][5][11][12]. For example, Di Lucca *et al.* [4] encode the sequences of tags of HTML and ASP pages into strings and identify pairs of cloned pages at structural level by computing the Levenshtein string edit distance [9] between these strings. A pair of web pages is considered a clone if their Levenshtein edit distance is zero. Ricca and Tonella in [11] enhance the approach based on the Levenshtein edit distance proposed in [4]. In particular, they adopt a hierarchical clustering algorithm to identify clusters of duplicated or similar pages to be generalized into a dynamic page. Similarly, in [12] the authors propose a semiautomatic approach to identify and align static HTML pages whose structure is the same and whose content is in different

languages. Pages with similar structures are identified by adopting an agglomerative hierarchical clustering algorithm. An approach based on a competitive clustering algorithm to identify similar pages at structural and content level is proposed in [2]. To group pages at structural level, their structures are encoded into strings and then the Levenshtein algorithm is used to achieve the distances between pairs of pages. De Lucia *et al.* [3] also use Levenshtein distance to compute the similarity of two pages at structure, content, and scripting code level. Clones are characterized by a similarity threshold that ranges from 0%, for disjoint code, up to 100%, for identical pages.

In this paper, we compare the results of applying some clustering algorithms belonging to the categories hierarchical and partitionial [6] in the identification of cloned page at structural level (i.e., pages with similar sequences of HTML tags). Concerning the hierarchical methods we consider the agglomerative [8] and divisive clustering algorithms [7], while within the partitionial category we selected a variant of k-means [10] algorithm and the Winner Takes All (WTA) [6] algorithm. Web pages can be both static and dynamic and can be implemented using any kind of server side scripting code. Similarly to the approaches proposed in [2][4][11], we consider the Levenshtein string edit distance [9] as basic metric to identify similarity between pairs of pages. We have compared the selected clustering algorithms on a case study composed of three web applications developed using JSP technology.

The remainder of the paper is organized as follows. In Section 2 we present the process to identify page similarity at the structural level, while the results obtained by applying the considered clustering algorithms are presented in Section 3. Final remarks conclude the paper.

2 Identifying Cloned Pages at Structural Level

The process we adopted is composed of the following sequential phases: *Preprocessing*, *Computing Distance Matrix*, and *Grouping Similar Pages*. The *Preprocessing* phase aims at turning the page structures implemented by specific sequences of HTML tags into a more suitable string representation. This representation is produced by means of a depth-first traversal of the abstract syntax tree of both static and dynamic page structures. Each node of the syntax tree of a page is decorated with a HTML tag and with a set of attributes, such as text attribute, target source code, image attribute, etc. The string representations of the structure of a web page is obtained encoding the HTML tags into symbols of an alphabet before being concatenated into the string corresponding to its structure.

The *Computing Distance Matrix* phase adopts the Levenshtein edit string distance [9] to obtain similarity measures between pairs of pages. The Levenshtein edit distance is defined as the minimum cost to align two strings. The Levenshtein distance between pairs of pages is then used to build the *Distance Matrix* considering all the possible pairs of pages of a given web application. The Distance Matrix is then provided as input to the phase *Grouping Similar Pages*, which can be instantiated with different clustering algorithms. In particular, we considered an agglomerative [8], a divisive [7], k-means [10], and the WTA (Winner Takes All) [6] clustering algorithms. The agglomerative hierarchical clustering algorithm [8] begins with each page in a distinct cluster, and then hierarchically merges clusters together. This results

in a dendrogram, which represents the nested groups of pages and similarity levels at which groups change. The dendrogram can be cut at different levels to obtain different clusters (i.e., the tuning values of our approach). In the divisive hierarchical clustering algorithm [7], clusters are divided until each page is placed in a distinct cluster. The choice of the divisive step represents the tuning values of our approach. The k-means algorithm [10] is used to classify pages of a given data set through a priori fixed number of disjoint clusters (the tuning value of our approach). The main idea is to define a centroid for each cluster of pages to identify. The algorithm iteratively refines the initial centroids, minimizing the average distance of pages to their closest centroids. Finally, we also considered WTA [6], an Artificial Neural Network (ANN) clustering algorithm. The used ANN is essentially based on a two dimensional grid of neurons, which weights are continuously adapted to the vectors of the distances between the pages of a web application. The number of neurons is provided as input (i.e., the tuning value of our approach) and represents the number of clusters that the algorithm should identify (i.e., the number of expected clusters). The neuron with weight vector most similar to the input vector is called the winner. The weights of the winner and its closer neurons are then adjusted towards the input vectors. The training process is concluded either the neurons do not change their position or a termination threshold for the iteration is reached. Clusters are identified associating the pages to the closer neuron in the ANN.

3 The Case Study

The clustering algorithms have been compared on web applications implemented using JSP technology. In particular, we considered the web sites of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 02) and the 1st International School of Software Engineering (ISSE 03), and the SRA (Student Route Analysis) web application. Some descriptive statistics of these web applications are reported in Table 1. The second row contains the number of files composing the application, while the second and third rows contain the number of static and dynamic pages, respectively.

Table 1. Descriptive statistics of the analyzed web applications

	<i>SEKE 02</i>	<i>ISSE 03</i>	<i>SRA</i>
Number of Component Files	1268 files in 63 folders	66 files in 14 folders	380 files in 45 folders
Number of HTML Pages	49	12	22
Number of JSP Pages	108	7	45

3.1 Assessing the Results

To compare the results achieved by applying the different clustering algorithms we adopted the precision and recall metrics. In our case the recall is defined as the ratio between the number of actual pairs of similar pages identified by the tool over the total number of actual pairs of similar pages in the web application. Differently, the precision is the number of actual pairs of similar pages identified by the tool over the total number of identified pairs. The precision and recall values are computed

turning both actual clusters and the clusters automatically identified by the tool into pairs of similar pages. In particular, the clones manually identified are reported in a gold matrix, while those identified by the tool are reported in result matrix.

The evaluation of precision and recall is not straightforward. Thus to better assess the achieved results we also considered the trade-off between the values of precision and recall. In particular, we consider the F-measure that is defined as the harmonic mean of the precision and recall values. The trade-off configuration corresponds to the tuning value that allows the identification of the larger F-measure value.

Table 2.a and 2.b report the trade-off configurations achieved on the SEKE 02 web application. In particular, the first columns contain the adopted clustering algorithms, while the trade-off configurations are reported in the second columns. Concerning the WTA algorithm these tables also report the number of expected clusters (i.e., the first number) and the number of clusters that the algorithm actually identified (i.e., the second number). The precision and recall values are shown in the third and fourth columns, respectively. Finally, the F-measures of the presented configurations are contained in the last columns. The precision and recall values corresponding to the trade-off configurations of the static pages of the SEKE 02 web application (see Table 2.a) revealed that the k-means clustering algorithm produces better results. The trade-off configurations are 38 for the divisive clustering algorithm and the hierarchical clustering algorithms, while is 31 for the WTA clustering algorithm. Note that the WTA clustering algorithm produced worse results (i.e. the precision and recall values were 0.523 and 0.578, respectively) than the other clustering algorithms. Table 2.b shows that the better values of the precision and recall were obtained by using the divisive hierarchical clustering algorithm (0.695 is the F-measure value). Indeed, all the clustering algorithms except the WTA clustering algorithm produced comparable results. We can also observe that the number of clusters identified by the WTA algorithm is lower than the number of clusters identified by the other algorithms. The results achieved by employing the trade-off configurations on the ISSE03 web application are reported in Table 3.a and 3.b, respectively. The results obtained by applying the considered clustering algorithms are very similar. Indeed, on the static pages (see Table 3.a) the WTA clustering algorithm produced more relevant results in terms of recall value (i.e., 0.888). The results obtained by using the trade-off configurations on the agglomerative clustering algorithm and the k-means algorithm were the same in terms of precision and recall values.

On the dynamic pages of the ISSE 03 web application better results were achieved when the WTA and k-means clustering algorithms were adopted (see Table 3.b). For both the clustering algorithms the precision and recall values are 1 and 0.666, respectively. Regarding the WTA clustering algorithm, the difference between expected and identified clusters is 1. Let us also note that the agglomerative and divisive clustering algorithms produced the same results.

Finally, Table 4.a shows the results achieved by using the trade-off configurations on the static pages of the SRA web application. This table shows that the divisive clustering algorithm produced better results (i.e. the precision and recall values were 0.959 and 0.979, respectively). Similar results were achieved by employing the agglomerative clustering algorithm and WTA (the precision and recall values were 0.921 and 0.974, respectively). Differently, for the k-means the precision value was 1, while the recall value was 0.812. Regarding the WTA clustering algorithm the

Table 2. Results obtained by using the trade-off configurations on SEKE 02

	Clusters	Prec.	Recall	F-meas.
Agglom.	38	0.565	0.684	0.619
Divisive	38	0.565	0.684	0.619
k-means	40	0.923	0.631	0.750
WTA	31/31	0.523	0.578	0.549

(a)

	Clusters	Prec	Recall	F-meas.
Agglom.	90	0.877	0.574	0.694
Divisive	87	0.820	0.603	0.695
k-means	86	0.781	0.574	0.662
WTA	57/57	0.552	0.660	0.601

(b)

Table 3. Results obtained by using the trade-off configurations on ISSE 03

	Clusters	Prec.	Recall	F-meas.
Agglom.	6	0.5	0.666	0.571
Divisive	5	0.437	0.777	0.559
k-means	6	0.5	0.666	0.571
WTA	3/3	0.363	0.888	0.515

(a)

	Clusters	Prec.	Recall	F-meas.
Agglom.	4	0.5	0.666	0.571
Divisive	4	0.5	0.666	0.571
k-means	5	1	0.666	0.800
WTA	6/5	1	0.666	0.800

(b)

Table 4. Results obtained by using the trade-off configurations on SRA

	Clusters	Prec.	Recall	F-meas.
Agglom.	6	0.921	0.979	0.949
Divisive	7	0.959	0.979	0.969
k-means	9	1	0.812	0.896
WTA	6/6	0.921	0.979	0.949

(a)

	Clusters	Prec.	Recall	F-meas.
Agglom.	33	0.437	0.466	0.451
Divisive	33	0.437	0.466	0.451
k-means	31	0.208	0.333	0.256
WTA	27/26	0.206	0.4	0.272

(b)

number of expected and identified clusters coincides. The results of the trade-off configurations on the dynamic pages are shown in Table 4.b. Let us note that due to the low number of dynamic pages similar at structural level the clustering algorithms generally produced bad results.

3.2 Remarks

To identify the trade-off value the software engineer has to try all the possible configurations and then the clusters identified at each iteration have to be manually assessed. Hence, methods to automatically filter out surely bad tuning configurations and to get more quickly the trade-off configuration should be devised in the identification of similar pages. Indeed, the web application used as case study provided some directions to better support the software engineer. In particular, we observed that in most cases the break-even configuration (i.e., the larger number of expected clusters such that the WTA identifies non empty clusters) is very similar to the trade-off configuration. In particular, we observed that on the static pages of SEKE 02 the break-even and the trade-off configurations are nearly identical. On the other hand, the break-even and trade-off configurations coincide in case WTA is applied on the structure of the static pages of SRA. We also note that a larger difference between break-even and trade-off configurations is obtained when the web applications are composed of few pages (for example, ISSE 03). A larger difference between the break-even and trade-off configurations was also obtained in case the web application has a low number of pages similar at structural level.

4 Conclusions

In this paper we have presented an initial experiment on the use of an agglomerative hierarchical clustering algorithm, a divisive clustering algorithm, a variant of the k-means partitioning clustering algorithm, and a widely employed partitioning competitive clustering algorithm, namely WTA in the identification of web page similarity at the structural level. These algorithms have been employed to detect similar web pages in dynamic and/or static web sites according to the Levenshtein string edit distance. This distance has been used as basic metric to identify similarity between pairs of pages considering their structures. The selected algorithms have been evaluated on two medium and one small web applications developed using JSP technology. The results of the presented case study revealed that all the selected clustering algorithms generally produce comparable results.

References

- [1] Boldyreff, C., Tonella, P.: Web Site Evolution. Special issue of *Journal of Software Maintenance* 16(1-2), 1–4 (2004)
- [2] De Lucia, A., Scanniello, G., Tortora, G.: Using a Competitive Clustering Algorithm to Comprehend Web Applications. In: *Proc. of 8th IEEE International Symposium on Web Site Evolution*, Philadelphia, Pennsylvania, pp. 33–40. IEEE CS Press, Los Alamitos (2006)
- [3] De Lucia, A., Francese, R., Scanniello, G., Tortora, G.: Identifying Cloned Navigational Patterns in Web Applications. *International Journal of Web Engineering* 5(2), 150–174, Rinton Press (2006)
- [4] Di Lucca, G.A., Di Penta, M., Fasolino, A.R.: An Approach to Identify Duplicated Web Pages. In: *Proc. of 26th Annual International Computer Software and Application Conference*, Oxford, UK, pp. 481–486. IEEE CS Press, Los Alamitos (2002)
- [5] Di Lucca, G.A., Fasolino, A.R., De Carlini, U., Pace, F., Tramontana, P.: Comprehending web applications by a clustering based approach. In: *Proc. of the 10th International Workshop on Program Comprehension*, Paris, France, pp. 261–270. IEEE Computer Society Press, Los Alamitos (2002)
- [6] Duda, R.O., Hart, P.E., Stork, D.G.: *Pattern Classification*. Wiley-Interscience Publication, JOHN WILEY & SONS, Inc. New York, pp. 576–581
- [7] Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York (1990)
- [8] King, F.: Step-wise clustering procedures. *Journal of the American Statistical Association* 62, 86–101 (1967)
- [9] Levenshtein, V.L.: Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory* 10, 707–710 (1966)
- [10] Mcqueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297 (1967)
- [11] Ricca, F., Tonella, P.: Using Clustering to Support the Migration from Static to Dynamic Web Pages. In: *Proc. of International Workshop on Program Comprehension*, Portland, Oregon, USA, pp. 207–216 (2003)
- [12] Tonella, P., Ricca, F., Pianta, E., Girardi, C.: Restructuring Multilingual Web Sites. In: *Proc. of International Conference on Software Maintenance*, Montreal, Canada, pp. 290–299. IEEE CS Press, Los Alamitos (2002)

Structural Patterns for Descriptive Documents

Antonina Dattolo¹, Angelo Di Iorio², Silvia Duca²,
Antonio Angelo Feliziani², and Fabio Vitali²

¹ Department of Mathematics and Applications R. Caccioppoli,
University of Napoli Federico II, Italy
dattolo@unina.it

² Department of Computer Science, University of Bologna, Italy
{diiorio, ducas, afeliziana, vitali}@cs.unibo.it

Abstract. Combining expressiveness and plainness in the design of web documents is a difficult task. Validation languages are very powerful and designers are tempted to over-design specifications. This paper discusses an offbeat approach: describing *any* structured content of *any* document by only using a very small set of patterns, regardless of the format and layout of that document. The paper sketches out a formal analysis of some patterns, based on grammars and language theory. The study has been performed on XML languages and DTDs and has a twofold goal: coding empirical patterns in a formal representation, and discussing their completeness.

Keywords: Patterns, grammars, descriptive schemas, completeness.

1 Introduction

The World Wide Web has become the greatest repository of information ever existed. The more the centrality of the WWW increases, the more web documents become heterogeneous and complex. We do not perceive *heterogeneity* as a problem, to be solved by flattening all those languages into a plain, unspecific and incomplete one. Neither we want to define 'The' universal exhaustive language able to describe any domain and any application. What we are rather looking for is a general model to design languages and documents, in order to make them simple, complete and processable.

This work is part of a more ambitious project aiming at defining, separating and extracting constituents of *any* document so as to reformulate a few of them, to reuse some of them in different contexts, or to convert and mix sub-components. Our overall approach relies on the definition of *abstract and generic formats* able to fully describe the most relevant bits of digital documents, regardless of their actual format, layout and storage. In particular, we consider *any* document as a composition of five components (dimensions) independent but connected each other: content, structure, presentation, behavior and meta-data. We are not interested here in the details of our model (a deeper discussion can be found in [DDID07]) but we only refer to the *structure*. Our goal is discussing how *any* structured content can be *described* by a very small set of objects and composition rules. We already presented in [DIGV05] some *patterns*

to capture the most common structures of digital documents. What we want to do here is resuming that analysis and proposing a theoretical framework to discuss some properties of those patterns.

In particular, this paper sketches out a grammar-based proof of patterns' completeness. The interested reader is referred to [DDID07] for the complete proof. The point here is understanding what 'completeness' means in the context of descriptive documents, and how the language theory help us in proving it. Researchers have already used grammars for XML, in different contexts: [MLMK05] classified and compared schema languages, [Via01] discussed opposition and synergy between databases and XML, and [BMNS05] studied the relation between DTDs and XML Schema. Our goal is quite different: proving that *any* DTD can be transformed into a *descriptive* one, based on few patterns but able to capture the same structural information. Our analysis is performed over XML-based languages and DTDs, for sake of simplicity, but could be extended to other languages with appropriate modifications.

The paper is organized as follows: section 2 introduces a taxonomy of different levels of descriptiveness, section 3 presents our pattern-based model, section 4 defines grammars for (common or pattern-based) DTDs. Finally the completeness of the pattern-based language is sketched out.

2 Structures for Descriptive Documents

The first step of our work consists of understanding which constructs and rules are really needed for descriptive schemas. The distinction between prescriptive and descriptive languages have been widely studied in the literature [Ren00]. What these approaches change is the role itself of the validation [Pic01]. Traditionally validation is *strict*, because it is used as a "go/non-go" gauge to verify *in advance* whether or not a data set conforms to a set of requirements. A *loose* validation is rather used to capture abstract and structural information about a text. Both strict and loose validations are useful. What is important is designing languages and schemas by keeping in mind their features and differences, and applying them in right contexts.

Different *levels of descriptiveness* exist, depending on what designers reckon as important, what can be relaxed, what can be omitted, what can be expressed in a different way. To discuss these levels, we use DTD-based examples since they are shorter and more direct but we can use any other language, in exactly the same way. We identify six relevant levels of descriptiveness:

- **Prescriptive (PRE):** a prescriptive DTD imposes a set of rules which all matching documents must follow. Prevent errors in a production chain, based on strict validation.
- **Descriptive No Alternatives (DNA):** a descriptive DTD without alternatives do not allow users to force a choice between two (or more elements). The basic idea is that alternatives are meant to inhibit incorrect structures, but they are not required when all documents already exist and the DTD is used to describe all those documents.

- **Descriptive No Cardinality (DNC):** a descriptive DTD without alternatives can be further generalized by relaxing constraints over the cardinality of each single element. The idea is that by forcing cardinalities some documents could be considered invalid, even if they belong to the same class.
- **Descriptive No Order (DNO):** constraints over the order can be relaxed as well. Imposing order is something extremely useful when invalid documents obstruct a complex process, but it makes much less sense when the goal is identifying components. A descriptive document is not meant to say where each object is located, but which objects compose the document itself.
- **Super Descriptive (SD):** relaxing both constraints over cardinality and order, besides alternatives, designers can create abstract DTDs which consider any object as a sequence of repeatable and optional elements. These DTDs are meant to only define the set of objects of the documents.
- **(Un)Descriptive (UD):** relaxing any constraint designers could say that anything includes anything. Not useful in practice, those DTDs are only mentioned to complete our spectrum.

Table 1 shows a simple DTD declaration, accordingly to each model:

Table 1. Different levels of descriptiveness

Descriptiveness level	Content Model
PRE - Prescriptive	<!ELEMENT X (A, (B C), D*)>
DNA - Descriptive No Alternativess	<!ELEMENTX (A, (B?, C?), D*)>
DNC - Descriptive No Cardinality	<!ELEMENT X (A*, (B*, C*), D*)>
DNO - Descriptive No Order	<!ELEMENT X (A & (B? & C?) & D*)>
SD - Super Descriptive	<!ELEMENT X (A (B C) D)* >
UD - (Un)Descriptive	Any

On the basis of previous analysis [DIGV05], we identify the DNO paradigm as a good solution to describe documents' structures. Actually we did not cite directly DNO but we studied situations where such descriptiveness is enough to express everything users need. Moreover we proposed and discussed some patterns, concluding that by adopting these and only these patterns, all those descriptive situations could be covered.

3 Patterns for Descriptive Documents

The set of patterns we proposed in [DIGV05] is very small:

- **Marker:** an empty element, in case enriched with attributes, whose meaning primarily depends on its position within the context.
- **Atom:** a unit of unstructured information. An atom contains only plain text and is meant to indicate a specific role or semantics for that information
- **Block and Inline:** a block of text mixed with unordered and repeatable inline elements that, in turn, have the same content model. They are used to model any objects which ultimately carry the text written by the author.

- **Record:** a set of heterogeneous, unordered, optional and *non-repeatable* elements. Records are first used to group simple units of information in more complex structures, or to organize data in hierarchical subsets.
- **Container:** a set of heterogeneous, unordered, optional and *repeatable* elements. The name itself emphasizes the generality of this pattern, used to group diversified objects, repeated and collected together.
- **Table:** a sequence of homogeneous elements. Tables are used to group similar objects into the same structure and, also, to represent repeating tabular data.

A deeper discussion of each pattern is out of the scope of this paper (see our previous work for details) but some properties deserve some more space. First, patterns are *orthogonal*: each of them has a specific role and covers a specific situation, and no content model is repeated. Second, specific rules are imposed over the class of objects allowed in each content-model. For instance, an inline element can be contained only within a block, a container cannot directly contain plain text, a record or a table cannot be contained in a block, and so on.

Wrappers. Our approach relies on the methodical use of specific elements, called *wrappers*, which allow us to transform a generic DTD into a pattern-based one and to guarantee the *homogeneity* of content models. Every time a content model contains a mixed presence of repeated elements and single ones (or alternatives), a new (wrapper) element is created. It will substitute that 'wrong' declaration fragment, inheriting the content model. Consider for instance an element declaration of type `<!ELEMENT X (A,(B|C))>`; we don't want a sequence and a choice at the same time. Then we create a new element `W (<!ELEMENT W (B|C)>)` and, substituting it in the previous declaration, we obtain a homogeneous declaration `<!ELEMENT X (A,W)>`.

Moreover, the introduction of wrappers permits to "by-pass" all those situations where a constraint among patterns is violated. Consider for instance, a container element declared as `<!ELEMENT C (A|B)*>`, when `B` is an inline. A new block element, the wrapper `W (<!ELEMENT W (#PCDATA|B)*>)` can be created and the `C` definition can be changed in `<!ELEMENT C (A|W)*>`.

All changes introduced by wrappers are then targeted to "clean" (or homogenize) structures and to make documents more descriptive.

4 Formal Representation of Patterns

In this section we sketch out a formal analysis of the completeness of our patterns, based on language theory. Complete definitions and proofs can be found in [DDID07]. The basic idea consists of deriving properties of validation schemas by analyzing grammars which produce them, as proposed by [MLMK05].

We chose DTDs because they are more direct, but similar considerations could be extended to other languages like XML-Schemas [TDMM01] or RelaxNG [Mur00]. Although these languages are more powerful, in fact, creating and even reading the corresponding grammars would be much more difficult and

Table 2. Our pattern-based grammar P

[p01] elementdecl	::= markerelementdecl atomelementdecl blockelementdecl inlineelementdecl recordelementdecl containerelementdecl tableelementdecl
[p02] markerelementdecl	::= '<!ELEMENT' S MarkerName S markercontentspec S? '>'
[p03] atomelementdecl	::= '<!ELEMENT' S AtomName S atomcontentspec S? '>'
[p04] blockelementdecl	::= '<!ELEMENT' S BlockName S blockcontentspec S? '>'
[p05] inlineelementdecl	::= '<!ELEMENT' S InlineName S inlinecontentspec S? '>'
[p06] recordelementdecl	::= '<!ELEMENT' S RecordName S recordcontentspec S? '>'
[p07] containerelementdecl	::= '<!ELEMENT' S ContainerName S containercontentspec S? '>'
[p08] tableelementdecl	::= '<!ELEMENT' S TableName S tablecontentspec S? '>'
[p09] markercontentspec	::= 'EMPTY'
[p10] atomcontentspec	::= '(' S? '#PCDATA' S? ')'
[p11] blockcontentspec	::= maicontentspec
[p12] inlinecontentspec	::= maicontentspec
[p13] maicontentspec	::= '(' S? '#PCDATA' (S? ' ' S? maiName)+ S? ')**'
[p14] recordcontentspec	::= '(' S? mabrctName '??' (S? '&' S? mabrctName'??')* S? ')'
[p15] containercontentspec	::= '(' S? mabrctName (S? ' ' S? mabrctName)* ')**'
[p16] tablecontentspec	::= '(' S? mabrctName S? ')**'
[p17] maiName	::= MarkerName AtomName InlineName
[p18] mabrctName	::= MarkerName AtomName BlockName RecordName ContainerName TableName

time-consuming. More important, the vast majority of existing schemas proved to be structurally equivalent to DTDs [\[BMNS05\]](#).

We first define a grammar P (shown in Table 2) able to generate all the pattern-based DTDs. Productions [p01-p08] are used to declare the seven different patterns, while the remaining ones are introduced to specify their content models.

Note that we perform some simplifications to make simpler and clearer the analysis: we (i) omit attributes declarations, (ii) do not consider some unusual declarations as $(\#PCDATA)^*$ (equivalent to $\#PCDATA$), (iii) do not consider the terminal symbol '+' both for shortness and because it could be associated to the terminal '*' from a descriptive perspective. Note also that we introduce the terminal symbol '&', that in SGML syntax means that all elements must occur in any order, in order to better formalize the DNO model.

We then compare our grammar with a general grammar G, provided by the W3C [\[BPSMM00\]](#), that produces all the possible DTDs.

The result is that for each DTD, producible from G, it exists a pattern-based DTD, producible from P, which is equally descriptive at DNO level. To do it, we present a reduction algorithm, which applied to a DTD, generates a pattern-based DTD, equally descriptive at DNO level. Formally:

Proposition 1. *Given $L(P)$ and $L(G)$, let $r: L(G) \rightarrow L(P)$ be a function that implements our reduction algorithm; we want to state that*

$$\forall d \in L(G) \exists p \in L(P) \ni d \xrightarrow{r} p \quad (1)$$

with p and d equally descriptive at DNO level.

The symbol \xrightarrow{r} indicates that d is reduced to p applying the function r .

The proof is a case-by-case analysis of the production rules of G , and, when needed, a reduction operation (based on wrappers insertion) to transform those rules into descriptive ones. During this reduction process, we relax some constraints, prescribed in grammar G ; in this way, the set of documents, accepted by the pattern-based DTDs, generated by P , is at least large as the set of documents generated by the original DTD.

5 Conclusions

In this paper we discussed how any document *structure* can be described by a very small set of objects and composition rules. Moving off an analysis of different levels of descriptiveness, we presented a grammar-based formalization of our patterns and some sketches of a formal proof of their completeness (whose extended version can be found in [DDID07]). In the future, we plan to inspect, in the same formal way, other properties like correctness and minimality.

References

- [BMNS05] Bex, G.J., Martens, W., Neven, F., Schwentick, T.: Expressiveness of xsds: from practice to theory, there and back again. In: WWW '05: Proceedings of the 14th international conference on World Wide Web, pp. 712–721. ACM Press, New York (2005)
- [BPSMM00] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E.: Extensible Markup Language (XML) 1.0 (2000), <http://www.w3.org/TR/REC-xml>
- [DDID07] Dattolo, A., Di Iorio, A., Duca, S., Feliziani, A.A., Vitali, F.: Patterns for descriptive documents: a formal analysis (2007), [ftp://ftp.cs.unibo.it/pub/techreports/2007-13.pdf](ftp://ftp.cs.unibo.it/pub/techreports/2007/2007-13.pdf)
- [DIGV05] Di Iorio, A., Gubellini, D., Vitali, F.: Design Patterns for Descriptive Document Substructures. In: Proceedings of the Extreme Markup Conference, Montreal, Canada (2005)
- [MLMK05] Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of xml schema languages using formal language theory. ACM Trans. Inter. Tech. 5(4), 660–704 (2005)
- [Mur00] Murata, M.: Relax (REgular LAnguage description for Xml) (2000), <http://www.xml.gr.jp/relax/>
- [Pie01] Piez, W.: Beyond the descriptive vs. procedural distinction. In: Proceedings of the Extreme Markup Conference, Montreal, Canada (2001)
- [Ren00] Renear, A.: The Descriptive/Procedural Distinction is Flawed. Markup Languages: Theory and Practice 2(4), 411–420 (2000)
- [TDMM01] Thompson, H.S., Beech D., Maloney, M., Mendelsohn, N.: XML Schema Part 1: Structures (2001), <http://www.w3.org/TR/xmlschema-1/>
- [Via01] Vianu, V.: A web odyssey: from codd to xml. In: PODS '01. Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 1–15. ACM Press, New York, NY, USA (2001)

Component-Based Content Linking Beyond the Application

Johannes Meinecke¹, Frederic Majer¹, and Martin Gaedke²

¹University of Karlsruhe, Institute of Telematics,
Engesserstr. 4, 76128 Karlsruhe, Germany
{Meinecke,majer}@tm.uni-karlsruhe.de

²Chemnitz University of Technology, Faculty of Computer Science
Straße der Nationen 62, 09111 Chemnitz, Germany
gaedke@cs.tu-chemnitz.de

Abstract. The content of many innovative Web sites today often originates from beyond the application. This paper is concerned with building Web applications that heavily integrate and link content from external sources, like e.g. Web services or RSS feeds. Unlike conventional applications, they are characterized by a very dynamic and distributed information space. In this context, traditional Web Engineering approaches suffer from the fact that they rely too much on a-priori knowledge of existing content structures. We present a support system and a method for building such applications in a very flexible way. Flexibility is achieved by managing links separately from the content in a dedicated Web service and by composing the application from fine-grained, reusable components that realize navigation, presentation, and interaction for the linked content.

Keywords: Web Engineering, Content Linking, Web Services, Reuse, Triple Stores.

1 Introduction

Today, Web Engineering is challenged by the construction of a wide variety of Web application types. As one important trend, sites are evolving from isolated content providers to functionality providers that are connected to other parts of the Web beyond simple HTML linking. In particular, Web applications combine their functionality with external services and the content contributed by large communities of participating Web users [11]. Prominent examples include sites like Google Maps, Flickr or del.icio.us that, in addition to offering a visible user interface, also provide programmable interfaces in the form of Web services to enable third parties to build applications on top of them (also referred to as *mash-ups*).

However, the existence of standardized Web service technologies only solves a part of the problem. Beyond just integrating content as e.g. separate sets of resources, real added value is only achieved, when resources from different sources are linked to each other. Here, *link* is understood as the representation of a semantic connection

between two resources, i.e. not exclusively related to navigation. We are especially interested in the question of how to account for this linked content when developing Web applications. In particular, we are addressing the following three challenges:

- **Continuous extensions with new content sources:** Unlike content sources that are under the control of the application provider, the set of autonomous sources that are potentially relevant changes frequently, as old services become obsolete and new services become popular. Hence, the development process must be capable of accounting for originally unknown sources, which are integrated and linked to the existing content later. Ideally, such changes should not require re-implementing or re-generating the system at code level, in order to achieve short revision cycles.
- **A repetitive implementation effort for content source linking:** Web developers perceive *getting content from others* as one of the most frequent problems in Web development work [13]. Concrete problems include distribution, caching, support for multiple service interfaces, or the realization of navigation across the linked content. On the other hand, a considerable part of this functionality does not depend on the application domain, but is rather general in nature. This raises the question of how we can package generic functionality in reusable components that abstract from specific applications.
- **Content sources that are unprepared to be linked.** As they originate from different organizations, they were most likely developed independently from each other. Without the means to make changes to the external sources, the linking structure has to be imposed retrospectively. Similarly, little can be assumed in terms of service capabilities. The interfaces may vary and only provide for basic data retrieval operations, instead of e.g. semantic query features.

In this paper, we present a support system and a method that aims at flexible solutions for these challenges by assembling applications from reusable components and services. In section 2, we introduce an example scenario to clarify the scope of our work. Section 3 presents the Linkbase method together with its individual activities and the architectural outline of the resulting applications. We then describe the implemented 5 contains a brief overview of related work. We conclude with a summary and an outline of planned future work in section 6.

2 The Tourism Portal Scenario

To characterize the targeted type of dynamic Web application, we begin by introducing a fictitious scenario. The scenario is concerned with the development of an advanced Web portal for tourists to aid them in planning their trips in advance. For this purpose, the portal provides reports of other people, who have already visited the travel destinations, as well as all sorts of additional tourist-relevant information. As the planning of trips generally requires accurate and up-to-date information, which the site provider organization cannot gather itself, the envisioned application is supposed to rely on the integration of Web services and other resources offered by third parties. Examples of relevant content include public geographical information about places, locally relevant news from multiple providers, weather forecasts, and events from

multiple calendars. As an example for content that is generated and stored at the site itself, the user community contributes their travel experience in the form of reviews or travelogues. Furthermore, information about relationships between the users of the portal, acquired from the service of a networking portal, can be exploited. The last case of content integration is based on an agreement of the tourism and the networking portal to share services and accounts, and thus allow their respective users to log on to each other's sites through single-sign-on mechanisms.

Fig. 1 gives a simplified overview of the possible content structure inside the portal, as the provider might initially plan it. As indicated, most content originates from outside the portal. In order to make the content more useful, the portal needs some additional knowledge about how the individual resources relate to each other. For example, descriptions about places can be linked to most other resources, as to specify where a given news article is relevant or where a given event occurs. Users are related to reports they authored, photos they took, events they participated etc. Furthermore, there are links that apply to resources of only one source, as e.g. a geographical hierarchy on the places, or the relations among the users in their networking communities.

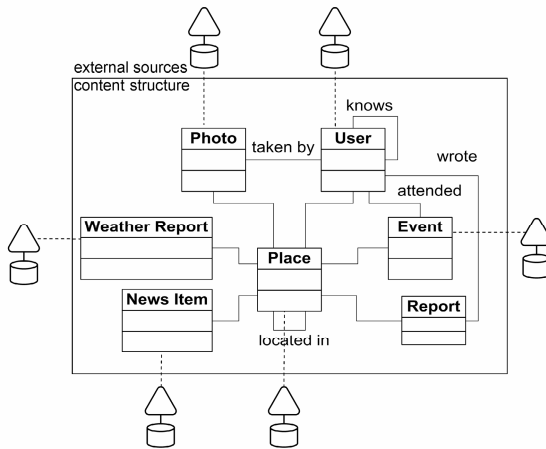


Fig. 1. Content in the Tourism Portal Scenario

Based on the linked content, the tourism portal can offer functionality beyond just letting users browse and contribute reports. For example, users cannot only find out, what other people say about a certain place, but also, whom of their friends they can ask for personal travel experiences, or where their friends will travel within the next year. The benefits produced by the portal can in turn be made available as services to other applications.

3 The Linkbase Method

This section presents the Linkbase method for building applications by linking autonomous content sources with the help of a support system. Fig. 2 gives an overview of the implementation architecture and the steps to be performed. The central idea is to

manage links between arbitrary resources with the help of a central Web service (the Linkbase service) in a uniform way. Inspired by fundamental principles from the Semantic Web, links are seen as triples, i.e. labeled connections between two resources. A resource is anything that can be described with a Uniform Resource Identifier (URI), including conventional Web resources accessible via HTTP and resources queried from Web services. The Web application itself is composed of generic, reusable components that operate on the links from the Linkbase and the content from the different sources to provide navigation, presentation, and interaction to the user.

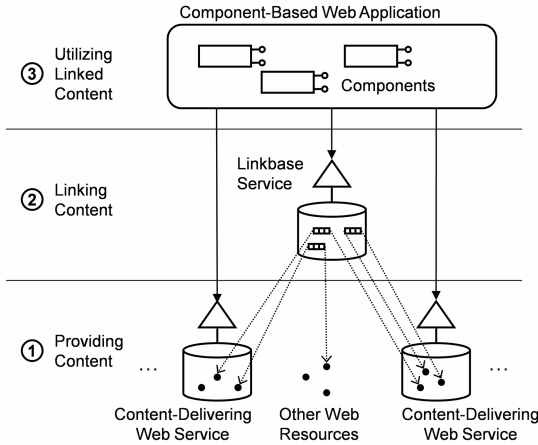


Fig. 2. Architectural Overview of the Linkbase Method

In the following subsections, we give more specific definitions of the concepts in Fig. 2 and discuss three groups of activities that are necessary for building applications. The information space has to be prepared for integration, the Linkbase has to be established, and the Web application has to be assembled. These guidances should not be seen as one-time activities, but more as tasks that become necessary every time the application is extended. Rather than re-implementing the system in every cycle, the Linkbase method focuses on reconstructing only the necessary parts (services and components), as well as their integration with the remaining system at runtime.

3.1 Providing the Content Sources

The aim of the first step is to fulfill the preconditions for the content sources to be linked to each other effectively by making them available in a uniform way. In the case of conventional Web resources¹, the existing Web standards already provide such uniform access methods, as e.g. addressing images via URLs over HTTP. Alternatively, Web services offer more advanced capabilities, e.g. related to querying for specific resources. In this paper, we use the term (content-delivering) *Web service* for

¹ In the sense of RFC 3986.

services that provide access to content to other services and applications via standardized Web protocols, as e.g. SOAP over HTTP or REST.

Along with the more advanced capabilities of Web services as content sources comes a wider choice of access methods, particularly in terms of Web service interfaces. As this variety stands in the way of an effective reuse of content, we propose to use a uniform interface that abstracts from the particular type of content delivered by the service. If the application provider organization controls the service, it can implement the interface directly. Otherwise, the provider creates a wrapping Web service, which is accessed with the uniform interface, and which in turn delegates the content requests to the proprietary interfaces of the third-party services. As one alternative, we propose the specification of an interface in accordance to the CRUDS metaphor [8]. CRUDS allows the querying and manipulation of arbitrary sets of content objects through the operations *Create*, *Read*, *Update*, *Delete* and *Search*. In cases where there is just read-only access, it is sufficient to implement the Read and Search operations. The interface is independent of the content type, as the method signatures contain XML parameters for passing over content objects. Thus, this service provides uniform access to e.g. databases, legacy systems, or, like in our case, other third-party services.

As a second precondition for making the service a part of the information space (in which resources can be linked to each other), we require a method for addressing the supplied resources uniformly. The object identifiers used in the CRUDS interface are inappropriate, as they are not guaranteed to be globally unique. In order to overcome this disadvantage, we propose their extension to so-called *Information Space Identifiers* (ISID). ISIDs are special types of Uniform Resource Nominators (URN) that contain, in addition to the local object identifiers of the resource, also an identifier of the service supplying the resource. The general format of an ISID is:

```
urn:isid:[service id]:[object id]
```

The service identifiers should correspond to the same identifiers that are already used to refer to the service within registries (like e.g. UDDI). Consequently, an application can resolve an ISID by first querying the service's URL from a registry (if not already known) and then invoking its *Read* method with the contained object identifier as a parameter.

Referring to the example in section 0, the first step of the Linkbase method would be to develop a new CRUDS service for handling travel reports, and to create wrapper services for the remaining content sources (weather report, events, etc.). If photos are integrated without any metadata (as e.g. provided by a Web service of a photo sharing site), they can be addressed by URLs, requiring no further implementations. In the other cases, the wrapping Web services introduce new identifiers. For example, an event could be identified with:

```
urn:isid:0a816d35-05fc-41eb-8a76-d54c111c8d2f:20070508-001
```

Here, *0a816d35-...* identifies the wrapping calendar service, and *20070508-001* is the identifier by which the event is known at the external service.

3.2 Linking the Content

The second step of the Linkbase method addresses the actual linking of the resources, within the unified information space. For this purpose, we propose using a *Linkbase service*, which we define as a Web service that provides at least read- and write-access to a set of links, where each link must at least contain a triple of URIs. Each triple consists of a subject, a predicate, and an object. In our case, the subject and object URIs refer to resources provided by the autonomous data sources, and the predicate URI serves as a label for the type of relationship. For the realization of the Linkbase service, there already exists a wide variety of triple stores, many of them related to Semantic Web projects [1], that implement the functionality for storing and managing triples.

Generally, triple stores do not only support storage and retrieval, but also the computation of new triples that logically follow from others. Based on an ontology that captures the necessary knowledge about the types of resources and relationships, the store can infer new facts from old facts. The Linkbase method focuses not so much on storing facts between arbitrary (virtual) concepts, but rather on storing links between concrete resources on the Web. However, reasoning can still be valuable in relieving the application from the burden of computing links by itself. For example, we can declare the *located in* relationship from Fig. 1 as transitive, to better reflect the geographical hierarchy.

So far, we have been concerned with content originating from external sources. In addition to that, it may also be necessary to retrieve the information about which resource is linked to which from the outside. We propose an extension of the triple store concept to not only provide stored and inferred triples, but also triples extracted from external sources at runtime (cf. Fig. 3). When such an extended store receives a query for links, it identifies an appropriate source (like a Web service), queries that source, and returns the result as a set of triples. The idea resembles in some ways the concept of distributed triple stores, where the triple space spreads across several connected stores. However, in this case, the external sources are not restricted to triple stores, but can e.g. also include legacy systems or third-party Web services.

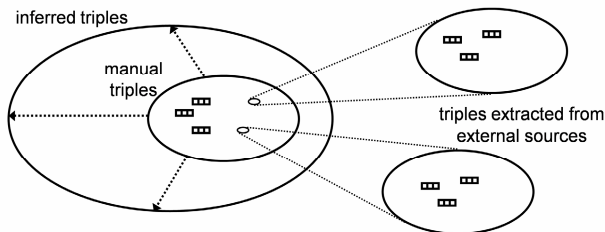


Fig. 3. Inferred and Extracted Triple Space Extensions

With the mentioned extension, there exist three principal ways of inserting links:

- Links are manually created from inside the application itself, e.g. according to information given by a large community of users.
- Links are automatically extracted from external sources at runtime. This is especially applicable when the set of links is very large or changes frequently.

- Links are imported from external sources once, e.g. by applying batch tools. This is preferable in cases where the source's availability is unreliable, or where the procedure of extracting the links takes too long to be performed at runtime.

In the tourism portal scenario, the step *linking the content* comprises setting up one Linkbase service that handles instances of all relationships from Fig. 1. For example, the *taken by* relationship could result in triples like:

```
<http://photo.site/photos/P1010023.JPG>
<http://purl.org/dc/elements/1.1/creator>
<urn:isid:77ff72af-...:smithj>.
```

Here, the triple links the URL of a photo with a description of a person supplied by a Web service. Assuming that the photo sharing site supplies this information, a plug-in can be developed for the Linkbase service that extends the triple space with always up-to-date information about the photos that people took. In contrast, the *located in* relationship is relatively stable and can therefore be supplied by a singular import step. The *wrote* relationship is an example for triples created by the application itself, e.g. every time a user contributes a new report.

3.3 Using Linked Content in the Application

Within the third step, the goal lies in allowing developers to construct the (frontend) Web application without having to program it. To achieve this, the Linkbase method focuses *Component-based Web applications*, which we understand as dynamic Web sites that are composed of server-sided, reusable components, usually assembled with the help of a framework. An example for such a framework is our previous work, the WebComposition Service Linking System (WSLS) approach [4], which we adapt to the challenge of linking content from autonomous sources. The idea behind WSLS is to provide a runtime environment for visual, interactive components. At runtime, WSLS allows application developers and administrators to place, rearrange and configure components on pages without recompiling the application. In order to support separation of concerns, components are developed as fine-grained implementation artifacts that can be combined with each other by following the Decorator software design pattern [5].

For the Linkbase method, we supplemented the WSLS approach with a catalogue of components that are dedicated to dealing with linked content. Since both links and content sources are provided in a uniform way, we can restrict the number of components to be implemented and focus on generic functionality. In the following, we present three components from our catalogue to exemplify the support for the three aspects navigation, presentation, and interaction: the Fisheye, the Timeline, and the Content Connector. To be of practical use, they have to be complemented with additional components, like the ones described in [4]. This includes particularly a component that retrieves and caches the content objects from the Web services and that supports the Web service interface chosen for unification (e.g. the CRUDS interface).

Fisheye Component (Aspect Navigation): The Fisheye allows users to navigate through the graph formed by the Linkbase, along selected types of links. This is achieved by decorating the presentation of a currently active object with smaller

navigatable preview presentations of related objects around it (cf. Fig. 4). The name *Fisheye* relates to the impression that the view skims over a web of objects, where the object in focus is always magnified. Technically, the component queries the Linkbase for any resources linked to the currently active resource. When a user activates a hyperlink, the Fisheye component notifies the content-supplying component of the next object to focus.

In order to customize the component for a concrete use case, the configuration has to specify, over which types of links the navigation should occur. In general, it is more advisable to restrict the navigation on a subgraph formed by certain types of connections. Depending on the types, we have to configure, where a related object should be placed (underneath, above, to the left, to the right ...) and how it should be rendered (e.g. with a template).

Example: Given a Linkbase that contains a geographical hierarchy, the Fisheye can be configured to display hyperlinks to geographical places (continents, countries, regions, cities) above, underneath or next to the currently selected place X, depending on whether the places contain, are contained by, or are situated near place X. A similar navigation support can be provided to browse through a network of friends.

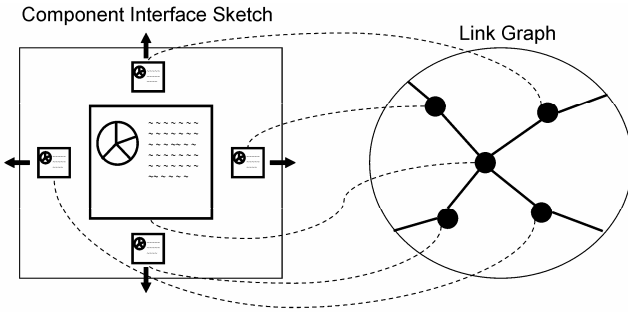


Fig. 4. Fisheye Component

Timeline Component (Aspect Presentation): The Timeline visualizes objects related to a given context in time. To this end, it decorates the presentation of a central object with a time axis and bars that represent the related objects and the time span of the relation (cf. Fig. 5). Similar to the Fisheye, the Timeline queries the Linkbase for all links connected to a particular resource, and renders small presentations of the related objects next to the time bars. The time spans are retrieved either directly from the Linkbase (if the Linkbase supports time-based links [6]) or from common metadata attributes contained in the objects that are supplied by the CRUDS service.

Besides presentation aspects of the timeline axis and bars, the configuration has to specify, how to render the related objects and how to retrieve the time spans.

Example: An application of the Timeline in the running example is to give an overview of submitted travel reports, that relate to a specific travel destination. Alternatively, another interesting view would be to display selected photos and travel reports related to the user or any friend of a user, in order to provide a personalized travel history line.

Content Connector Component (Aspect Interaction): The Content Connector facilitates the interaction between the user and the links by allowing the user to add new links with a single mouse click (cf. Fig. 5). Again acting as a decorator, the component adds hyperlinks or buttons to the presentation of content objects. When activated, it creates a new link in the Linkbase between the decorated object and another object, according to context and configuration. This other object can e.g. be the account of the logged-in user or a pre-defined URI. Alternatively, the user selects an object within a second step, from a choice of objects queried from a CRUDS service.

The configuration determines how the rather technical act of adding a link is presented to the user. Furthermore, it must specify the type of link to add (i.e. the triple predicate URI) and what to link to (see above).

Example: Applied to a list of travel destinations, the Content Connector can provide “Been There”-Buttons, that allow portal users to mark the places they have visited and the tourist activities they have attended on the fly.

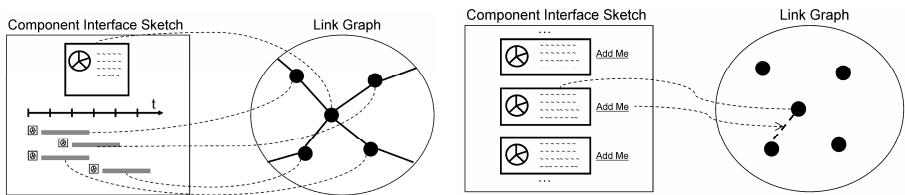


Fig. 5. Timeline Component and Content Connector Component

Based on the described component-based Web application architecture, the third step of the Linkbase method comprises a number of activities to be conducted that follow the principle configuring instead of programming. First, the components required for the construction or extension of the application are provided, either by developing them or by falling back on existing components (e.g. from a component repository). Components and content sources are then registered at the framework, and thus become potential building blocks for the application. Following that, the visible part of the Web application is created or extended by configuring new pages and page sections where the components are instantiated. In other words, the components become responsible for specific parts of the application’s hyperspace. Finally, the generic components are configured in accordance to their intended purpose within these sections. This includes as a vital step the wiring of the component to the registered content sources. Another example of an aspect to be configured is the specification of templates for rendering content objects. Apart from the development of new components, all activities involved in assembling the application are supported by the framework’s Web interface.

4 The Linkbase Applied

The Linkbase method relies on support by system to guide the application developer. In the following section, we present the support system we implemented to validate

our approach, including a triple store that realizes the extensions discussed in section 0. The Linkbase support system has been used to realize a number of scenarios that demonstrate the applicability of its standard building blocks in different situations.

4.1 Experiments with an Implemented Support System

The central component of the Linkbase support system is the Linkbase service. Rather than using an existing triple store, we implemented a Linkbase service especially with dedicated support for our purposes. The main reason for this was to be able to dynamically extend the triple space with triples extracted from external sources, which is not supported by existing triple stores, and to experiment with extensions to the triple schema (like e.g. time-based triples).

Fig. 6 displays the architecture of the implemented Linkbase that was developed as an ASP.NET Web service. To the outside, the service provides a CRUDS interface, which applications can use for querying (Read, Search) and modifying (Create, Update, Delete) the triple space. Hence, the same service interface can be used to access both the content objects and the links between the objects, resulting in additional re-usability. To provide rudimentary support for reasoning, an inference module was included that computes and stores implied triples every time the triple space is changed. Thus, the Linkbase takes into account properties of link types, which can be defined in an OWL-XML ontology file. For example, the following entry has the effect that whenever a person is linked to an acquainted person, a link is automatically generated in the opposite direction:

```
<owlx:ObjectProperty
owlx:name="http://xmlns.com/foaf/0.1/knows
owlx:symmetric="true" />
```

This declaration of link types and knowledge about the linked domain is optional, i.e. links with new types can be added to the Linkbase anytime. Additional triples are generated by the plug-in module, which delegates queries for selected link types to external sources. As the mechanism for extracting external links may vary from source to source (e.g. due to different Web service interfaces), it is performed by plug-ins that are added to the Linkbase at runtime.

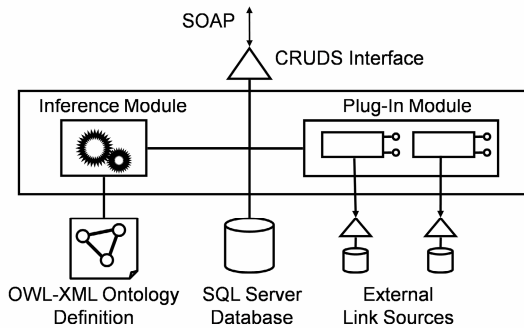


Fig. 6. Linkbase Service Implementation Architecture

To validate the applicability of the support system, we used its various components for the construction of a Web application. This development was conducted in the context of the project Software Engineering for Information Appliances at Home, whose outcome included a Web portal targeted at families at home. Relating to the step *providing the content sources*, we developed a number of CRUDS Web services. Some of them were generated with the help of a custom Visual Studio plug-in, as e.g. a service for handling personal profiles of family members. Others were programmed to wrap existing non-Web systems, as e.g. a calendar service that provides read- and write access to appointments from a Microsoft Exchange server. For example, we created a CRUDS Web service that provides slides as individually addressable content objects (JPG images and metadata entries). Then, we filled it with slides from existing PowerPoint presentation files, including a lecture of 800 slides.

Following that, we set up the Linkbase service to provide the links according to the three principal ways described in section 0. For instance, the links that make up the relationships between the family members are specified by the portal users themselves during the lifetime of the Web site. Here, we took advantage of the built-in reasoning support, which in this case supplements the user’s input to work out any implied relationships. As an example for a dynamically extracted link type, we developed a plug-in that queries a public Web service of the popular photo sharing site Flickr, to provide links between photos and concepts represented by tags. In the case of the slide service, the hierarchical structure of the presentations (chapters, sections, learning units...) was converted to links during the initial import step.

The visible part of the Web application was exclusively assembled and configured from components, i.e. no code was written to wire the components. Fig. 7 contains a screenshot of the application in administration mode, together with an extract from the configuration screen. The depicted page section is composed of a combination of four components: a CRUDS component to communicate with the profile Web service, a template presentation component to render the personal profile in HTML, a navigation component to provide previous- and next-buttons, and finally, a timeline component, to decorate the person with an overview of related photos and events.

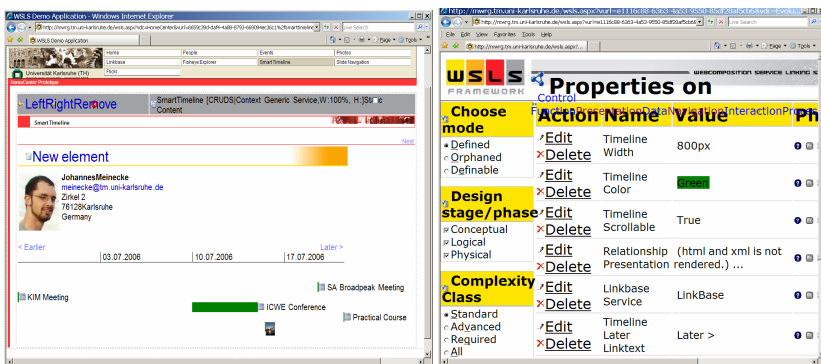


Fig. 7. Timeline Component: Events and Photos

Fig. 8 shows the Fisheye component applied for two different purposes. In the first screenshot, it supports the user in navigating through a family tree and browsing the profiles of family members. According to the specific configuration, the Fisheye places links to parents above, links to children below and links to partners next to the current profile. In the second screenshot, the same concept is applied to realize a presentation slide browser. Here, the horizontal dimension lets users skim through slides on the same level, while the vertical dimension lets them move up and down the lecture hierarchy, e.g. in order to return to the title slide of the current section.

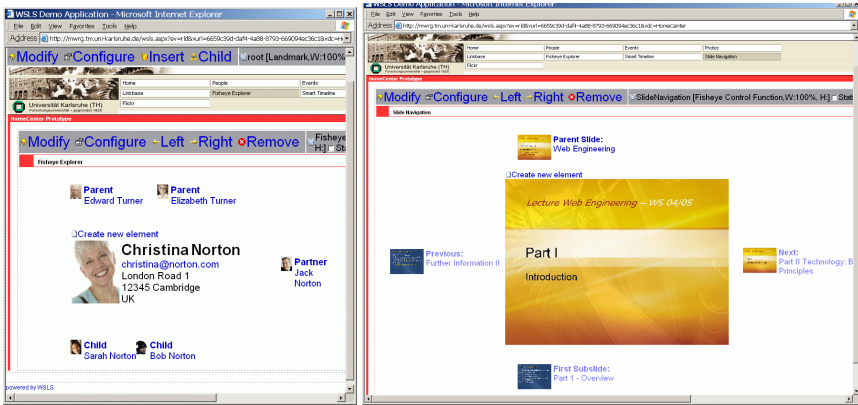


Fig. 8. Fisheye Component: Family Tree and Lecture Slides

All involved content sources were initially unprepared for linking. Some of them were even beyond our control in a sense that we could not influence their content structure (like of the Exchange Calendar or the Flickr Web service). Although programming was necessary, the implementation was limited to generic, application-independent components that can be reused later on to save effort in the long run. Rather than planning everything from the very beginning, we extended the application in several cycles: The first version only featured family members and photos; then the calendar was added, and later on the presentation slides. Hence, the chosen scenario corresponds to the three issues in content linking identified in the introduction.

4.2 Lessons Learned

While the project's outcome confirmed the approach's suitability to meet the targeted problems, we also gained experience related to implementation issues and potential for improvement. Concerning the data model of the link, applications can benefit from extensions of the basic triple concept. Additional information, like the already applied integration of time intervals or the inclusion of standard metadata attributes enable more advanced ways of using the links.

To increase performance, redundant information about the linked resources can be stored with the links in cases where the information is unlikely to change. This proved especially helpful, as it drastically reduced the frequency at which components had to

query the linked resources. Beyond that, the need for caching at the application to achieve adequate response times became evident. Within the project, this was achieved with the help of the WSLS framework, which caches content retrieved from CRUDS Web services at the granularity of individual content objects. For example, in order to render an object that is linked with l other objects inside a Fisheye or Timeline component without caching, the Linkbase is queried once and the content-supplying service $(l+1)$ times (or once, if the triples contain enough information for rendering the previews). While these measurements can ease performance problems caused by the Web service communication overhead, the Linkbase remains a bottleneck in the architecture. As the number of linked sources grows, it may become necessary to distribute the Linkbase (e.g. as proposed in [3]), which was however beyond of the scope of the presented work. As a major advantage, the Linkbase architecture provides for a great degree of flexibility, accounting for changes to the application's information space from the very beginning.

5 Related Work

In the following, we give a brief overview of several approaches that are related to the challenge of linking autonomous content sources in Web application development.

Several model-oriented Web Engineering methods address the aspect of external content and services. For example, WebML, a visual language for the specification of data-intensive Web applications, has been extended for the model-driven development of Web applications that consume web services as data sources [2]. The focus here lies on applications with most of the data stored in a database (i.e. under the control of the provider), whereas the external content only supplements this data. Directly opposed with respect to this aspect, the HERA methodology targets distributed Web-based information systems where data is retrieved from Web sources with semantic query capabilities [16]. While this approach enables the user to perform very intelligent search capabilities over heterogeneous sources, it requires relatively high development effort for ontology integration, even in cases where such queries are not necessary. The OOWS method has been conceptually extended with support for Web Services [14] and content aggregation [15]. Generally speaking, there is a discrepancy between the very systematic, model-based development methodologies and the dynamic nature of the information space exploited by modern Web applications.

From the architectural point of view, the idea of managing content links in a separate service is closely related to Open Hypermedia Systems (OHS) and the concept of the *link service* found there [12]. This relatively old idea has also been applied to Web technologies [3] as well as to Web services in particular [10]. In the context of this field, our approach can be understood as an application of the OHS concept to a unified information space on top of Web services, and as an alternative method for exploiting the links in a reusable manner. Related to our idea of dynamically extending the triple space, [10] proposes the storage of dynamic hypermedia links that contain queries instead of rigid identifiers, in order to realize stable links on a evolving information space. While this approach does not explicitly target Web services, it could be combined with ours by delegating these queries to the (unified) Web services.

With respect to the goal of composing functionality and content from existing Web-based systems, our approach is related to the field of Semantic Web services. As one example, the WSMX system supports the execution and combination of semantically described Web services [7]. This approach is very powerful, as the system can dynamically chose and combine appropriate services to fulfill goals given by the requestor. However, it is less suitable for dealing with the specific problem of handling and linking sets of resources provided by these services in a uniform way. As an example for a more manual approach to application composition, [9] proposes the end-user-driven definition and wiring of components that wrap existing Web applications. The idea has advantages in cases where there is no Web service interface to build on, but entails the same problems that are common to all Web site wrapping strategies, e.g. in terms of sensitivity to layout changes on the wrapped sites.

6 Conclusion

The contribution of this work was a novel way of constructing Web applications that integrate content from external sources. The development of such applications require very flexible implementation techniques, as the set of content sources to be integrated is often unknown at design time and is subject to changes later on. In this paper, we described a way to support application development with a support system, whose major component is a Web service for providing links between content objects from different sources (the Linkbase service). To apply this system, we first unify the application's information space by introducing a standardized interface for (wrapping) Web services, and a standardized way of addressing content objects. Content objects are linked by triples of URIs, stored by the Linkbase service. For the application architecture, we propose to assemble Web sites from configurable generic components that work with the unified content sources and links.

We are now working on variations of the triple data model, in order to examine the added value gained by augmenting the triple with metadata. Another open question for future research is the applicability of older distribution concepts for link services to the problem at hand to better address performance issues. Furthermore, dynamic aspects of the Linkbase, like change management, remain to be investigated, also taking into account existing approaches.

Demonstration Videos

Demo videos of the described support system are available for download at the MWRG homepage, <http://mwrgrg.tm.uni-karlsruhe.de/downloadcenter/systems/demos>.

Acknowledgements

This material is partially funded by Microsoft Research Cambridge, within the context of the research project 2005-053.

References

1. Beckett, D.: SWAD-Europe Deliverable 10.1: Scalability and Storage: Survey of Free Software/Open Source RDF storage systems -, W3C: (12-10-2006)http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/
2. Brambilla, M., et al.: Model-driven Development of Web Services and Hypertext Applications. In: SCI2003, Orlando, Florida (2003)
3. Deroure, D., et al.: A Distributed Hypermedia Link Service. In: Third International Workshop on Services in Distributed and Networked Environments: IEEE, pp. 156–161. IEEE Computer Society Press, Los Alamitos (1996)
4. Gaedke, M., Nussbaumer, M., Meinecke, J.: WSLs: An Agile System Facilitating the Production of Service-Oriented Web Applications, in Engineering Advanced Web Applications, Matera, S.C. M. (eds.), Rinton Press, 26-37 (2005)
5. Gamma, E., et al.: Design patterns: elements of reusable object-oriented software. In: Addison-Wesley professional computing series, Reading, Mass., vol. xv, p. 395. Addison-Wesley, London, UK (1995)
6. Gutierrez, C., Hurtado, C., Vaisman, A.: Temporal RDF. In: European Conference on the Semantic Web (ECSW'05), pp. 93–107 (2005)
7. Haller, A., et al.: WSMX-a semantic service-oriented architecture. In: 2005 IEEE International Conference on Web Services (ICWS 2005), Orlando, Florida, pp. 321–328 (2005)
8. Ibm: Elements of Service-Oriented Analysis and Design -, IBM Homepage: <http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/>
9. Ito, K., Tanaka, Y.: A visual environment for dynamic web application composition. In: ACM Conference on Hypertext and Hypermedia, Nottingham, UK, pp. 184–193. ACM, New York (2003)
10. Karousos, N., et al.: Offering Open Hypermedia Services to the WWW: A Step-by-Step Approach for Developers. In: Twelfth International Conference on World Wide Web, Budapest, Hungary, pp. 482–489 (2003)
11. O'reilly, T.: What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software - (18.10.2005) Online Article: <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
12. Pearl, A.: Sun's Link Service: A Protocol for Open Linking. In: 2nd Annual ACM Conference on Hypertext, Pittsburgh, USA, pp. 137–146. ACM Press, New York (1989)
13. Rosson, M.B., et al.: Designing for the Web Revisited: A Survey of Informal and Experienced Web Developers. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 522–532. Springer, Heidelberg (2005)
14. Ruiz, M., et al.: A Model Driven Approach to Design Web Services in a Web Engineering Method. In: 1st International Conference on Advanced Information Systems Engineering Forum (CAiSE Forum), Porto, Portugal (2005)
15. Valderas, P., Fons, J., Pelechano, V.: Extending Navigation Modeling to Support Content Aggregation in Web Sites in Fourth International Conference on Web Engineering (ICWE'04). In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 98–102. Springer, Heidelberg (2004)
16. Vdovjak, R., Barna, P., Houben, G.J.: Designing a Federated Multimedia Information System on the Semantic Web. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 357–373. Springer, Heidelberg (2003)

A Double-Model Approach to Achieve Effective Model-View Separation in Template Based Web Applications

Francisco J. García¹, Raúl Izquierdo Castanedo², and Aquilino A. Juan Fuente²

¹ Departamento de Matemáticas y Computación (Universidad de La Rioja)
C/ Luis de Ulloa s/n
26004, Logroño (La Rioja), Spain
francisco.garcia@unirioja.es

² OOTLab: Laboratory of Object Oriented Technology (Universidad de Oviedo)
C/ Calvo Sotelo s/n
33005, Oviedo (Asturias), Spain
{raul, aajuan}@uniovi.es

Abstract. Several works [20,22] have tried to enforce strict isolation between the model and the view in template based web applications by restricting the computing possibilities of the used templates. From the point of view of graphic designers this is a limitation that may make their work difficult. Besides, in this paper we state that this claimed strict isolation is impossible to achieve in practice for HTML template systems. We propose another approach to study and to attain an effective separation between model and view that does not necessarily restrict the expressive power of the template: the double-model approach. Finally we present an implementation of this approach in a renewed template system called JST2.

Keywords: template systems, Web applications, software architectures, MVC.

1 Introduction

When developing a web site with dynamic content, instead of writing the HTML code directly to the client's browser, developers have tended to make the most of some kind of *template system* (language and engine). Doing so, long and tedious code writing tasks can be alleviated and, a priori, among other undeniable advantages, it promotes the separation between presentation and business logic and it increases the clarity and maintainability of the application. The problem is to decide which is the more suitable template system, and, what are the more helpful features expected to be present in the template system. The answer, as usual, depends on the situation. But simplifying we can find two non exclusive tendencies: to provide the template system with a complex set of aids to reduce the programming effort, or to provide a reduced set of functional features, but in turn, contribute to force the separation between presentation and business logic.

The web developer's community has released plenty of template systems (at the time of writing, a simple search at SourceForge.net of the term "template engine" produces 214 results). We could define a template, in the context of web applications,

as an HTML document in which several placeholders have been inserted. At execution time, the template engine replaces these placeholders by the actual dynamic content taken from the application. The eternal promise of template systems is that the template meets the rest of the application only at the end of the development. That is, in theory, graphic designers work on the templates completely independent from application programmers. But in practice, this is not true for most template systems.

The way the template placeholders are replaced by actual content has been evolving as the problems related to code inclusion in the template have been detected. In fact, the first template systems we can refer to (JSP [26], ASP or PHP [23]) allow the inclusion of code in the template. This code is programmed in the same language as the application, and it can gain access to the application in order to obtain the actual data necessary to generate the dynamic content. The possibility of placing code in the templates is a temptation that developers can hardly overcome. It is a back door for the inclusion of business logic in the template, a practice that involves evident coupling problems between business logic and presentation. These problems have been already evidenced, e.g. in relation to JSP, in the classic reference of Hunter [15].

The coupling problems can also be studied from the point of view of the relationship maintained between both parts of a typical web application development team: the graphic designers and the programmers. The use of this kind of template systems makes this relationship very close, and it is characterized by the constant flow of information from programmers to designers, who need to know what the application is like in order to program the access to it in the template. This need of communication have been already treated by us [16,17] in the context of XML-XSLT [30], but can be perfectly extrapolated to other environments.

A second step in the template systems evolution leads us to systems in which the inclusion of native code in the template is not possible. This fact eliminates the chance of accessing the application business logic, but does create other subtle coupling problems. Often, these template systems force the designers to learn a completely new, but equivalent, pseudo programming language in order to develop the template documents: e.g.: XSLT [30], FreeMarker [11], WebMacro or Velocity [31,3], Zope Page Templates [34] or Smarty [25]. Other systems, in order to be used, make necessary the development of special classes or adapters (Tapestry [2] or Tea). Designers are also compelled to use development tools that they are not familiar with. In our opinion one of the main objectives of any template system must be to allow designers to use the tools they are used to using. That is, it should not be necessary to use any tool apart from HTML (plus JavaScript, and perhaps Flash). This is the reason why we find interesting the *attribute template languages* like Zope Page Templates (ZPT) [34], Tapestry [2] or the one we present in this paper, JST2. In them the template directives are inserted in the host HTML document in the form of tag attributes. This feature carries with it one of the more useful aids that a designer may expect: the *previewability*, that is, the ability to see how the page will look like without using neither the template engine, nor tools different from the HTML editor or browser.

The fact is that, despite the efforts, the coupling between presentation and business logic, or *view* and *model* using MVC terminology [7], still remains. Very few works have been devoted to analyzing the sense and causes of this coupling in order to attack them at their root. A good introductory reference can be [10], in which several

template systems are analyzed. There is only one paper that formalizes the model and view separation [20]. In our opinion, the author forgets to consider one factor, JavaScript, that makes his work hardly applicable. This leads us to provide another way of facing the model-view separation problem by means of the use of a double-model, one for the view and another for the application. This can be considered the main contribution of this paper.

Finally, in the last two years a new breaking idea has emerged in the web development world, leveraging the development of Rich Internet Applications: AJAX [12]. Using already existing technologies in conjunction, AJAX has persuaded web designers to begin thinking of enriching their designs with features more typical of desktop application, even though this decision implies facing the difficulties derived from the use of the cryptic JavaScript DOM API or the need of learning how to use AJAX toolkits such as DoJo [8] or GWT [13], and so on. AJAX drove us to think of the possibility of processing templates in the user's browser. As we will show in this paper this strategy is the key to implement the double-model approach we propose in our renewed template system: JST2. The impossibility of strictly isolating the view and the model, the pernicious demands for the graphic designer required by the existing template systems, and the irruption of AJAX, with its corresponding processing power in the client side, are all forces that concur on JST2 as double-model based template system, decoupled, processed in the client, previewable and familiar to the designer.

Once has been presented the motivation and background, the remainder of the article is organized as follows. Section 2 deals with the related work. Section 3 presents the double-model approach. Section 4 is devoted to JST2. And to conclude, section 5 collects several conclusions and future work lines.

2 Related Work

There is plenty of documentation about template systems in the Internet. As for more academic research papers, the first reference we find is [19], where *TML* is presented, a language based on XML tags added to an HTML document. It relies on *TRiX*, a framework for TML templates processing, being its extensibility its main advantage. *Mixer* is presented in [32], a simple template system for servlet based applications that superficially analyzes the separation problem. Another reference is [1], where *PageGen*, an ASP based template system, is described. In this case the tight coupling of *PageGen* to its Data/Control database may prove restrictive. In [14] the convenience of using "standard templates", i.e. templates in which no non-HTML tags are allowed, is discussed. *STRUDEL* [9], is a framework for intensive data based Web application that includes its own template language, which can be criticized because it allows the direct inclusion of java code in the templates. Other interesting papers are [6,5], in which the *<bigwig>* language for Web applications is presented. It includes the *DynDoc* sublanguage to specify HTML template documents, and the *DynDocDag* data structure [5], that allows the representation of the template document in a way that allows the reconstruction of the document in the client

browser using a JavaScript processor. *DynDocDag* was designed to improve the caching of DynDoc documents, but it does not pay attention to the separation concern.

Related work about model-view separation. Nevertheless, when we look for bibliography related to the analysis of the coupling between the model and the view of a web application, only one work stands out. Parr [20] provides a definition for *strict separation* between model and view in template engines, as well as a definition of template and a classification of the different types a template can belong to. From a more practical point of view, he formulates a set of five rules that a template engine must follow in order to enforce *strict separation* and gives an *entanglement index* that can be used when evaluating template engines.

We can briefly summarize the Parr contribution enumerating his rules: (1) “the view can not modify the model”, (2) “the view can not perform computations upon dependent data values”, (3) “the view cannot compare dependent data values”, (4) “the view cannot make data type assumptions”, and (5) “data from the model must not contain display or layout information”. The entanglement index “is the number of separation rules that a template may violate”, being the minimum 1, since there is no way to prevent data of the model from containing display or layout information.

Parr warns of template engines that encourage separation and lays the responsibility of preserving it on the hands of well-intentioned developers. The essence of Parr’s work is to give the rules to enforce, rather than encourage, the separation, forbidding, e.g., that views can perform computations of any type upon model data values. E.g., a view must not either compute book sale prices as “\$price*0.9” (rule 2), or compare a price with a certain limit like in “\$price<25” (rule 3), or index an array of names with the value of another variable, that is supposedly an integer, like in “\$names[\$id]” (rule 4).

Despite Parr’s rules being well stated, we think that several of them can not be imposed upon most template system that produce HTML as output. Parr does not take into account the possibility of using JavaScript in the HTML documents. It’s impossible to prevent template expressions to be inserted in the middle of a JavaScript code, like in:

```
<html><body>The final book price is
<script>document.write($precio$*0.9)</script> &euro;
</body></html>
```

The previous code is a StringTemplate [21] sample that, once processed and taking 30 as the price variable value, results in the following HTML output:

```
<html><body>The final book price is
<script>document.write(30*0.9)</script> &euro;
</body></html>
```

This fact may cause the violation of rules 2 and 3, and it makes the minimum value for Parr’s entanglement index be 3. This leads us to state that it is impossible to enforce strict model-view separation while using HTML template systems. In the next section we show another approach to the analysis of the separation problem which, if applied, makes the entanglement index be at most three

3 The Double-Model Approach

The essence of model-view separation is that it pursues the protection of both sides of the application from changes in the other. If separation is not provided, changes in one part draw the other to be modified. If separation is provided each part can live isolated, even be developed independently.

Once we have stated that strict separation as Parr defined is impossible to achieve in HTML templates, we analyze another solution to obtain a weaker, but effective, degree of isolation in HTML templates.

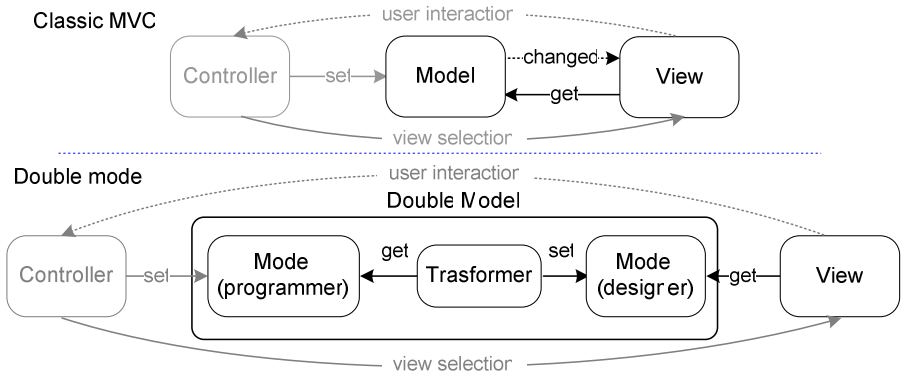


Fig. 1. MVC versus double-model approach

The foundation of the double-model is that each part of the application has a different model. On one hand, the view uses its private model that holds the necessary data to support the functionality of the page. The view is therefore completely matched up to its model. In fact the view can only use its model. No direct reference to the application logic is allowed, not even read-only accesses. HTML designers make up this model as a set of attributes (name-value pairs) that can be simple or compound, and populate it with *test values*. These values can be used during development in order to see the final appearance of the page. This helps achieving the previously mentioned previewability. Compound values may be structured according to data structures that clarify the design. We call this model, *designer's model*. It is developed during the graphical design process based on the functional requirements of the view. Once all these requirements are fulfilled, the model can be considered stable and it not need to change. E.g.: if the page shows a set of books of a certain shopping basket, the model will define data structures to hold the data of the books of the basket, as well as the user's identification and, perhaps, the current date.

It is important to say that in order to change the appearance of the view the designer's model does not need to change, unless a new functional requirement appears that demands new attributes in the model. It does not need to, but it can be changed. The subtle difference is that the decision whether to change it or not is only

taken by the one who created it: the graphic designer.. It is not a change caused by another change in the application, as it happens in other template systems.

Views are developed apart from the other model, which we call *programmer's model*, because it is developed by programmers. This is the classical application model, i.e., the M in the MVC acronym, usually implemented as a set of classes persisted on a data base, or even without classes, i.e. only a data base schema. The programmer's model can freely evolve as needed, surely subject to continuous refactoring during development. But this evolution does not affect the designer's model, which remains isolated in the view.

Obviously both models must be integrated so that the views can show actual information in their pages at execution time. This task is performed by the programmer, who is the one who knows the actual application model and how to extract actual data from it. This integration consists of supplying the view with the actual data from the programmer's model *re-structured* as the designer's model mandates. It is important to say that the view must be passive with respect to this data load, that is, the actual data must be pushed into the template, instead of the template pulling the data from the model. This requirement in favour of the *push strategy* was already argued as necessary by Parr in [20], and we adopt it as necessary also.

At this point of the presentation, the versed reader could think that almost any template system that follows the push strategy, implements the double-model approach. Consider for example Fremarker [11] (with its DataModel), WebMacro or Velocity [31,3], Smarty [25], StringTemplates [21] or HTML::Template [27]. All of them have templates with attributes that are processed pushing values taken from the application model.

Push strategy is not enough. So, what is the key aspect that makes us say that we are in front of a double-model or not? This key is in an above paragraph: "... supplying the view with the actual data from the programmer's model *re-structured* as the designer's model mandates". This *transformation* between models is mandatory.

All the previously referenced template systems do not follow the double-model approach because they allow the designer's model to adopt data structures from the programmer's model. That is, they allow the programmer to suggest to the designer what the most appropriate designer's model is so that the models integration does not require transformation. This allows the programmer to push objects or data structures directly from its model into the templates, which are irremediably entangled with the application. Obviously these template systems encourage developers to use completely different models, but, the fact is that the back door is open, and that under deadline pressure, developers tend to save work using these kinds of resorts. Of course, data structures in both models may coincide (a book has a title and an author anywhere), but this coincidence cannot be used to simplify the transformation, which is always compulsory. And this must be taken not as a drawback, but as a model-view coupling vaccine.

Note that if the programmer changes its programmer's model, he is also responsible for changing the code that makes the models adaptation, but in no case the programmer's model modification is propagated to the view.

As a consequence of double-model approach, templates developed following this can be used in different applications, with the sole requirement that they belong to the same domain. This is a good test that allows us to state that double-model favours the separation between model and view. In fact, isolation is so high that the view can be displayed without a model (which results in a prototype).

3.1 Application Architecture for the Double-Model Approach (MVC+mT)

The transformation of data taken from the programmer’s model into the format specified by the designer’s model is performed in a new architectural component of the application, which is very close to the controller (the C in MVC). We call this component *transformer*.

The presence of the transformer modifies the typical control flow in MVC (or its adaptation to web applications, Model-2 [24]). We will refer to the architecture with the MVC+mT acronym. The control flow is represented in Fig. 2. The Transformer (T) is activated (3 in Fig.2) once the Controller has updated the Model (1, 2). Then the transformer loads (4) the suitable template (View), takes (5) the necessary data from the programmer’s Model (M), and performs (6) the models transformation (the designer’s model is the ‘m’). Then the transformer pushes (7) the resulting data into the template, which is sent to the client (8).

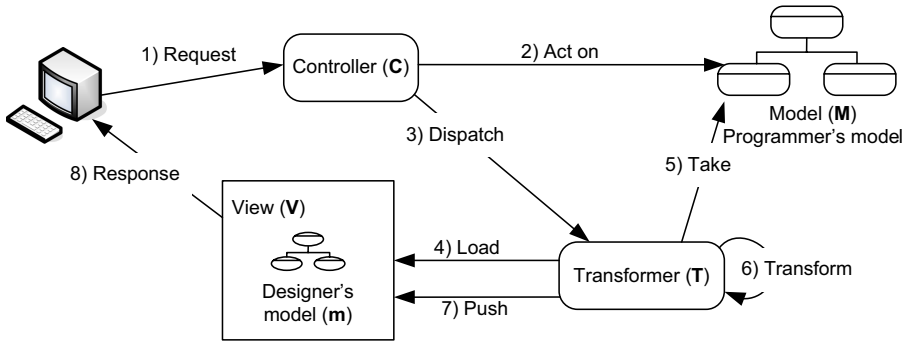


Fig. 2. Application flow in MVC+mT architecture

3.2 Double-Model Requirements

As a summary, we enumerate the conditions we require for a certain template system to be considered double-model compliant:

1. Both parts of the application, model and view, have their own model supporting their requirements, probably expressed using different languages, even paradigms.
2. The view is not allowed to access the programmer’s model in any way.
3. The designer’s model is initialized with test values.
4. The application uses the push strategy [20] to feed data into the template.
5. Data taken from the programmer’s model can not be pushed directly into the template. They must be transformed in the way the designer’s model mandates.

3.3 Double-Model and Entanglement Index

It is possible to relate Parr's work with the double-model approach. In fact we can say that *a template system that follows the double-model approach has a maximum entanglement index of 3*. This property is equivalent to saying that a template system that follows the double-model approach observes Parr's rules 1 and 4. Compliance of rule 1 is directly deduced from requirement 2 of section 3.2. Compliance of rule 4 is supported by requirements 1 and 5, because they guarantee that both parts of the application (classical model and view) use different data types that must be adapted.

4 JST2

JST2 (JST version 2) is an attribute based, script-driven, double-model based HTML template system that is processed in the user's browser. JST2 is an evolution of JST, previously presented at [16,17].

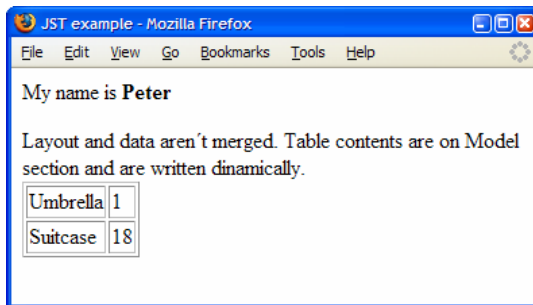
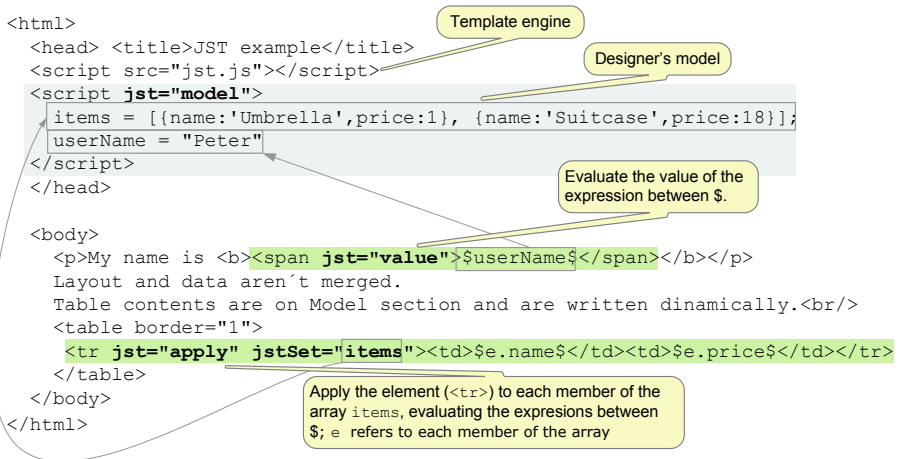


Fig. 3. JST2 example and its visualization on an internet browser

An important feature of JST2 is the use of JavaScript, which is at the foundation of JST (in such a way that JST stands for JavaScript Templates). JavaScript is used to define the designer's model, to populate it with test values, and to fill in the "holes" that the designer places in the HTML page as placeholders for dynamic data. Actually JST2 has no associated language, like other HTML templates have. JST2 defines a set of non-standard HTML-attributes that can be inserted in the HTML elements of the template. These HTML-attributes are processed by the JavaScript engine to perform the page rendering¹. This is relevant for designers, who do not need to learn any other template language. They get by with their familiar tools: HTML and JavaScript, as well as the fact that they can work disconnected from the applications server in order to see their designs working. In fact, JST2 is valid for the development of standalone HTML documents. Fig. 3 shows a JST2 example, and its visualization in the Firefox browser (we stress the fact that the page does not need a server in order to be viewed, just double-clicking on the template document; this is full-previewability).

4.1 Working with JST2

From the point of view of designers, working with JST2 involves creating the HTML page by focusing only on the navigation and look and feel. Designers must identify the dynamic part of the page (in the example in Fig. 3 a user's name and a table of items). Then, they must define the variables within an `script` section that must be marked with the value `model` for the HTML-attribute `jst`. Variables are given a name and assigned a test value to them. Variables can be simple (of any JavaScript type), like `userName` in the example, or compound, like `items` in the example (an array of objects). These variables make up the so-called *JST model section*, which can be easily identified as the designer's model in the double-model approach.

A key point is that designer is free to structure his model as he wants, or knows. This is important for us because, up to certain extent, we do not want to force the designer to learn any particular syntax or to apply a unique way of doing things. E.g.: in the example, object literal notation is used to populate the `items` array, but there exist other possibilities, such as to use constructors of a JavaScript `Item` class, or to use two arrays of simple data (one for names and another for prices).

JST2 expressions. Wherever designers want to insert a data value, they place the variable name between a couple of `$` symbols (e.g., `$userName$`). A variable reference can appear alone or as part of a JavaScript expression, like in `$price*0.9$`. Therefore, designers have all the JavaScript power at their disposal,

¹ A JST2 template is processed by a script loaded with the page. For the processing of the template, the DOM JavaScript API is used. As was expected, many problems have arisen during the template engine development due to different browser peculiarities. A lot of tests have been performed, including form controls, structural elements (tables, divs ...), styles, event handlers, and so on. A total of 112 different versions of browsers have been tested, resulting that JST2 can be used in Internet Explorer (from version 5.5), Firefox, Mozilla, SeaMonkey, Netscape (from version 6), Opera (from version 7), DeepNet, Safari (for Mac platforms) and Konqueror (for Linux). This covers approximately 98% of the available browsers (depending on the consulted browser statistics).

without the cost of learning a new programming language. JST2 expressions may appear as part of the value of HTML elements or attributes.

JST2 non-standard HTML-attributes. The way JST2 expressions are processed is determined by JST2 HTML-attributes. These attributes contain instructions for the template engine. Using attributes we get most editing tools not to complain about they do not understand these attributes, and they will not remove them. Besides, they will not change the structure or appearance of the template when loaded into a WYSIWYG editor or a web browser.

The most important JST2 HTML-attribute is `jst` indeed. Its value specifies the type of processing that the element that carries it must undergo. We call HTML elements marked with this `jst` attribute *JST2 elements*. There are six values for `jst`, which cover evaluations, conditionals, iterations and sub-template inclusions:

- `value`: the JST2 expressions inserted in the JST2 element, or any of its children, must be evaluated and their values must replace the expressions.
- `if`: the JST2 element will be processed only if the value of the conditional expression contained in an auxiliary HTML-attribute named `jstTest` is evaluated to `true`. If the condition is evaluated to `false`, the JST2 element is removed from the final document. All JST2 expressions are also evaluated as in `value`.
- `apply`: the JST2 element must be replicated for each one of the values contained in the array specified in an auxiliary HTML-attribute named `jstSet`. All JST2 expressions are also evaluated as in `value`. There are three implicit variables associated to the underlying iteration process that can be used in the expressions: `i`, the iteration index, `e` the iteration value, and `values`, the iterated set.
- `compApply`: like `apply`, but for a group of JST2 related elements. It also uses the auxiliary HTML-attributes `jstSet` and `jstTest`. For each set value, all the JST2 elements in the `compApply` group are evaluated. Only those whose `jstTest` expression evaluates to `true` are included in the final result.

Several simple examples². Our objective is not to write here a detailed JST2 tutorial. Nevertheless, we want to illustrate the previously presented JST2 syntax with a few simple examples. Let's consider as the designer's model, the following script block:

```
<script jst="model">
  items = [{name:'Umbrella',price:1},{name:
           'Suitcase',price:48},{name:'Belt',price:18}];
  userName = "Peter"
  temperature = -2;
</script>
```

And below the samples:

```
<button jst="value" onClick="alert('Hello,
$username$')">Say hello to $username$</button>
<table border="1">
  <tr jst="apply" jstSet="items">
    <td>$e.name$</td><td>$e.price$ \$</td>
  </tr>
```

² JST2 tutorial and samples can be found at <http://www.unirioja.es/cu/fgarcia/jst/>

```

</table>
<p jst="if" jstTest="temperature<10">
  $userName$, you should wear your coat.
</p>
<table border="1">
  <tr jst="compApply" jstSet="items" jstTest="i%2==0"
    bgcolor="#CCCCCC">
    <td>$e.name$</td><td>$e.price$ \$</td>
  </tr>
  <tr jst="compApply" jstSet="items" jstTest="i%2!=0">
    <td>$e.name$</td><td>$e.price$ \$</td>
  </tr>
</table>

```

Once processed the resulting document will contain the following HTML:

```

<button onClick="alert('Hello, Peter')">Say hello to
Peter</button>
<table border="1">
  <tr><td>Umbrella</td><td>1 $</td></tr>
  <tr><td>Suitcase</td><td>48 $</td></tr>
  <tr><td>Belt</td><td>18 $</td></tr>
</table>
<p>Peter, you should wear your coat.</p>
<table border="1">
  <tr bgcolor="#CCCCCC"><td>Umbrella</td><td>1
  $</td></tr>
  <tr><td>Suitcase</td><td>48 $</td></tr>
  <tr bgcolor="#CCCCCC"><td>Belt</td><td>18 $</td></tr>
</table>

```

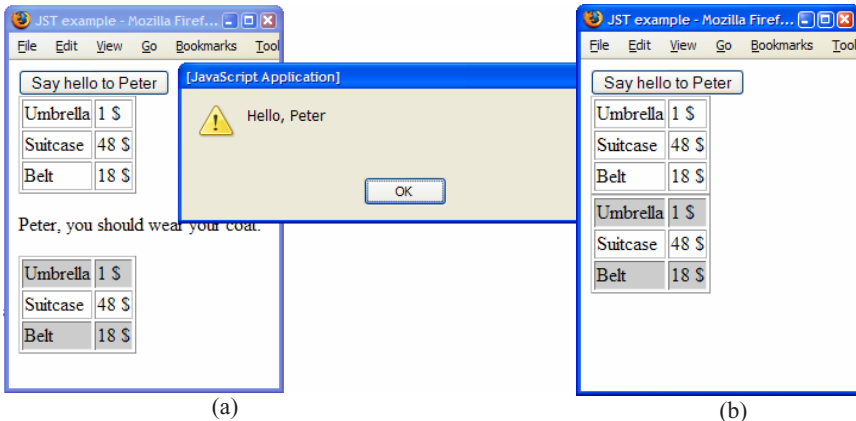


Fig. 4. JST2 examples, a) with temperature value -2; b) with temperature value 25

JST2 in the server-side. Once the work of designers is finished, programmers can proceed. The model section in the JST2 template shows the dynamic data that the page needs to be displayed, thus reducing the communication need between designers and programmers. The programmer must implement the code executed when a request for a

page is received. This code will fetch the final data from an object model, XML, or DBMS (the programmer's model) and will generate a string with JavaScript code that will replace the template model section. The JST2 preprocessor is extremely lightweight; just string substitution. Other template engines processing are not comparable in terms of speed and used memory. E.g.: let's assume that the actual name is "Mary" and there are three items: a leather belt, a handbag and an umbrella; then the page sent to the user will be just as before in Fig. 3, except for the model section:

```
<script>
  name = "Mary"
  items=[{name:'Leather belt',price:1}, {name:
    'Handbag',price:79},{name:'Umbrella',price:18}];
</script>
```

4.2 JST2 Implements the Double-Model

JST2 follows the double-model approach because it meets all of its requirements: (1) The designer's model is constituted by the model section, and the programmer's model can be supported by a DBMS, and XML data source, or any type of data structure in any programming language. (2) The view can not access the programmer's model. The view can only access its model section including the variable names inside JST2 expressions. This is stressed by the fact that the template can even be processed disconnected from the server where the application resides. (3) The model section is initialized with test values. (4) The template is passive from the point of view of its processing. At execution time, the previous test values are replaced with new data values. In the server side of the application, the part that acts as the transformer in MVC+mT architecture, collects all the necessary data and composes a string defining the new JavaScript model section (push strategy). And (5) during the previous processing, data taken from the programmer's model can not be pushed directly into the template, because they belong to different realms, they are even modelled in a different language. They must be transformed in the way the model section specifies.

4.3 JST2 Analysis

The use of JST2 effectively separates the model and view of a web application, and therefore further enhances the extensibility, maintainability, and reusability of the page templates. Besides of these positive properties, profusely treated in the related bibliography, we can mention another set of advantages.

Simplicity. JST2 relies on JavaScript, but it is difficult to appreciate it (apart from the model section and, perhaps, some expressions). This is a great help for designers, who do not need a lot of knowledge of JavaScript in order to use JST2. Besides, the syntax of JST2 is very simple and the learning curve is very soft.

Previewability. Thanks to the use of HTML-attributes in the JST2 elements, HTML editors allow designer to manipulate JST2 elements visually. Designers can work with JST2 elements as if they were normal HTML. Besides, due to the template engine being included as a script in the template page, the final appearance of the template can be observed without needing a server, with the only burden being the

visualization of test data instead of actual data. But this is not relevant for the aesthetics of the template.

Rapid prototyping. As a consequence of the previous feature, JST2 can be used for rapid prototyping of web applications, reducing deployment time. In fact, JST2 can even be used to rapidly develop stand-alone HTML.

Independence of the language used in the server-side of the application. JST2 processing consists only in string substitution, and this task can be performed with any programming language.

Exploitation of the client processing power. JST2 templates are processed in the client browser, thus relieving the server of this task.

JST2 helps to avoid cross-site-scripting (XSS). The usual technique to avoid XSS consists of escaping the potentially dangerous characters (such as `<`, `>`, `&`, ...) by replacing them with their corresponding entity (`<`, `>`, `&`, ...) . Using JST2, text is inserted in the JST2 elements as DOM TextNodes, which automatically escapes those types of characters.

JST2 simplifies AJAX interactions. AJAX processing is very simple with JST2. In fact, for an AJAX interaction to occur, we only have to record the original JST2 element that is going to be asynchronously changed, ask the server for the new data values required for the change, and then re-process the JST2 element with the new values. AJAX data interchanges in JST2 do not require the use of XML, not even JSON [18]; new data is received in the same format as was specified in the model section. That is, in the server side, AJAX requests, can be processed by the same modules that service normal requests.

JST2 problems. But we would not be honest if we did not recognize JST2 problems. One of the main JST2 drawbacks is precisely one of its foundations. JST2 relies on JavaScript and therefore it can not be used in browsers that have it disabled (at the time of writing, 90% of browsers has the use of JavaScript enabled, according to the last browser statistics). This is a limitation for the use of JST2 in devices such as mobile telephones or PDAs, which have available simpler browser versions without JavaScript support. However, it is a matter of time and technological maturity for this not to be a problem any more. Another potential problem is related to web accessibility. We have not investigated yet how accessibility tools process a JST2 template, but we can guess that problems may arise. And finally, a JST2 template is not an XHTML [28] document. Therefore JST2 pages can not show off the W3C stamp that marks it as XHTML compliant. This can be solved using the XHTML modularization mechanism described in [29].

5 Conclusions

When developing a web application, it is not enough to use a template system in order to achieve separation between model and view. The template system must have a set

of features that prevent developers from using programming shortcuts and backdoors that reduce their work load in first term, but entangle the view and the model.

Taking as a starting point the work in [20], we have stated that strict separation of model and view is desirable but impossible to achieve in HTML template systems due to JavaScript. Existing template systems can not observe all the rules stated in [20], but this does not mean we be resigned to release entangled web applications.

We have provided another less strict but effective approach for the analysis of the model-view separation problem, consisting of using two models, one for the view and another for the application. We have also proposed a modification of the MVC architecture that supports the double-model approach (MVC+mT). The double-model approach has been applied in JST2, a template system based on JavaScript. JST2 allows developing templates that are processed in the client browser.

As for future research lines we propose to relate our work to others such us [33], where a similar adaptation process occurs between the front-end and the back-end of a two-tier MVC based architecture that supports the MODFM methodology for the development of web applications; or [4] that presents the *dual-mvc* approach, in which the controller of a MVC application is split between the client and the server. The approach reduces client/server interactions in web applications by returning to the client more data than strictly necessary for page visualization, but that can be used in response to certain user actions. The idea is that the page in the browser has its own controller that fetches local model data instead of having to access the server. This approach has nothing to do with template systems, but shares the idea of having a double model for the application. The existence of these works leads us to consider that perhaps we are in front of an architectural pattern, and that it may appear in more fields apart from web applications. This will be our main line of future work, as well as to keep working in the study of the accessibility in relation to JST2 and the extension to make JST2 XHTML compliant.

Acknowledgments. Partially supported by Comunidad Autónoma de La Rioja, project ANGI-2005/19.

References

1. Al-Darwish, N.: PageGen: An Effective Scheme for Dynamic Generation of Web Pages. *Information and Software Technology* 45(10), 651–662 (2003)
2. Apache: Tapestry, <http://tapestry.apache.org/>
3. Apache: Velocity, <http://velocity.apache.org/>
4. Betz, K., Leff, A., Rayfield, J.T.: Developing Highly-Responsive User Interfaces with DHTML and Servlets. In: *Proceedings of the 19th IEEE International Performance, Computing, and Communications Conference – IPCCC-2000*, pp. 437–443 (2000)
5. Brabrand, C., Møller, A., Olesen, S., Schwartzbach, M.I.: Language-Based Caching of Dynamically Generated HTML. *World Wide Web: Internet and Web Information Systems* 5, 305–323 (2002)
6. Brabrand, C., Møller, A., Schwartzbach, M.I.: The bigwig Project. *ACM Transactions on Internet Technology* 2(2), 79–114 (2002)
7. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture*. In: *A System of Patterns*, vol. 1, John Wiley & Son, West Sussex, England (1996)

8. Dojo, <http://dojotoolkit.org/>
9. Fernández, M., Florescu, D., Levy, A., Suci, D.: Declarative Specification of Web Sites with STRUDEL. *VLDB Journal* 9(1), 38–55 (2000)
10. Ford, N.: *Art of Java Web Development*. Manning Publications Co. (2004)
11. FreeMarker, <http://freemarker.sourceforge.net/>
12. Garrett, J.J.: Ajax: A New Approach to Web Applications (February 18, 2005) <http://adaptivepath.com/publications/essays/archives/000385.php>
13. Google: Google Web Toolkit, <http://code.google.com/webtoolkit/>
14. Halasz, S.J.: An Improved Method for Creating Dynamic Web Forms Using APL. In: *Proceedings of the international conference on APL-Berlin-2000 conference*, Berlin, Germany, pp. 104–111 (2000)
15. Hunter, J.: The Problems with JSP (2000), <http://www.servlets.com/soapbox/problems-jsp.html>
16. Izquierdo, R., García, F.J., Andrés, M., Juan, A., Manrubia, P.: JST: Towards a Usable Web Site Development Method. In: *Proceedings of the IADIS International Conference WWW/Internet 2003*, vol. 1, pp. 515–522. IADIS Press (2003)
17. Izquierdo, R., Juan, A., López, B., Devis, R., Cueva, J.M., Acebal, C.F.: Experiences in Web Site Development with Multidisciplinary Teams. In: Lovelle, J.M.C., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) *ICWE 2003*. LNCS, vol. 2722, pp. 459–462. Springer, Heidelberg (2003)
18. JSON: <http://www.json.org/>
19. Kristensen, A.: Template Resolution in XML/HTML. *Computer Networks and ISDN Systems* 30, 139–249 (1998)
20. Parr, T.J.: Enforcing Strict Model-View Separation in Template Engines. In: *Proceedings of the 13th International Conference on World Wide Web, WWW 2004*, May 17–20, pp. 224–233. ACM Press, New York (2004)
21. Parr, T.J.: StringTemplates. <http://www.stringtemplate.org/>
22. Parr, T.J.: Web Application Internationalization and Localization in Action. In: *Proceedings of the 6th International Conference on Web Engineering, ICWE 2006*, ACM Press, New York (2006)
23. PHP: <http://www.php.net>
24. Seshadri, G.: Understanding JavaServer Pages Model 2 architecture. Exploring the MVC design pattern. *JavaWorld.com* (December 1999) <http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html>
25. Smarty. <http://smarty.php.net/>
26. Sun Microsystems: Java Server Pages™ Specification, Version 2.1. Sun Microsystems, Palo Alto, USA (2006), <http://java.sun.com/products/jsp/>
27. Tregar, S.: HTML:Template, <http://html-template.sourceforge.net>
28. W3C: XHTML™ 1.0 The Extensible HyperText Markup Language (2nd Edition), <http://www.w3.org/TR/xhtml1/>
29. W3C: XHTML™ Modularization 1.1. <http://www.w3.org/TR/xhtml-modularization/>
30. W3C: XSL Transformations (XSLT) Version 2.0 (2007), <http://www.w3.org/TR/xslt20/>
31. WebMacro, <http://www.webmacro.org/>
32. Wijkman, P., Dissanaik, S., Wijkman, M.: Mixer, Supporting the Model-View-Controller Design Pattern in Servlets. In: *Proceedings of the IASTED International Conference on Software Engineering SE 2004*, Innsbruck, Austria, February 16–18, pp. 658–661 (2004)
33. Zhang, J., Chung, J-Y.: Mockup-Driven Fast-Prototyping Methodology for Web Application Development. *Software-Practice and Experience* 33(13), 1251–1272 (2003)
34. Zope: Zope Page Templates, ZPT, <http://zpt.sourceforge.net/>

Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces

Damiano Distante¹, Paola Pedone², Gustavo Rossi³, and Gerardo Canfora⁴

^{1,4} Research Centre on Software Technology (RCOST), University of Sannio, Italy
{canfora,distante}@unisannio.it

² Faculty of Engineering, University of Salento, Italy
paola.pedone@unile.it

³ LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
gustavo@lifia.info.unlp.edu.ar

Abstract. This paper presents a model-driven approach to the development of web applications based on the Ubiquitous Web Application (UWA) design framework, the Model-View-Controller (MVC) architectural pattern and the JavaServer Faces technology. The approach combines a complete and robust methodology for the user-centered conceptual design of web applications with the MVC metaphor, which improves separation of business logic and data presentation. The proposed approach, by carrying the advantages of Model-Driven Development (MDD) and user-centered design, produces Web applications which are of high quality from the user's point of view and easier to maintain and evolve.

Keywords: Web Engineering, Model-Driven Development, Model Transformation, Model Driven Architecture, UWA, UML, MVC, JavaServer Faces.

1 Introduction

Web applications design methodologies available in the literature can be classified into those which focus on “what” the application is required to do (conceptual design in the problem domain) and those which focus on “how” the application can satisfy its requirements and implement the “what” (logical design in the domain of solutions). UWA [38], OOHD [36], OOWS [32] and OO-H [9] pertain to the first category; Conallen’s proposal [13] belongs to the second one; UWE [19][22] and WebML [10][11][12] can be considered hybrid methodologies, as they cover both conceptual and logical design.

Conceptual design methodologies abstract from implementation details and offer an overall view of the system from the view point of the users. Conceptual modeling is the right starting point to implement complex systems. However, the big distance between the conceptual model of a web application and its implementation makes the use of conceptual design methodologies insufficient for the development of a web application. If intermediate design levels for translating conceptual specification into implementation design are not provided, then the implementation activities of a web application may proceed independently from conceptual design, which will thus be under-exploited or even a waste of effort.

The trend of well-known design methodologies to evolve towards Model-Driven Development (MDD) shows the need for integrated approaches that support the whole web application lifecycle. Methodologies that have recently moved in this direction include OO-H [3], OOWS [3], UWE [18][20] and OOHD [35], discussed more in detail in the related work section.

This paper presents a model-driven approach to developing web applications based on the Ubiquitous Web Application (UWA) conceptual design methodology and the Model-View-Controller architectural design pattern. Compared to others, the UWA design framework, with its methodology and models, is particularly suited for designing web applications which are intended to be ubiquitous (accessible by different user types in different usage contexts and with different goals) and user-centered. By combining the characteristics of UWA with the advantages of the MVC architecture and the MDD paradigm, the resulting approach is particularly suited for developing ubiquitous web applications in a user-centered perspective as well as for supporting their maintenance and evolution.

UWA models are made at the right abstraction level to be used with application stakeholders and, at same time, can be detailed enough (design in the small) to provide useful information to the application developers. The methodology is also supported by a number of tools for drawing up UWA models and generating application design documentation. Nevertheless, since UWA models are independent from implementation details, they do not specify aspects such as the architecture, the software components, nor the database that the developer should implement to realize the designed application. By creating an intermediate design model to represent the application being implemented and a set of heuristics to map the UWA conceptual models onto it, we lay the basis for a model-driven approach to developing UWA-based web applications.

The intermediate design model (referred to as the logical model in the following) is based on standard UML diagrams and adopts the Model-View-Controller (MVC) architectural design pattern [8].

The main contributions of the paper are:

1. The definition of a logical model that describes the application code to be developed and maps it to the application requirements;
2. The definition of a set of heuristics to translate UWA conceptual models to the new logical model;
3. The definition of an MDD approach based on UWA to support the entire web application lifecycle.

The remainder of the paper is organized as follows. The next section discusses the background and motivations of the work. Section 3 presents the proposed UWA-based MDD approach and an example of its application. Related work is presented in Section 4 while Section 5 concludes the paper and provides an overview of future work.

2 Background and Motivations

2.1 A General Framework for the Development of Web Applications

Despite the differences between the various methodologies for engineering web applications found in the literature, a set of common concerns and development steps

can be identified [37]. Fig. 1 graphically represents these common features. Basically any design methodology models web applications at three construction levels (content, navigation and presentation) and addresses a number of aspects spanning from structure to behavior. Aspects are orthogonal to all construction levels. Structure applies to content (how contents are structured) as well as to navigation (hypertext structures) and presentation (e.g., page organization). Similarly, behavior (e.g., business rules and user operations) can be associated with content, navigation and presentation.

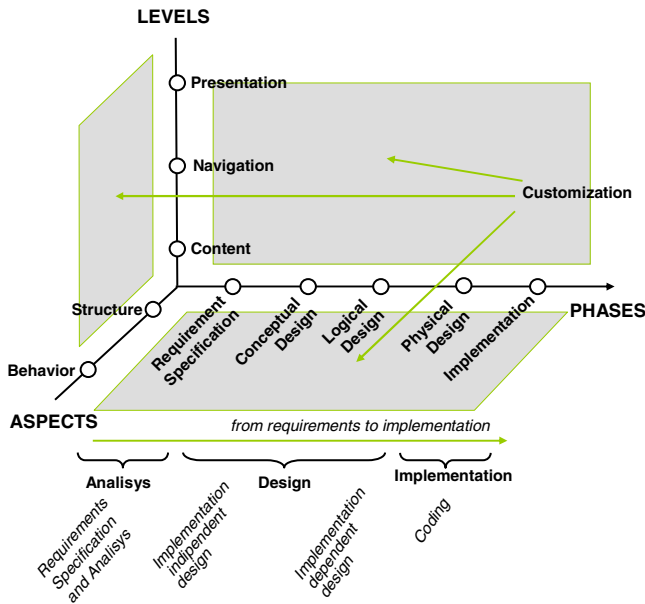


Fig. 1. A general development framework for web applications

As represented in Fig. 1, the development process starts with requirement specification and analysis and proceeds through a number of design steps to the implementation phase. Design steps include:

1. Conceptual design, focused on describing the problem domain and what the system is expected to do, independently of any technological detail.
2. Logical design, focused on the operation of the system while hiding implementation details specific to a particular platform.
3. Physical design, which adapts the logical model of the application to obtain detailed specifications for implementation with the chosen platform.

2.2 Model-Driven Development

Though the use of design methodologies is not yet a common practice in the field of web engineering, it is known that a model-based approach provides a better alternative to the ad-hoc development of web applications and its inherent problems

[37]. The need for systematic approaches to be adopted when developing complex systems and for design prior to implementation is now widely accepted.

Model-Driven Development (MDD) [5] sees software development as a process whereby a high-level abstract model is successively translated into increasingly more detailed models, in such a way that eventually one of the models can be directly executed by some platform [3]. The MDD approach not only advocates the use of models (such as those resulting from the design steps described in the previous section) for the development of software, but also emphasizes the need for transformations in all phases of development, from system specification to implementation and testing [23]. Transformations from one model to the next create a chain which enables the automated implementation of a system starting from requirements.

The possibly most well-known initiative of MDD is the Model Driven Architecture (MDA) proposal [25] defined by the Object Management Group (OMG). The central idea of MDA is to separate platform-independent design from platform-specific implementation of applications, delaying the dependence on specific technologies for as long as possible. Therefore, MDA advocates the construction of platform-independent models (PIM) and the support of model transformations. The model that is directly executed by a platform (PSM) which satisfies all requirements, including non-functional ones, is also called “code”, and is usually the last model in the refinement chain.

The development of web applications is a specific domain in which MDD can be successfully applied, due to the web-specific separation of concerns: content, navigation, presentation and customization.

2.3 UWA

The Ubiquitous Web Applications design framework (UWA) [38] provides a methodology and a set of models and tools for user-centered conceptual design of ubiquitous web applications. Mapping UWA models onto the general web application development framework depicted in Fig. 1, produces the diagram shown in Fig. 2. UWA covers the phases of requirement specification and analysis and conceptual design. Requirement elicitation is the activity devoted by UWA for the identification of the application’s stakeholders, their goals and, through a refinement process, the resulting requirements for the following design of the application. Requirements are classified into content, structure of content, access to content, presentation, and behavior (system and user operations).

Following the requirement elicitation phase, the conceptual user-centered design of the application is developed. The UWA Information Model, Navigation Model and Publishing Model cover the three levels of design represented on the vertical axis in Fig. 1, respectively. Each of these models comprises the structure aspect (e.g., the Information Model gives information on the application content, their structure, the structures for accessing content, etc.). The behavior aspect is addressed by two models: the Transaction Model and the Operation Model, the former addressing the business process design and the latter the design of more elementary user functionalities.

Finally, the UWA Customization Design activity, defines the customization rules that may apply to any of the UWA design models, thus providing the designed application with the ability to adapt to different usage contexts.

The UWA methodology enables separation of design concerns by devoting a design activity and a resulting model to each of the levels and aspects which characterize data- and operation-intensive web applications [34][14]. User-centered design ability (the perspective in developing the different UWA models is that of the different user types of the application) and context awareness for the resulting applications makes UWA one of the most valuable conceptual design methodologies for web applications. The availability of a MOF-Compliant metamodel for UWA [4] makes it possible to specify the semantics associated with each modeling concept, the valid configurations and the constraints that apply. At same time, being the UWA metamodel MOF-compliant it is possible to create modeling tools that generate models that can be easily exchanged, imported in different design tools, rendered into different formats, transformed, and used to generate application code [24].

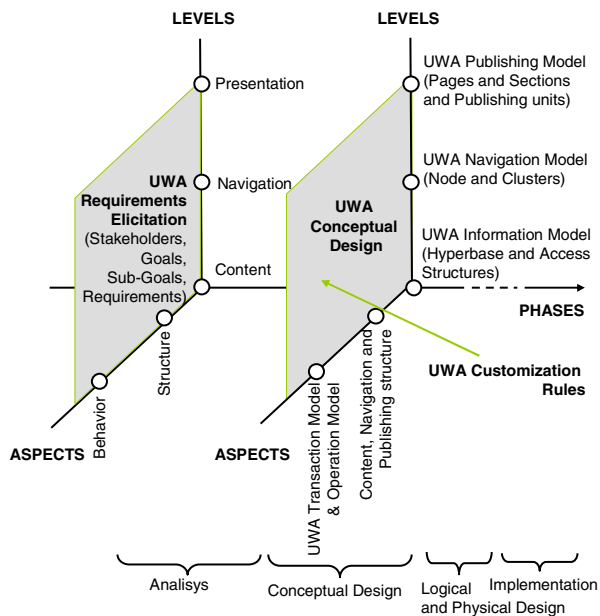


Fig. 2. UWA Requirement Elicitation and Conceptual Modeling overview

3 A UWA-Based MDD Approach

3.1 Process Overview

The proposed UWA-based MDD approach to the development of web applications uses a series of models and successive model transformations to progressively refine

and enrich the application requirements so as to obtain the application source code. The different models are also the basis for updated project documentation, at different levels of abstraction.

Fig. 3 depicts the overall development process and the generated models. The process begins with UWA Requirements Elicitation, to identify the stakeholders of the application and their goals, from which the application requirements are derived. Stakeholders, goals, sub-goals and requirements are represented by means of stereotyped UML use-case diagrams.

The conceptual design is carried out using the UWA methodology and the conceptual model of the application produced. This model includes the Information Model, the Transaction Model, the Navigation Model, the Operation Model, the Publishing Model and Customization Model.

The conceptual design phase is followed by logical design. The UWA conceptual models are transformed, by means of suitable translation rules, into a logical model that is closer to the specifics of implementation but still platform-independent. The adoption of standard UML diagrams and of the MVC architecture for the resulting application gives the name “UML-MVC” to our proposed logical model.

The final phase of the MDD process consists of specializing the UML-MVC logical model to the specific platform chosen for implementation, obtaining a

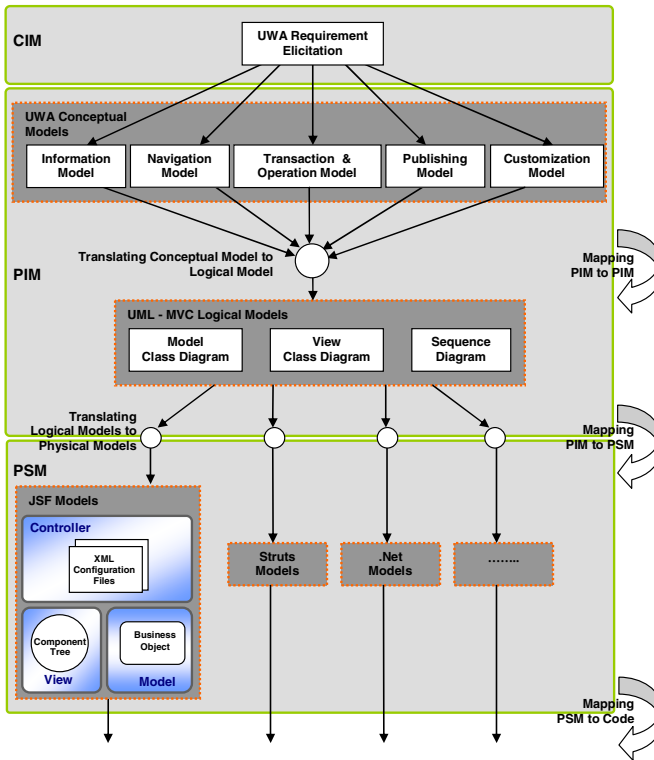


Fig. 3. An overview of the UWA-based MDD process

platform-specific model. Any implementation of the MVC architectural design pattern is suitable as the destination platform for the UML-MVC logical model defined in the previous step. In particular, we detail the case of the JavaServer Faces (JSF) technology [15][17] which is the technology we chose for experimenting our approach with an e-commerce web application.

As emphasized by Fig. 3, the proposed process, in addition to be model-driven, adheres to MDA. The UWA Requirement Elicitation model can, indeed, be considered a CIM as it focuses on functionalities required for the application. The UWA conceptual model and the UML-MVC logical model are PIMs, as they do not imply any specific technology to be used for the implementation. Finally the JSF model is a PSM as it is the specialization of the UML-MVC PIM for the JavaServer Faces technology.

In the following of this section we describe the UML-MVC logical model, the transformation rules to obtain this model from the UWA conceptual models, and the guidelines that can be used to map the UML-MVC model onto the JavaServer Faces implementation framework.

3.2 The UML-MVC Logical Model

The logical model we elaborated consists of stereotyped standard UML diagrams and is intended to model the software components to be developed for each of the three layers of the MVC architecture [8] as well as the relationships among them. Specifically, the model is structured into the following UML diagrams:

1. Model Class Diagram (MCD): an UML class diagram modeling the classes that participate in the Model layer and implement the application business logic and data persistency.
2. View Class Diagram (VCD): an UML class diagram representing the client and server pages in the View layer. These pages have the responsibility of presenting data and content to the user and enabling user interaction with the system. This diagram also models:
 - a. Classes making part of the Controller layer;
 - b. Associations between user interactions (e.g., the “Submit” of a form of a View page) and methods of classes of the Model that are in charge to serve them;
 - c. Associations between attributes of classes of the View (which correspond to data provided/requested by the application) to methods of classes of the Model that manage them.
 - d. Navigation links between pages of the View.
3. One or more UML Sequence Diagram describing the interactions between the various components of the system and their state transitions during the execution of complex user activities and web transactions. In addition, this diagram is used to describe the navigation steps between pages of the View that are associated with the different return values of the execution of the methods of Model classes.

The resulting overall model is independent of the specific technologies chosen for the implementation of the application, thus it is a PIM in the MDA architecture, but it is already sufficiently detailed to guide the application's development team. The

model can be used to create the application with any implementation technology based on the MVC pattern. On an experimental basis, we used the Java ServerFaces framework [15].

3.3 Mapping UWA Conceptual Models onto UML-MVC Logical Model

The transformation of UWA conceptual models into UML-MVC logical models is accomplished by means of a set of mapping heuristics which create appropriate correspondences between the UWA models and modeling concepts, with the components of the MVC architecture and the elements of the logical model. Table 1 summarizes these heuristics: the first column lists the UWA conceptual models; the second lists the main modeling concepts included in each of the UWA models; the third column reports the layers of the MVC architecture onto which the considered UWA modeling concept is mapped; finally, the fourth column reports the specific elements of the logical model that originate from the UWA concept. Elements of the UML-MVC logical model we defined include: (i) Classes, Class Attributes and Class Methods of the MCD; (ii) Client and Server Page Classes of the VCD; (iii) Controller Classes mapping the user interactions with pages of the VCD onto methods of classes on the MCD; (iv) Associations between Classes of the MCD, (v) Associations

Table 1. Summary of UWA to UML-MVC transformation rules

UWA Conceptual Model	UWA modeling Concept	MVC Component	UML-MVC Modeling Element
Information Model	Entity Type	Model	Class into MCD
	Entity Component	Model	Class into MCD aggregated to the Class created for the Entity Type
	Slot	Model	Private Attribute and associated set and get methods into MCD
	Semantic Association	Model	Association between Classes into MCD
	Association Center	Model	Association Class into MCD
	Collection Type	Model	- Class into MCD associated to the Collection Center - Method added to the Class involved in the collection
Transaction Model	Complex Activity	Model	Class into MCD with a method originated from the Activity PropertySet
	Suspendable Transaction	Model	Class into MCD with attributes <i>TransactionHistory</i> , <i>CurrentActivity</i> , <i>ExecutionState</i> , <i>Suspension</i> and <i>DeviceType</i>
	Suspendable Complex Activity	Model	Class into MCD with attributes <i>TransactionHistory</i> , <i>CurrentActivity</i> , <i>ExecutionState</i> , <i>Suspension</i> and <i>DeviceType</i>
	Elementary Activity	Model	Method added to the involved Class of the MCD to implement the activity
	Execution Flow	Controller	Sequence Diagram representing the interaction between elements of the logical model and the navigation rules associated to the different return value from the execution of methods of Classes of the MCD
Navigation Model	Navigation Node	View	Server Page Class or Client Page Class into VCD
	Navigation Cluster	View	Navigation Links between Pages into VCD
	Activity Node	View	- Server Page Class into VCD with input/output attributes - Associations between input/output attributes of the Server Page Class into VCD with methods of MCD classes managing the data - Association between action elements of the Server Page Class (e.g. Submit button of a Form) and Controller classes to serve them
		Controller	- Controller Class into VCD with an attribute Result. - Association between attribute Result and method of the MCD class to be invoked in correspondence of the user interaction with the page
	Activity Cluster	View	- Navigation Links between Pages into VCD
Publishing Model	Publishing Unit	View	Client Page or Server Page Class into VCD
	Section	View	Client Page or Server Page Class into VCD aggregating classes originated from Publishing Units
	Page	View	Client Page or Server Page Class into VCD aggregating classes originated from Publishing Sections
	Page Template	View	Frameset Class into VCD
	Link	View	Navigation Links between Pages of the VCD

between attributes of Classes of the VCD with methods of MCD classes; (vi) Navigation Links between Pages of the VCD; (vii) Sequence diagrams representing the workflow of a transaction and navigation rules to be implemented by the Controller.

Broadly speaking, the UWA Information and Transaction Models merge into the MCD, while the UWA Navigation and Publishing Models merge into the VCD. Associations between attributes and user interaction elements of the View pages with methods of Model classes originate from the Navigation Model and, indirectly, from the Transaction Model. In addition, from the UWA Transaction Model the UML Sequence Diagrams are also created.

3.4 Mapping UML-MVC Logical Model onto JavaServer Faces Platform Specific Model

The JSF technology is a Java implementation of the MVC architectural design pattern which simplifies the building of user interfaces for web applications by assembling reusable UI components in a page, connecting these components to an application data source; and wiring client-generated events to server-side event handlers [17].

As synthesized by bottom part of Fig. 3, the JSF architecture is a specialization of the MVC architecture in which the Model component is realized by means of Java business objects, the View component is made up of JavaServer Pages (JSP) in which custom tag libraries are used for expressing the JSF user interface components, and the Controller is implemented by a Servlet named FacesServlet.

To map the UML-MVC logical models onto the JSF software components the following guidelines can be used:

1. Classes of the MCD are mapped onto Java Business Objects, such as JavaBeans (JB) or Enterprise JavaBeans (EJB).
2. Classes of the VCD are implemented by means of Java Server Pages including the JSF user interface components specified by means of the JSF tags.
3. Associations between pages of the VCD with methods of classes in the MCD are used to define associations between presentation JSF components included in the Java Server Pages and attributes and methods of the Java Business Objects.
4. Associations between Controller classes and classes of the MCD in the VCD, together with information derived from the Sequence Diagrams on navigation rules associated with the different return values of the execution of a method of a Model class are used to define the “faces-config.xml” configuration file for the FacesServlet Controller.

3.5 An Example Application

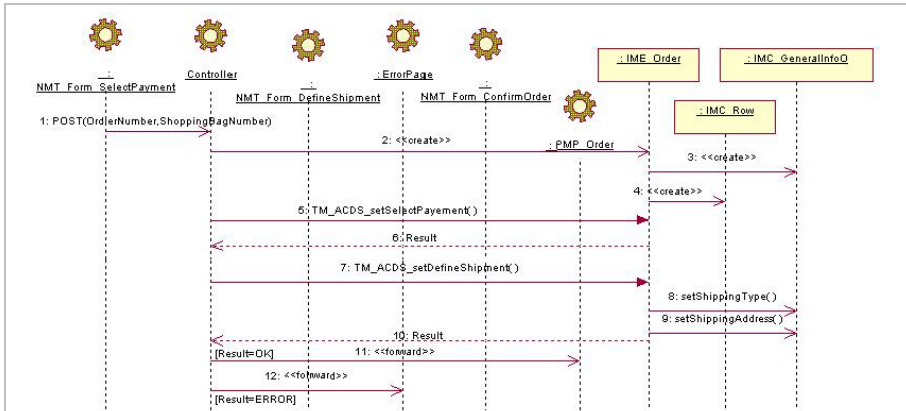
The proposed UWA-based MDD approach was applied to the development of an e-commerce website, of which we report here the portion concerning the order checkout and visualization. The process involved the analysis of the requirements and the conceptual design of the application by means of the UWA methodology. The transformations rules summarized in Table 1 were then used to obtain the logical UML-MVC model of the application, which was finally specialized for

Table 2. An excerpt of the conceptual and logical models generated by the UWA-based MDD process for developing an e-commerce web application

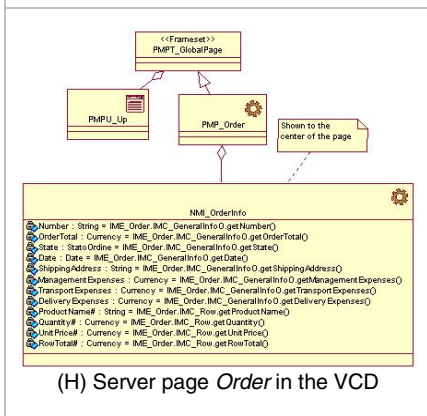
<p>(A) Order UWA Entity Type Diagram</p>	<p>(B) Order UWA Navigation Cluster</p>
<p>(C) Order Checkout UWA Organization Model</p>	<p>(D) Order Checkout UWA Execution Model</p>
<p>(E) Order UWA Publishing Page</p>	<p>(F) Class Order in the MCD</p>

implementation with the Java ServerFaces technology. Table 2 shows an excerpt of the different models that were generated during the process and a screenshot of the final application. More in detail, the table reports the Entity Type diagram (A) and the

Table 2. (Continued)



(G) UML-MVC Sequence Diagram for the checkout transaction



(H) Server page Order in the VCD



(I) A screenshot of the Order View web page

Navigation Cluster (B) associated to an *Order* made by a customer. The first diagram models the information characterizing the *Order* and is part of the UWA Information Model. The latter models the navigation associated with the *Order* and the actions that can be invoked by the user, such as *Order Checkout*. The Organization Model (C) and the Execution Model (D) describe the activities in which the *Order Checkout* transaction is organized and represent a portion of the UWA Transaction Model of the application. Diagram (E) of the table shows the UWA Publishing Model for the *Order Page*. By applying the transformation rules, the Entity Type *Order* of the UWA conceptual model was transformed into the class *Order* in the MCD (F) of the logical model, while the Navigation Cluster and the Publishing Page *Order* originated the server page *Order* in the VCD (H). The UML sequence diagram (G) was derived from the *Order Checkout* UWA Execution Model.

Attributes of the *Order* class were derived from Slots of the *Order* Entity Components in the conceptual model. Get and Set methods were automatically associated to each of the attributes. The set of activities included in the Organization

Model of the *OrderCheckout* transaction were transformed into methods for the class *Order*. These methods include: *setSelectPayment*, *setDefineShipment*, *BuyLater* and *ConfirmOrder*. The class diagram modeling the server page *Order* in the View Class Diagram (H) also reports the associations between attributes of the page (i.e., JSF user interface components in the implementation) and methods of the class *Order* in the Model Class Diagram. To support traceability, class names and methods of the UML-MVC logical model are prefixed with the acronyms of the UWA model and modeling concept from which they originate.

The structure of the Navigation and Publishing Models impacted onto the VCD, in which they appear also the dependency with the respective classes of the MCD apt to the management of the information. A screenshot of the web page resulting from the implementation of the prototype of the application with the JSF technology is shown by figure (I).

3.6 Costs/Benefits of the Approach

The introduction of an additional design phase in the development process of a web application cause the process to lengthen and complicate, and more effort to be required if supporting tools towards MDD are not provided.

On the other hand, no matter of if tools for the automatic transformation of conceptual models into logical models are available, providing developers with models which are closer to the implementation simplifies implementation choices, reduces coding time and helps in producing higher quality software. In fact, having a model which from one hand is directly linked to the conceptual model and from the other is very close to the implementation details helps the development of applications which exhibit:

- greater internal consistency, as they fully satisfy the requirements of conceptual design;
- greater usability, as they are more able to satisfy the expectations of the users;
- greater maintainability, as design and requirements, since the impact on the code of any changes to the model (or indeed the impact on the model of any changes to the code) can be traced.

The proposed approach, by defining the UML-MVC logical model, makes it possible to establish a correspondence between the UWA conceptual design of a web application and its implementation. Maintenance and evolution operations become easier, requirements traceability is made possible as well as alignment between software and documentation during the entire application lifecycle.

4 Related Work

The Model-Driven Development paradigm is applied successfully by a number of web engineering methods, such as UWE, OO-H, OOHMDMA, and WebML. These methods use models to separate the platform-independent design of web systems from the platform-dependent implementations as much as possible. They have associated development environments that support code generation from model specifications, either fully or partially automated.

The UWE [23] process to developing web systems follows the MDA principles and uses the OMG standards [24-29]. The process makes use of model transformations defined at metamodel level and specified in general purpose transformation languages, such as QVT [26] and graph transformations. Currently, many of the transformations have already been automated, thanks to the OpenUWE [31] tool suite. One of the main characteristics of this suite is its open architecture based on established standards. These standards are supported by both open-source and commercial tools. The common data exchange language within this architecture is based on the extensible UWE meta-model [37].

OO-H [9] supports the transformation-based construction of a presentation model based on modelling elements of the navigation model, and code generation based on the conceptual, navigation and presentation models [18]. OO-H transformation rules are a proprietary part of a CASE tool called VisualWADE [39]. This tool supports modelling and automatic generation of applications based on XML, ASP, JSP, and PHP.

OOHDM [36] may be considered as a platform-independent domain-specific language for web applications that provides an object model, in contrast to other web application modeling languages. OOHMDA [35] generates servlet-based web applications from OOHDM models. The OOHMDA approach follows MDA principles by employing the OOHDM conceptual and navigational scheme of a web application as the basic PIM for the MDA process, using any UML-based design tool, such as Rational Rose [33], which produces an XMI-file as output. The basic PIM is transformed into the intermediate PIM, by adding to it the behavioural semantics of the OOHDM core features and business processes. This transformation is achieved by modifying the XMI-file of the basic PIM. The PIM is then transformed into a servlet-based PSM.

WebML [10-12] is a model-driven method for the development of data intensive web applications, with an associated supporting CASE tool called WebRatio [40]. WebML follows an MDD approach for mapping its modelling elements onto the components of the MVC Model 2 architecture, which can be transformed into components for different platforms [11]. The web application generated by WebRatio is deployed in a runtime framework based on a set of Java components, which can be configured by use of XML files. The runtime architecture is based on the MVC design pattern and is suited for the Apache Struts open-source web application development framework [1] and the JSP tag libraries [30].

Similar to our approach, UWE, OO-H and OOHMDA adopt an MDD process that follows MDA principles for the models. Differently from our approach they do not explicitly adopt the MVC architecture pattern in their PIMs. As above shortly reported, WebML differs from our and other considered approach in that its process is MDD but not MDA. Similar to our approach, WebML uses MVC as architectural pattern for its PIMs. All the considered approaches enable different technologies to be used for the implementation of the PSMs. Our choice of adopting MVC as architecture for the PIM logical model guarantees the availability of a wide range of open-source and commercial technology frameworks to choose from for the different platforms, such as J2EE, .Net and PHP.

A final as obvious as notable difference between our approach and others proposed in the literature lies in the conceptual design methodology on which the approach is

based. Our UWA-based MDD approach enables the development of web applications which faithfully implement their UWA conceptual model, thus benefiting from the UWA peculiarities and the quality design characteristics as summarized in Section 2.3. This difference is the main motivation for the present work.

5 Conclusion

Conceptual design methodologies enable the analyst to abstract from implementation and technological details and focus on application requirements in the problem domain. This is referred to as “what” the application has to do.

The UWA design methodology and associated models are particularly suited for the user-centered design of complex ubiquitous web applications. UWA provides different related models to design the different levels and aspects characterizing a web application (content, navigation, presentation, structure, behavior).

Despite the benefits deriving from the use of conceptual design methodologies, there is a wide gap between the models they generate and the implementation of the application. A further design step and related models are required to specify “how” to implement the final application adhering to the conceptual models.

This paper proposed an intermediate model, named UML-MVC logical model, to be used between UWA conceptual design phase and the implementation phase, and a set of transforming heuristics.

The UML-MVC logical model is a platform independent model, based on standard UML diagrams and incorporating the MVC architecture design pattern. The set of heuristics permits the automatic transformation of the UWA models into the logical ones. The model and the mapping heuristics are the basis of a Model-Driven Development approach based on the UWA methodology, for web applications. In addition, the rules to transform the logical model to a platform specific model for Java ServerFaces applications has been defined and experimented in an example case study.

Overall, the resulting approach combines the advantages of MDD, such as requirement traceability and maintenance and evolution better support, with the ability of UWA to design application in a user-centered perspective and ubiquitous. The logical model creates a link between the application implementation and the UWA design models and application requirements.

We are currently working on the specification of our model transformations rules using the QVT language and on the development of appropriate tools to support the whole UWA-based MDD approach by extending the design tools provided with the UWA methodology.

References

1. Apache STRUTS open-source framework: <http://struts.apache.org/>
2. ArgoUML: <http://www.argouml.org>
3. Arraes Nunes, D., Schwabe, D.: Rapid Prototyping of Web Applications combining Domain Specific Languages and Model Driven Design. In: ICWE'06. Proceedings of the 6th International Conference on Web Engineering July 11-14, 2006, Palo Alto, California, USA (2006)

4. Baresi, L., Garzotto, F., Maritati, M.: W2000 as a MOF Metamodel. In: Proceedings of World Multiconferemce On Systemics, vol. 1 (2002)
5. Bézivin, J.: In Search of a Basic Principle for Model Driven Engineering. UPGRADE V(2), Novótica (April 2004)
6. Brambilla, M.: Extending hypertext conceptual models with process-oriented primitives. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 246–262. Springer, Heidelberg (2003)
7. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. ACM Transactions on Software Engineering and Methodology (TOSEM) (in print, 2006)
8. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Patter-Oriented Software Architecture - A system of pattern, vol. 1. John Wiley & Sons Ltd., West Sussex, England (2000)
9. Cachero, C., Gómez, J., Pastor, O.: Object-Oriented Conceptual Modeling of Web Application Interfaces: the OO-H Method Abstract Presentation Model (2000)
10. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing 6(4) (2002)
11. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Application. Morgan Kaufmann Publishers, Elsevier Science (USA) (2003)
12. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. Computer Networks 33(1-6), 137–157 (2000)
13. Conallen, J.: Building Web application with UML, 2nd edn. Addison Wesley, Redwood City, CA, USA (2002)
14. Distante, D., Rossi, G., Canfora, G., Tilley, S.: A Comprehensive Design Model for Integrating Business Processes in Web Applications. International Journal of Web Engineering and Technology (IJWET) 2(1), pp. 43–72 (2007)
15. Dudney, B., Lehr, J., Willis, B., Mattingly, L.: Mastering JavaServer™ Faces. Wiley, New York (2004)
16. Gomez, J., Cachero, C., Pastor, O.: Extending a Conceptual Modeling Approach to Web Application Design. In: Wangler, B., Bergman, L.D. (eds.) CAISE 2000. LNCS, vol. 1789, pp. 5–9. Springer, Heidelberg (2000)
17. JavaServer Faces Technology: <http://java.sun.com/javaee/javaserverfaces/>
18. Koch, N.: Transformation Techniques in the Model-Driven Development Process of UWE. In: MDWE 06. Proceedings of the 2nd Model-Driven Web Engineering Workshop, Palo Alto, CA, July 11, 2006, ACM Press, New York (2006)
19. Koch, N., Kraus, A.: The expressive Power of UML based Web Engineering. In: Proceedings of the 2nd International Workshop on Web Oriented Software Technology (IWWOST02) June 10, 2002, Málaga, Spain (2002)
20. Koch, N., Zhang, G., Escalona, M., j.: Model Transformations from Requirements to Web System Design. In: ICWE'06, Palo Alto, California, USA, July 11-14, ACM Press, New York (2006)
21. Langham, M., Ziegeler, C.: Cocoon: Building XML Applications, Sams Publishing (2002)
22. Meliá, S., Gómez, J., Koch, N.: Improving Web Design Methods with Architecture Modeling. In: 6th International Conference on E-Commerce and Web Technologies (EC-Web 2005) August 22-26, 2005, Copenhagen, Denmark (2005)
23. Moreno, N., Fraternali, P., Vallecillo, A.: A UML 2.0 Profile for WebML Modeling. In: MDWE 06. Proceedings of the 2nd Model-Driven Web Engineering Workshop, Palo Alto, CA, July 11, 2006, ACM Press, New York (2006)

24. Object Management Group (OMG) Meta Object Facility Specification (MOF): <http://www.omg.org/mof/>
25. Object Management Group (OMG). Model Driven Architecture (MDA): www.omg.org/mda/
26. Object Management Group (OMG). Query/Views/Transformations (QVT): www.omg.org/
27. Object Management Group (OMG). UML 2 Object Constraint Language (OCL): www.omg.org/docs/ptc/03-10-14.pdf
28. Object Management Group (OMG). Unified Modeling Language (UML): Superstructure, version 2.0 www.uml.org/
29. Object Management Group (OMG): XML Metadata Interchange (XMI) www.omg.org/
30. Open Source JSP Tag Library: <http://www.java-source.net/open-source/jsp-tag-libraries>
31. OpenUWE: <http://www.pst.ifi.lmu.de/projekte/uwe>
32. Pastor, O., Abrahão, S., Fons, J.: An Object-Oriented Approach to Automate Web Applications Development. In: Bauknecht, K., Madria, S.K., Pernul, G. (eds.) EC-Web 2001. LNCS, vol. 2115, pp. 16–28. Springer, Heidelberg (2001)
33. ROSE: IBM Rational Software. Online at www.ibm.com/rational
34. Rossi, G., Gordillo, S., Distante, D.: Improving Web Applications Evolution by Separating Design Concerns. In: STEP 2005. IEEE Software Technology and Engineering Practice 2005, September 24-25, 2005, Budapest, Hungary, Workshop on Evolution of Software Systems in a Business Context (2005)
35. Schmid, H.A., Donnerhak, O.: OOHMDA - An MDA Approach for OOHDM. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 569–574. Springer, Heidelberg (2005)
36. Schwabe, D., Rossi, G.: An Object-Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems (TAPOS) 4, 207–225 (1998)
37. Schwinger, W., Koch, N.: Modeling Web Applications, in Web Engineering - Systematic Development of Web-Applications. In: Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds.) John Wiley & Sons Ltd., West Sussex, England (2006)
38. UWA Consortium, Ubiquitous Web Applications. In: Proceedings of the eBusiness and eWork Conference 2002, (e2002: October 16-18 2002, Prague, Czech Republic) (2002)
39. VisualWADE. <http://www.visualwade.com>
40. WebRatio. <http://www.webratio.com>

On Refining XML Artifacts

Felipe I. Anfurrutia, Oscar Díaz, and Salvador Trujillo

University of the Basque Country - San Sebastián (Spain)
{felipe.anfurrutia,oscar.diaz,struji}@ehu.es

Abstract. Step-wise refinement is a powerful paradigm for developing a complex program from a simple program by adding features incrementally where each feature is an increment in program functionality. Existing works focus on object-oriented representations such as Java or C++ artifacts. For this paradigm to be brought to the Web, refinement should be realised for XML representations. This paper elaborates on the notion of XML refinement by addressing what and how XML can be refined. These ideas are realised in the XAK language. A Struts application serves to illustrate the approach.

1 Introduction

So far, most Web applications are conceived in a one-to-one basis. A recent study indicates a cloning rate (i.e. code repetition throughout the application) of 17-63% within Web applications of the same organisation [5]. This cloning evidences the existence of a common, although implicit, theme throughout the applications, and confirms an intuition felt in most organisations: code similarities among applications. These similarities are being handled in various ways such as IFDEFs, configuration files, installation scripts or cloned software copies *à la* “copy-paste-modify”. However, these solutions do not scale and can hinder maintenance as the number of variations increases.

One technique to handle similarities is step-wise refinement [1]. Step-wise refinement is a powerful paradigm for developing a complex program from a simple program by incrementally adding details. This approach attempts to depart from current “clone&own” practise by leveraging reuse of the common parts, and separating variable and changing parts as *program deltas*. The final product is obtained through composition: the common parts are composed with the program deltas that realise the variations for the product at hand.

Existing works focus on object-oriented representations such as Java or C++ artifacts [1]. However, recent studies revealed that the cloning rate of web-specific artifacts (e.g. mainly XML files) was considerably higher than general artifacts (e.g. *Java*, *C++*, etc) [5]. Indeed, XML artifacts play a preponderant role in current software practises, specially in the Web setting. This omnipresence of XML vindicates the existence of modularisation techniques specially tuned for XML. Attempts have been made to bring object-orientation (OO) and componentisation to the HTML realm [3], [4], [6]. But in the same way that OO falls short to face the increasing complexity of conventional software (crosscut handling is a case in point), so does it happen for XML artifacts.

This work addresses the use of refinements as a modularisation technique for XML artifacts. But, what is meant to refine an XML artifact? Does it mean that we can arbitrarily insert or delete a node anywhere in an XML tree? To this end, a language

for defining refinements in XML documents is introduced: XAK (pronounced “sack”). The language is accompanied by a *validator* and a *composer*. The validator checks the correctness of XML refinements at compile time, whereas the composer builds incrementally a document by composing a base document with a refinement document. XAK composer is currently available as part of the AHEAD Tool Suite [7].

In this way, an XML artifact can also be conceived incrementally. This accounts for the following advantages: enhanced reusability of XML artifacts since commonalities and variabilities can be defined separately; flexibility in the selection of variable content and their composition; and decoupling validation of XML core artifacts from XML refinements. First, a brief on the notion of refinement is provided.

2 On the Notion of Refinement

A refinement can be thought of as a function that takes an artifact as an input, and returns another similar artifact but leveraged to support a given feature (denoted by *Feature1*•*Base*). In other words, a base artifact (e.g. a Java class) can now be incrementally extended (i.e. refined in our parlance) by adding a new module (e.g. a method) that extends the functionality of the base with a new feature (i.e. *Feature1*). At first sight, this resembles regular inheritance, but there is a difference: *there are not two classes but a single class that is being incrementally extended to account for a new feature*. Furthermore, the class being extended is not fixed at compile time (like in regular inheritance) but decided at composition time. In this way, a refinement behaves like a mixin inheritance, i.e. a class whose super class is parametrised [2]. Since the super class is not fixed until composition, distinct refinements on different (and unpredictable) order may be composed to yield a class. When the artifact is source code, a class refinement can introduce new data members, methods and constructors to a target class, as well as extend or override existing methods and constructors of that class. But, what is meant to refine an XML artifact? The next section introduces a sample case.

3 A Motivating Example Using Struts

Let *CurrencyConverter* be a web application that facilitates information about converting distinct currencies¹. The application exhibits a J2EE architecture where Apache Struts is used. Struts follows the MVC pattern where the Controller separates the control-flow from the Model and the View. Space limitation makes us focus on the controller. However, similar remarks can be made for the artifacts realising the Model and the View.

The control-flow of a sample base application is realised through the *struts-config* document depicted in figure 1(b). The description so far accounts for the *base* controller. The term “base” refers to the stable core which is free for any variations. The issue arises when this base functionality needs to be leveraged with additional capabilities due to either perfective maintenance or versioning. For instance, consider two additional

¹ For a working example see www.oanda.com/convert/classic

<pre> <xs:schema> <xs:element name="struts-config" xak:modularizable="yes"> ... <xs:element name="form-beans" xak:modularizable="yes"> ... <xs:element name="form-bean" xak:modularizable="yes"> ... </xs:element> <xs:element name="global-forwards"> ... <xs:element name="action-mappings" xak:modularizable="yes"> ... <xs:element name="action" xak:modularizable="yes"> ... <xs:element name="plug-in"> ... </xs:schema> </pre> <p style="text-align: center;">(a)</p>	<pre> <struts-config xak:artifact="struts-config.xml" xak:feature="base"> <form-beans xak:module="mForms"> <form-bean xak:module="mConverterForm" ...> <form-property name="amount" initial="1" .../> <form-property name="sourceCurrency" .../> <form-property name="targetCurrency" .../> </form-bean></form-beans> <action-mappings xak:module="mActions"> <action path="/converter" xak:module="mButtons" ...> <forward name="button.convertNow" path="/convertNow.do"/> <forward name="button.cheatsheet" path="/cheatsheet.do"/> </action> <action path="/convertNow"> ... </action> <action path="/cheatsheet"> ... </action> </action-mappings> </struts-config> </pre> <p style="text-align: center;">(b)</p>
--	--

Fig. 1. (a) Schema-based and (b) instance-based modularisation

features: (1) the *DateRate* feature, which allows end users to introduce a date in order to make the currency conversion with the rates at the given date. This implies to refine the *base* controller with new form properties, and some additions to the control-flow; (2) the *Customisation* feature, which permits end users to personalise the application by providing default values for both the *sourceCurrency* and the *targetCurrency* properties. This simple feature impacts all the model, the view and the controller.

Despite their simplicity, we are unaware of any mechanism that permits to incorporate these features *incrementally*. That is, start with a simple product (i.e. the *base*) and compose *deltas* to progressively add features to the base. Notice, you can add *ifdef* tags to the base code that a pre-compiler can leave or remove depending on the features to be finally exhibited by the application. But, this is more a kind of configuration or parametrisation mechanism that requires the designer to foresee all possible extensions where superfluous extensions are removed at configuration time. By contrast, refinements work the other way around: start with a simple product and apply deltas (i.e. program refinements) to progressively elaborate the desired product. Refinements are defined separately from the *base* in both time and space. In time, because the refinement can be added at any time. And in space, since the refinement is handled separately from the *base* artifact. Therefore, product synthesis rests in the ability to implement and compose refinements.

4 The Unit of Refinement

We aim at synthesising XML documents incrementally through refinements. But, what is the granularity of this refinement? A first approach could be to consider *any* XML node as the unit of refinement. However, this implies handling XML documents as mere data structures where any element node can be subject to refinement. This is too fine-grained granularity that defers the principle of modularity whereby high level abstractions (i.e. the modules) encapsulate their low level realisation (i.e. the instructions). Indeed, the

Open-Closed Principle (OCP) states that modules should be both open (for extension and adaptation) and closed (to protect the content against certain modifications).

To this end, a distinction is made between elements playing the role of modules (and hence, being subject to refinement) and elements that describe the realisation of these modules (and hence, protected against punctual updates). Thus, an XML module is defined as an element of a document that carries out a specific function and is liable to be re-used by/combined with other modules.

Besides realising abstractions, modules should be univocally identified. Xpath relies on element location. If the position of the element changes, so does it the Xpath expression output. Therefore, location-based Xpath expressions can not be used for element identification when the position of this element is liable to be changed, as it is the case for refinements. The order of refinements (e.g. *Feature1*•*Feature2*•*Base* vs. *Feature2*•*Feature1*•*Base*) can make the location of a given element to change. Thus, XML modules must have and preserve an ID property which permits to address this module unambiguously.

Similar to code artifacts, the identification of the main abstractions (modules) depends on the domain at hand. In an XML setting, this domain is partially defined by vocabularies. W3C XML Schema is one of the most popular schema languages. A schema states the element, attribute and atomic type names, in addition to structural constraints that instances of this schema must obey.

Next, we need a way to indicate the elements playing the role of modules. For our sample case, we want to stay that only element types `<struts-config>`, `<form-beans>`, `<form-bean>`, `<action-mappings>` and `<action>` can be modules (liable to be refined). The rest of the element types can not be refined (e.g. `<controller>` served only for implementation). This is the purpose of the “*xak:modularizable*” attribute. Figure 1(a) shows the Struts schema now annotated with this attribute. The attribute indicates whether an element type is eligible to be a module or not. For a given document instance, this does not force every occurrence of a modularisable element type to be refined, but prevents non-modularisable element types from being refined.

However, stating modularity at the schema level can be too general. Frequently, the notion of module depends on the document at hand. Hence, “schema-based” modularisation is complemented with “instance-based” modularisation (using the *xak:module* attribute). This approach permits to further restrict what can be refined among the modularisable elements. For our sample case, only `/converter` is a refinable `<action>`; whereas `/convertNow` and `/cheatsheet` can not be refined. Figure 1(b) illustrates this situation for our sample case. This moves the decision of what can be refined to the instance level.

“Schema-based” and “instance-based” approaches to module definition offers a good balance between the controlled approach that offers the schema, and the freedom that programmers’ creativity requires. This is akin to the openness and subsidiary way of working that characterise the XML world (e.g. schema management in XML Schema). *Schema designers* can use a schema approach to define the “refinable” element types, *the schema users* can work at the instance level by indicating the “refinable” elements, and finally, *the instance users* compose the features to synthesise the final application, refining some element contents, should it be required.

```

<xak:refines xak:artifact="struts-config.xml"
  xak:feature="customisation">
  <form-beans xak:module="mForms">
  <xak:keep-content/>
  <form-bean xak:module="mCustomizeForm"
    name="customizeForm" type="DynaActionForm">
    <form-property name="defaultSourceCurrency" .../>
    <form-property name="defaultTargetCurrency" .../>
  </form-bean>
  </form-beans>
  <action xak:module="mButtons">
  <xak:keep-content/>
  <forward name="button.customize" path="/customize.jsp"/>
  </action>
  <action-mapping xak:module="mActions">
  <xak:keep-content/>
  <action path="/customize" name="customizeForm" ...>
  <forward name="success" path="/converter.jsp"/>
  </action>
  </action-mapping>
  </xak:refines>
  (a)

```

```

<struts-config xak:artifact="struts-config.xml"
  xak:feature="customisation_base">
  <form-beans xak:module="mForms">
  <form-bean xak:module="mConverterForm"> ...</form-bean>
  <form-bean xak:module="mCustomizeForm"
    name="customizeForm" type="DynaActionForm">
    <form-property name="defaultSourceCurrency" .../>
    <form-property name="defaultTargetCurrency" .../>
  </form-bean>
  </form-beans>
  <action-mappings xak:module="mActions">
  <action path="/converter" xak:module="mButtons" ...>
  <forward name="button.convertNow" path="/convertNow.do"/>
  <forward name="button.cheatsheet" path="/cheatsheet.do"/>
  <forward name="button.customize" path="/customize.jsp"/>
  </action>
  ...
  <action path="/customize" name="customizeForm" ...>
  <forward name="success" path="/converter.jsp"/>
  </action>
  </action-mappings>
  </struts-config>
  (b)

```

Fig. 2. (a) A XAK refinement for the *Customisation* feature and (b) the resulting document from the composition *customization*•*base*

5 The Ways of Refinement

Product synthesis starts with a base product and apply deltas (i.e. refinements) to progressively incorporate new features to this product. Thus, there are two kinds of artifacts: base documents (i.e. values) and refinement documents (i.e. functions).

Base document. Any traditional XML document can be a *base* document. The only difference is that now a distinction is made between XML elements, liable to be refined (i.e. modules), and those that can not be refined (i.e. the implementation). To this end, the XAK namespace provides three attributes (see figure 1b), namely: `@xak:artifact`, which specifies the name of the document that is being incrementally defined; `@xak:feature`, which indicates the name of the feature being supported; and `@xak:module`, which identifies those elements that play the role of modules. Notice that the designer is not forced to turn into modules all elements of a modularisable type.

Refinement documents. A refinement is an increment in program’s functionality. This is specified through the following XAK elements: `<xak:refines>` and `<xak:keep-content>`. The former is the root element of the refinement document. Its content describes a set of module refinements (i.e. elements annotated with the `xak:module` attribute) over a given base document (i.e. the `xak:artifact` attribute). Moreover, the `<xak:keep-content>` attribute indicates the place where the content of the refined module will be placed once it is synthesised.

As an example, consider our sample case. The *customisation* feature enhances the *base* by providing default values for both the *sourceCurrency* and the *targetCurrency* properties. Adding this feature impacts on all the aspects: the model, the view and the controller. Thus, three refinement documents are needed, all with `@xak:feature = “customisation”`. Let’s focus on the controller, i.e. “*struts-config.xml*”. This artifact is

² For base documents, this attribute keeps the value “base”.

gradually defined as features are being composed. The *base* is shown in figure 1b where *mForms*, *mActions* and *mButtons* are set as modules. *Customisation* implies: (1) adding the *customizeForm* form-bean into the *mForms* module; (2) extending the *mButton* dispatcher action to show the *customise.jsp* page of the feature, and (3) defining a new action. At synthesis time, *customisation* • *base* will deliver the enhanced *struts-config.xml* file shown in figure 2b.

A refinement realises just an increment, i.e. a delta. Hence, it is most unlikely that the refinement obeys the schema of the type of document being refined. For instance, our previous refinement (see figure 2a) is not a valid *struts-config* document since it holds and `<action>` element outside an `<action-mappings>` element. Moreover, the elements and attributes of the XAK namespaces are intermingled with the elements of the given schema vocabulary.

Therefore, the validity of a refinement can not be checked directly against neither the schema of the document being refined (e.g. *struts-config.xsd*) nor the XAK schema. However, the element names, types and some structural constraints still hold. For instance, the elements and attributes used in the refinement should be permitted by the content model of the module being refined. This implies to define which are the laws of refinement and develop a tool for checking it.

6 Conclusions

Step-wise refinements permits to conceive artifacts incrementally, hence, distinguishing between stable, base artifacts and refinement artifacts that realise the variations. This work addresses refinement of XML artifacts. The peculiarities brought by markup languages as opposed to object-oriented ones have been exposed where the notion of refinement offers an alternative way to modularise source code for languages where no other modularisation technique is available.

References

1. Batory, D., Sarvela, J.N., Rauschmayer, A.: Scaling step-wise refinement. *IEEE Transactions on Software Engineering* 30(6), 355–371 (2004)
2. Bracha, G., Cook, W.: Mixin-based inheritance. *SIGPLAN* 25(10), 303–311 (1990)
3. Gellersen, H.-W., Wicke, R., Gaedke, M.: WebComposition: an object-oriented support system for the Web engineering lifecycle. *Computer Networks and ISDN Systems* 29(8-13), 1429–1437 (1997)
4. Klapsing, R., Neumann, G., Conen, W.: Semantics in Web Engineering: Applying the Resource Description Framework. *IEEE Multimedia* 8(2), 62–68 (2001)
5. Rajapakse, D.C., Jarzabek, S.: An investigation of cloning in web applications. In: Lowe, D.G., Gaedke, M. (eds.) *ICWE 2005*. LNCS, vol. 3579, Springer, Heidelberg (2005)
6. Schranz, M.W., Weidl, J., Goschka, K.M., Zechmeister, S.: Engineering complex World Wide Web services with JESSICA and UML. In: Proc. of the 33rd Annual Hawaii Int. Conf. on System Sciences (HICSS'00), Maui, HI, USA (2000)
7. Trujillo, S., Batory, D., Díaz, O.: Feature Refactoring a Multi-Representation Program into a Product Line. In: Proc. of the 5th Int. Conf. on Generative Programming and Component Engineering (GPCE'06) (2006)

Mixup: A Development and Runtime Environment for Integration at the Presentation Layer

Jin Yu¹, Boualem Benatallah¹, Fabio Casati², Florian Daniel³,
Maristella Matera³, and Regis Saint-Paul¹

¹ University of New South Wales, Sydney NSW 2052, Australia
{jyu,boualem,regiss}@cse.unsw.edu.au

² University of Trento, Via Sommarive, 14/I-38050, Trento, Italy
casati@dit.unitn.it

³ Politecnico di Milano, Via Ponzio, 34/5-20133, Milano, Italy
{daniel,matera}@elet.polimi.it

Abstract. In this paper we present a development and runtime environment for creating composite applications by reusing existing *presentation* components. The granularity of components is that of stand-alone modules encapsulating reusable functionalities. The goal is to allow developers to easily create composite applications by combining the components' individual user interfaces.

1 Introduction

User interface (UI) development is one of the most time-consuming tasks in the application development process [3]. As a result, reusing UI components is critical in this process. There is a large body of research in areas such as component-based systems, enterprise application integration and service composition [1], but little work has been done to facilitate integration at the presentation or UI level. While UI development today is facilitated by frameworks (such as Java Swing) providing pre-packaged UI classes such as buttons, menus and the likes, high-level presentation components encapsulating reusable application functionalities have received little attention.

This demo presents a development and runtime environment, called *Mixup*, for integration *at the presentation level*, that is, integration of components by combining their presentation front-ends, rather than their application logic or data. The goal is to be able to quickly build complex user interfaces – and in particular web interfaces – by dragging and dropping existing web UIs (called components) on a canvas and by specifying how components should synchronize based on user and application events. As an example, consider building a US National Park Guide application that includes a park listing, an image displayer showing images given a point of interest and a map displaying the location of a given point of interest. The application can be built out of presentation front-ends such as Google Maps and Flickr.NET¹. Once integrated, the components should present information in an orchestrated fashion, so that for example when a user selects a national park from the park listing, the image displayer shows an image of the selected park, while the map displays its location (Fig. 1).

¹ The .NET version of Flickr.

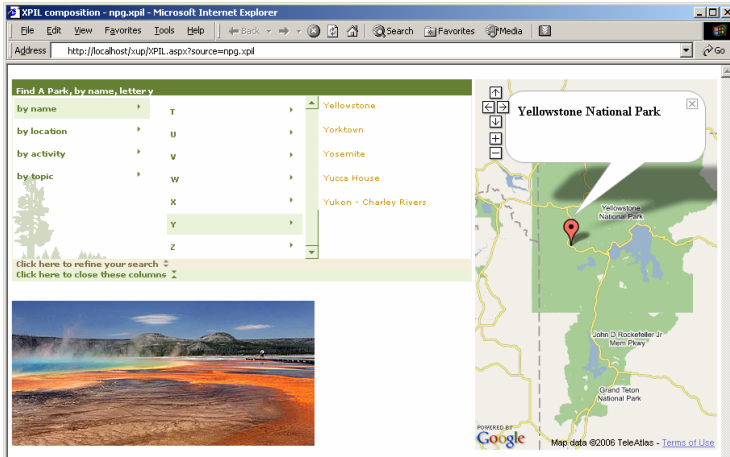


Fig. 1. The National Park Guide

In this demo we show how composite applications can be created by combining the front-ends of existing applications, how to define the orchestration logic among them as well as their layout, and how all these can be done quickly via a visual editor.

Details on the motivations, rationale, challenges and the proposed approach are provided in [4]. A slideshow of the demo scenario is available on the web².

2 The Conceptual Framework

Borrowing lessons from application integration, we argue that integration at the presentation layer needs the definition of four basic elements: component model, composition model, specification language, and runtime environment.

For the **component model**, we argue that presentation components are centered around the notion of *presentation state*, which is a conceptual, application-specific description of what the component is showing. For example, for a map component, the state may describe the location currently being shown. In addition, a component exposes *events* to notify state changes (e.g. due to user interaction with the component’s UI) and *operations* to allow other components to request state changes. Finally, a component has *properties* to represent appearance characteristics (e.g. text color) and customization parameters. Note that the Mixup component model is abstract; that is, it is independent of the technologies used for native component implementations.

The **composition model** allows the specification of event-based integration logics, as we argue that presentation integration is mostly event-based for synchronizing UIs. The integration logics are specified via a set of *event listeners*, each linking an event in one component to an operation in another component. For example, the park listing component fires a park selection changed event when the user selects a different park; this causes the invocation of an operation on the map component to show a map of the newly selected park and the invocation of an operation on the image displayer to show an image of the new park.

² <http://www.cse.unsw.edu.au/~jyu/icwe07/demo.pdf>

To model components and compositions, Mixup includes the **Extensible Presentation Integration Language** (XPIL), which contains a set of XML elements for describing the component model (i.e. *component descriptor*), similarly to WSDL for Web services, and a set of XML elements for specifying the composition model.

Finally, a **runtime middleware** executes the resulting composite application by interpreting the specifications in XPIL. In addition, the middleware includes a component adapter framework that allows bindings from the abstract component model to a concrete (native) component implementation. A *component adapter* thus facilitates the communication between the runtime middleware and the native component implemented in a particular component technology (e.g. ActiveX, Java Applet, etc.).

3 Mixup Demo Flow

In this section we demonstrate how the National Park example can be developed and executed using our framework. To create the composite application, the developer follows these steps: i) creating abstract component descriptors (if not yet available) out of UI front-ends of existing applications and save them in a component registry; ii) creating the composite application by specifying its components, their interactions and their layout information; iii) generating the XPIL documents and deploy the composite application to the runtime environment.

Creating Component Descriptors. The abstract component descriptors for the three components in our National Park example can be created either semi-automatically or manually via the *component editor* (Fig. 2).

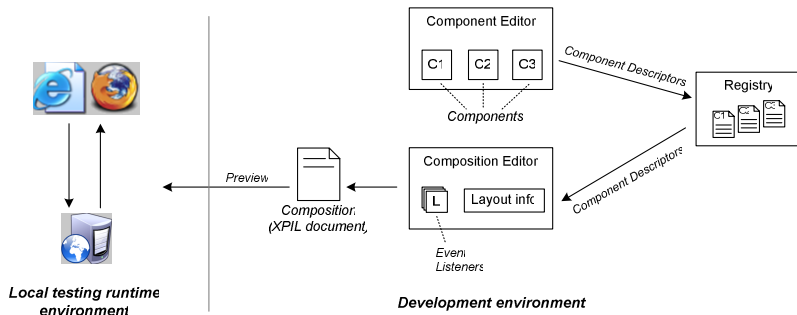


Fig. 2. Development environment

First, if a particular component technology supports abstract descriptions (e.g. WSRP) or meta-language facilities such as reflection (e.g. Java Applet), the component editor may call a suitable *component inspector* to find out the component's native events, operations and properties and then generate the component model descriptor with the appropriate bindings to the native implementation (this is similar to automatically generating a WSDL document from a Java class).

The component developer may take the automatically generated component descriptor and directly deposit it into the registry. However, typically not all events, operations and properties are needed for the integration. The developer may choose to filter them and to keep only the relevant ones, possibly renamed for better readability.

In cases where meta-language facilities are unavailable, the component editor allows the developer to manually create component descriptors. For the Google Maps component (as well as other AJAX components), the developer can create an abstract operation and specify its binding by pointing to the appropriate JavaScript function.

As depicted in Fig. 3, the editing panel (left hand side of the editor) shows the three components in our National Park example. The yellow flash icon denotes events, and the red rhombus icon denotes operations. Note that the label under the operation or event contains the name of the corresponding native method (e.g. JavaScript function, .NET method) in the component implementation.

Creating Composite Application. To build the National Park example, the developer needs to specify both composition logic and layout information via the *composition editor* (Fig. 2).

First, the developer must select the appropriate components from the registry window (upper right corner of Mixup Editor in Fig. 3) and places them in the editing panel (left hand side of the editor). She then defines event listeners by drawing arrows that link events of one component to operations of other components. For example, the arrow from the "ParkSelectionChanged" event of the park listing component to "showPOI" operation of Google Maps implies that once the park selection is changed by the user, the "showPOI" operation should be called to display the map of the newly selected park. Similarly, the arrow to the "search" operation of Flickr specifies that Flickr should display an image of the park newly selected by the user.

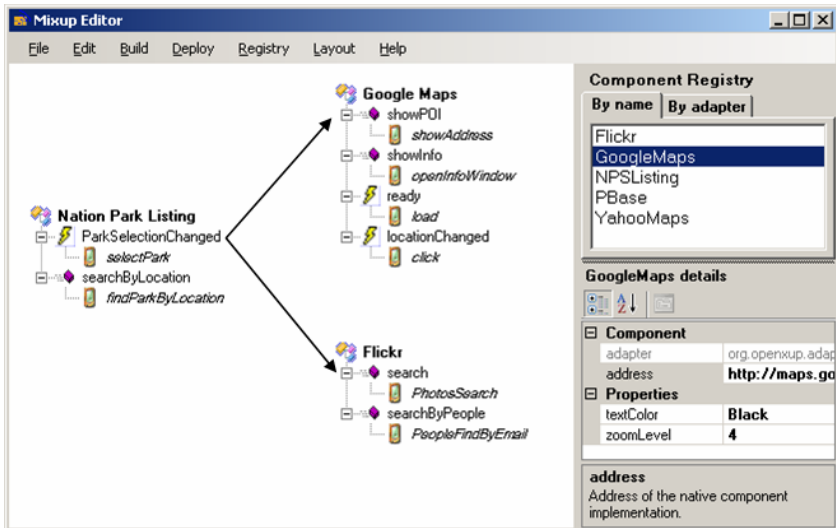


Fig. 3. Mixup Editor

If the event parameters and operation inputs do not match (e.g. number of parameters, data types), the editor will request additional input from the developer, such as XSLT or XQuery statements that can transform the event parameters to operation inputs. In addition, the developer may also specify additional integration and/or transformation logics in the form of scripts or references to external code.

In Fig. 3, the registry window contains several additional components. Specifically, we could use Yahoo Maps instead of Google Maps as the map component in our National Park example. All we need to do is to select Yahoo Maps and drop it into the editing panel and then link the "ParkSelectionChanged" event of the park listing component to the appropriate operation of Yahoo Maps. Similarly, we could also use PBase image service instead of Flickr. Conversely, presentation components can be reused in a variety of composite applications. For example, the Google Maps component defined as per our framework can be also used in real estate applications and wherever maps are needed as part of the UI functionality.

Finally, the editor includes a layout view for specifying layout information. Non-markup based components (e.g. Java applets, ActiveX controls) are represented by boxes corresponding to their location and size; the developer can move and resize the boxes to define the components' layout information. For markup-based components (e.g. AJAX components), the developer can provide additional CSS statements so that the components can have non-rectangular shapes and can be mixed and overlapped.

Deployment and Runtime. Once the component descriptors are created and the composition logic and layout information are fully specified, an XPIL document can be generated and deployed to the runtime middleware. Our development environment includes a testing runtime, which consists of a browser and a web server running in the local development machine (Fig. 2).

Finally, the demo shows the integrated UI at work – when the user interacts with one component, the other related components will change in a synchronized fashion according to the specifications in the composition model. The result of executing the composite application, US National Park Guide, is shown in Fig. 1.

4 Related Work

There are numerous web application frameworks for building composite GUI applications from reusable modules; for example, Java Portlet, ASP.NET Web Parts, and WSRP. These frameworks all require the components to be built using their specific interfaces or APIs. On the other hand, our framework is at a higher level - our component model provides an abstract layer on top of any existing component interfaces; and we do not require or enforce any specific APIs. Furthermore, our component model is generic enough to model existing presentation components developed in these frameworks (as a matter of fact, we plan to provide component adapters for the frameworks mentioned above).

Detailed discussions on related work in this area can be found in [2].

References

1. Alonso, G., et al.: Web Services: concepts, architectures, and applications. Springer, Heidelberg (2004)
2. Daniel, F., et al.: Understanding UI integration: a survey of problems, technologies, and opportunities. *IEEE Internet Computing* (May/June 2007)
3. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: the Proceedings of SIGCHI'92, Monterey, California (May 1992)
4. Yu, J., et al.: A framework for rapid integration of presentation components. In: the Proceedings of WWW'07, Banff, Canada (May 2007)

Squiggle: an Experience in Model-Driven Development of Real-World Semantic Search Engines

Irene Celino, Emanuele Della Valle, Dario Cerizza, and Andrea Turati

CEFRIEL – Politecnico di Milano, Via Fucini 2, 20133 Milano, Italy
{celino,dellavalle,cerizza,turati}@cefriel.it

Abstract. Search engines are becoming such an easy way to find textual resources that we wish to use them also for multimedia content; however, syntactic techniques, even if promising, are not up to the task: future search engines must consider new approaches. In order to prove that Semantic Web technologies provide real benefits to end users in terms of an easier and more effective access to information, we designed and developed **Squiggle**, a Semantic Web framework that eases the deployment of semantic search engines. Following a model-driven approach to application development, **Squiggle** makes ontologies part of the running code. We evaluate the advantages of **Squiggle** against traditional approaches in real world deployments.

1 Introduction

Searching everything everywhere is becoming our habit when we need to find something. We search Web pages in Web search engines, music using search engines integrated in multimedia players, pictures in images organizer applications, even personal stuff using desktop searches. However, *finding* what we need is often a hard job. Current search engine technology is very good in finding complete Web pages published all over the world, but it lacks the desired precision¹ and recall² when searching for multimedia resources. For instance, searching “jaguar” in an image search engine results in a mix of felines and cars, which are difficult to tell apart. Moreover, current technology is unable to cope with results that requires either to extract a part of a resource (e.g., a scene from a movie) or to aggregate numerous resources (e.g., relevant but scattered information regarding a person).

Furthermore, searching is an *expensive activity*. For instance, in a medium-sized enterprise with 100 employees, each one of them would perform around 10 searches per day (some on the Web, some on their mailboxes, etc.), stopping, successfully or not, in 1-2 minutes. This means that 20-30 hours a day are spent in searching.

¹ Precision is the proportion of relevant data of all data retrieved.

² Recall is the proportion of retrieved relevant data, out of all available relevant data.

What we really need is a search engine able to find any kind of multimedia resource with the required level of granularity; but, how can we achieve this *search engine of the future*? We believe that Tim Berners-Lee was right when, drawing the “Semantic Web Roadmap” [1], he said:

If an engine of the future combines a reasoning engine with a search engine, it may be able to get the best of both worlds.

Keeping in mind Tim Berners-Lee claim, we conceived, implemented and deployed Squiggle (<http://squiggle.cefriel.it>) an extensible semantic search framework designed to add a conceptual flavour at indexing time and to exploit as much as possible ontological elements to improve searching time.

2 Existing Approaches to Improve Search Engines

A standard “syntactic” search engine’s implementation is mainly based on three phases [2]: (a) *crawling time*, the phase in which the resources are collected in order to build a coherent (and more homogeneous) set; (b) *indexing time*, the phase during which the crawled resources are parsed and indexed in some particular data structures, optimized to quickly answer to queries; (c) *searching time*, the run-time phase in which final users submit their queries in order to retrieve meaningful results, possibly ranked and/or clustered.

When the resources to be indexed are multimedia files instead of Web pages, the automatic process of their content becomes very difficult and the lack of links makes crawling a tricky problem and Google PageRank algorithm useless. The two ways out are the use of smart machines and smart data. By *smart machine* we mean a bunch of techniques that includes text processing, audio processing and image/video processing. Several search engines that exploit smart machines are appearing (e.g., Retriever or Musipedia [3]). On the other side, *smart data* is the base for search engines that exploits semantics at search time to increase both recall and precision. I.e., Semantic Web standards offer the possibility of modeling the domain both at lexical and at knowledge level. Explicit representation of semantics gives search engines the ability to disambiguate between homonyms and expand the search to synonyms, pseudonyms and any other relation.

There are several examples of existing approaches that try to *combine Semantic Web technologies with smart machines* in search engines. One of the most interesting is represented by KIM [3], which includes a semantically enhanced information extraction system, which provides automatic semantic annotation with references to classes and instances in the ontology.

3 Our Steps Towards the “Search Engine of the Future”

In our opinion, what Tim Berners-Lee calls the “search engine of the future” should have a structure similar to existing “syntactic” search engines, but should

³ <http://labs.systemone.at/retrievr> and <http://www.musipedia.org/>

also be enriched with machine-processable semantics. In our vision, domain ontologies can be employed in empowering searching, indexing and also crawling.

At *crawling time*, a previous knowledge about the domain can assist the collecting of resources, because this “know-how” can drive the crawler to focus on relevant information even if links are not explicit. At *indexing time*, the input information can be analyzed by means of smart machines and tagged with respect to its meaning before it is processed by the indexer tool. In this way, the tool is able to index both the syntactic content of an input document and its attached semantics. At *searching time*, domain ontologies can be employed to customize search engine applications and to improve the user experience in terms of value added and effectiveness of the search. The tool can help the user in refining his query both by clarifying the matter of his search and by suggesting possible expansions of his query to related subjects. The result is that the user can find more easily what he was looking for.

We conceived Squiggle, keeping in mind Tim Berners-Lee claim and the above analysis. Squiggle is an extensible framework designed to add a conceptual layer to indexing process and to exploit as much as possible ontological elements to improve searching time, leaving to each domain-dependent instantiation of the framework the choice of using ontologies also at crawling time.

4 Conceptual Architecture of the Squiggle Framework

As a result of our analysis and design, Squiggle is:

- a *semantic search-engine*, i.e. a semantic web application with searching functionalities; and
- a *semantic-search engine*, i.e. a search engine that is able to deal with the “meaning” of the searched information.

While the latter objective concerns the semantics of the *data* and can be achieved through an opportune modeling of knowledge, the former purpose is strictly related to the model-driven development of a semantic web *application*.

Squiggle is indeed a semantic application, since its design heavily grounds on a common model, provided by SKOS [4], which permeates all its structure; Squiggle assumes SKOS as its application model and, therefore, is naturally able to exploit SKOS’ semantics. Moreover, being SKOS a horizontal ontology (therefore not bound to any specific subject), Squiggle is completely domain-independent and can thus serve as a *framework* to build domain-specific search applications. In essence, Squiggle is not a search engine itself, but it allows users to customize their own engine on the basis of a particular domain knowledge.

Squiggle is designed to provide both syntactic and semantic indexing and searching primitives, seamlessly combining the speed of syntactic search tools with improved recall and precision, based on the ability to assign alternative designations and wordings in multiple languages to their meaning. Among the constituents of Squiggle, Sesame [5] is used as the semantic engine that queries the knowledge base, whereas the syntactic search engine Lucene [6] is used to

quickly perform text searches. Therefore the described architecture lends itself well both to overcome the limitations of purely syntactic approaches and to improve the performance of semantic engines.

Technically speaking, Squiggle's innovation consists in its Conceptual Indexing and Semantic Search capabilities. The *Conceptual Indexing* consists of a *semantic annotation process*, during which the input information is scanned and analyzed in order to identify and extract the concepts that characterize it (Squiggle expects resources to be annotated with keywords and it searches in the domain ontology for concepts whose SKOS labels match those keywords), and of an *indexing process*, during which these concepts are stored in an index for subsequent search and retrieval. On the other hand, the *Semantic Search* analyzes user's queries and tries to identify the ontological elements that can be related to the request, "suggesting" to the user the potential *meanings* of his query; the user is therefore presented with both the results of the syntactic search and the available meanings extracted from the query, which can help him to refine his request, "disambiguating" among its the possible acceptations. Moreover, when a user query is re-conducted to a specific meaning, Squiggle also seeks other concepts that could be of interest for the user; this is possible because Squiggle Semantic Search can navigate across the graph of interconnected elements of the domain ontology, following "semantic paths" denoted by relations and attributes.

5 Squiggle Real-World Deployments

In order to prove the feasibility of our approach, we briefly present some test beds. We successfully developed some search engines on top of the Squiggle framework, in different application fields.

Squiggle Ski – CEFRIEL, as Official Supplier of the XX Olympic Winter Games for Applied Academic Research, caught the opportunity to demonstrate Squiggle in the context of CEFRIEL's activities related to Torino 2006⁴. Our aim in deploying Squiggle Ski, a service available on CEFRIEL's portal, was to help the international public of Torino 2006 in finding images of the athletes involved in the alpine skiing races.

Squiggle Ski is on-line at <http://squiggle.cefriel.it/ski>; during Torino 2006 event, it was visited by almost one thousand visitors searching for the various athletes that won a medal in the alpine-skiing races. When you open the home page, you are presented with an ordinary search box. If you try searching for "Herminator abfahrt" (being "Herminator" a nickname for Hermann Maier and "abfahrt" the German for downhill), you receive a plain syntactic search, and in a box on the right Squiggle Ski asks if you mean the athlete "Hermann Maier" and the discipline "downhill". If you eventually follow Squiggle Ski suggestions, all the images of Hermann Maier in a downhill race are retrieved, disregarding

⁴ See also <http://www.cefriel.it/press/olimpiadi2006.html>

the language used in the initial query; an explanation box shows how Squiggle Ski expanded the query to achieve the result.

Squiggle Music – Squiggle Music is an instantiation of Squiggle framework in the music field. We noticed that both very diffuse media-players and popular sites for buying music fail to retrieve tracks when alternative wordings or translations are used (e.g. searching “rhcp” does not always retrieve the list of all Red Hot Chili Peppers tracks in the repository). Squiggle Music indexes audio files (mainly mp3 files) enriching them with information about authors, song titles and music genres. Squiggle Music is publicly available at <http://squiggle.cefriel.it/music>. Combining the *smart data* from MusicBrainz and MusicMoz⁵ with a *smart machine* like QuickName⁶ that makes use of audio fingerprints, we built an automatic semantic annotator that acts as a domain-dependent plug-in of Squiggle framework during the *Conceptual Indexing* phase. This annotator is therefore able to add to each file all its metadata (artist, song title, etc.).

From the final user’s point of view, besides the usual “suggestion” of meanings, Squiggle Music is able to perform a query expansion and to present the user with other results that could be of his interest. Squiggle Music can suggest related artists when searching for a performer, songs by the same artist when looking for a song, broader and narrower styles when asking for a music genre.

6 Conclusions

In this paper, we presented Squiggle, a Semantic Web framework that eases the deployment of semantic search engines in specific applications. We enlightened how the employment of Semantic Web technologies to the development of search engines provides real benefits to end users, enabling an easier and more effective access to information; a semantic search engine, in facts, improves current syntactic engines in terms of both precision and recall, thanks to an explicit characterization of the domain at lexical and conceptual level. Semantic Web technologies show their whole potentialities in the expansion of queries to include related meanings: a semantic search engine built on Squiggle appears to be more usable, in that users are supported with semantic “suggestions”, as our test-beds demonstrate at a glance.

Moreover, we designed Squiggle foreseeing possible extensions to the framework: for example, the adoption of smart machines allows the exploitation of their media-dependent capabilities and, in the meantime, the generation and aggregation of smart data.

Finally, we admit that a semantic search engine developed with Squiggle is strongly domain-dependent and cannot compete with general-purpose search engines; however, we definitely believe that such an approach provides better results, because a focused tool better meets specialized needs, helping you in finding what you’re really looking for.

⁵ <http://www.musicbrainz.org/> and <http://www.musicmoz.org/>

⁶ <http://phonascus.sourceforge.net/>

Acknowledgements

This research has been partially supported by the NeP4B Italian-funded FIRB project (MIUR-2005-RBNE05XYPW).

References

1. Berners-Lee, T.: Semantic Web Road map (1998), Available on the web at <http://www.w3.org/DesignIssues/Semantic.html>
2. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* 30, 107–117 (1998)
3. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. *Elsevier's Journal of Web Semantics* 2(1) (2005)
4. Miles, A., Brickley, D.: SKOS Core Guide, W3C Working Draft (November 2, 2005), <http://www.w3.org/TR/swbp-skos-core-guide>
5. Kampman, A., van Harmelen, F., Broekstra, J.: Sesame: A generic architecture for storing and querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) *ISWC 2002*. LNCS, vol. 2342, Springer, Heidelberg (2002)
6. Gospodnetic, O., Hatcher, E.: *Lucene in action*. Manning Publications (2004)

WebTE: MDA Transformation Engine for Web Applications

Santiago Meliá¹, Jaime Gómez¹, and José Luís Serrano²

¹ Universidad de Alicante, Spain

² Caja de Ahorros de Alicante, Spain

santi@dlsi.ua.es, jgomez@dlsi.ua.es, jlserrano@cam.es

Abstract. Transformations are of crucial importance for the success of Model-Driven Web Engineering (MDWE) approaches. Therefore, we need transformation engines to improve and obtain the results of the different approaches. However, very few model-driven Web approaches provide a transformation tool which would allow them to obtain an implementation from their models. In this paper, we present a tool called WebTE (WebSA Transformation Engine) which is able to introduce all the input artefacts of the WebSA approach and to establish a refining process based on model-to-model and model-to-text transformations which gives us the final implementation of a Web application.

1 Introduction

Model-Driven Engineering (MDE) is becoming a widely accepted approach for Web Engineering discipline. In fact, existing model-based Web engineering approaches currently provide excellent methodologies focused on the functional aspects. However, due to the novelty of this discipline, there are some aspects which have not been tackled yet: (1) the architectural aspects which would permit to obtain some quality attributes such as distributed computation, scalability, maintenance, connectivity with legacy systems, etc. (2) Traceability from the design models to the implementation. (3) The existence of too many notations to represent the same functional concepts in the different Web methods. (4) Currently, there is not any tool which allows us to define a complete model-driven approach based on transformations to obtain the final implementation.

To overcome these limitations, WebSA [7] defines a specific development process for Web Applications which regards software architecture artefacts as first-class citizens introducing automation mechanisms that accelerate this process. To do so, this approach defines a set of architectural models which complements the functional models of other Web methodologies providing a complete specification of the application. From these models, this approach starts an automatic and traceable process through a set of model-to-model transformations which carries out the integration of the architectural and functional aspects in a design model. In the last step, the process defines a set of model-to-text transformations which converts the integrated design model into different platform implementations. In order to give support to this approach, we have implemented a Web Tool called WebTE [12] which is basically a

Web application which, through Web pages, allows us to introduce WebSA's models and transformations into a transformation engine which will execute them and give us the result of the WebSA process.

Section 2 gives an overview of the WebSA development process. Section 3 presents the specification of the WebTE tool and its user interface. Finally, section 4 presents some future steps and the conclusions.

2 The WebSA Approach: An Overview

WebSA [7] is a proposal whose main objective is to cover all phases of Web application development focusing on software architecture. It contributes to fill the gap currently existing between traditional Web design models and the final implementation. In order to achieve this, it defines a set of architectural models to specify the architectural viewpoint which complements current Web engineering methodologies.

The WebSA development process is based on the MDA development process [4], which includes the same phases as those included in the traditional life cycle of an application (analysis, design and implementation). However, unlike in the traditional life cycle, in the MDA approach [10] the artefacts that result from each phase must be models, which represent the different abstraction levels in the system specification. In the analysis phase the Web application specification is vertically divided into two viewpoints. On the one side, the functional-perspective is given by the Web functional models provided by approaches such as WebML [2], OO-H [3] or UWE [6]. On the other side, the Subsystem Model (SM) and the Configuration Model (CM) define the software architecture of the Web Application. Defining the application architecture orthogonally to its functionality allows for its reuse in different Web applications. The models-to-model transformation which goes from analysis models to platform independent design model provides a set of artefacts in which the conceptual elements of the analysis phase are mapped to design elements. It integrates the information about functionality and architecture in a single Integration Model (IM). This transformation type will be called T1 in the rest of the article. The Integration Model is the basis on which several model-to-text transformations, one for each target platform, can be defined. The output of these mode-to-text transformations is the implementation of the Web application for a given platform. This transformation type will be named T2 in the rest of the article.

3 WebTE: WebSA Transformation Engine

To give a support to the WebSA approach, we have implemented a Web tool called WebTE (WebSA Transformation Engine) [12] which is developed using the J2EE platform. The WebTE tool is based on the standards provided by the OMG (UML, XMI, MOF and OCL) mainly for optimizing the implementation effort, through off-the-shelf components and facilitating the use of any type of UML tool that possesses the support for class diagrams.

The main characteristic of the WebTE Tool is that it is a Web application and can therefore be used in a remote way. This is due to the fact that this tool does not need to provide a modelled graphic interface, because models and transformations are

specified from any UML tool and it is from them that their representation can be generated in XMI. Besides, the text transformations are sent using text files.

Fig. 1 describes a complete process execution of the WebTE application, representing the different artefacts and components, each of which performs a task within the WebSA process. We will start the description in a gradual form using numbers that indicate the order of the process from source models to implementation.

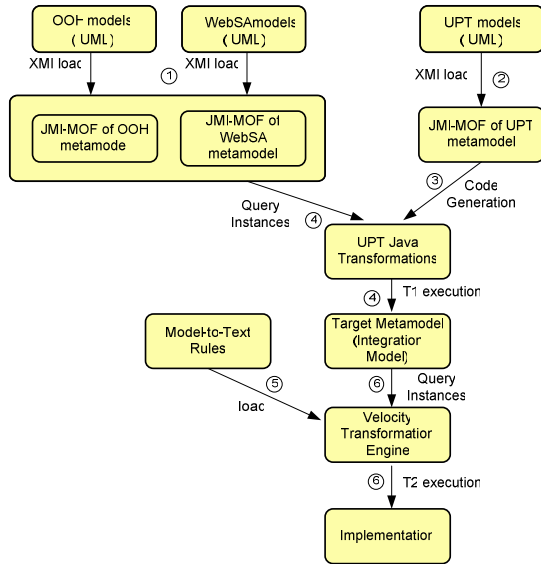


Fig. 1. The process of the WebTE Tool

It consists of the following steps:

1. The Web modeller expert defines the UML models of WebSA and the OO-H functional Web method using a UML tool, and then he/she generates the XMI representation which is loaded by the XMI parse of the JMI metamodels. The MOF metamodels of WebSA and the Web approaches are defined in JMI (Java Metamodel Interface) [4]. The use of JMI has two advantages: (1) it manages the instances of the MOF metamodel classes, that is, it makes it possible to manage models which are compliant with the metamodels, and (2) it provides the parses for reading and writing from XMI documents of UML that represent the models.
2. The Transformation expert establishes the UML transformations using the UPT language [8] in any UML tool capable of generate XMI documents of UML. The tool imports the XMI of the UPT transformations and from them creates the instances of the UPT metamodel.
3. It initiates a process of compilation in order to obtain the transformation in Java code and its subsequent compilation. This step is optional, because once the WebSA transformations have been generated, the rest of users can reuse them. However, in some cases, the user could be interested in introducing an extension of the WebSA transformations for a specific Web application.

4. The T1 transformation (models-to-model) is launched to convert the instances of JMI origin metamodels (WebSA and OOH) into the instances of the JMI target metamodel that correspond to the WebSA Integration Model.
5. The last phase of the WebSA process starts at this point with the T2 transformation which transforms the Integration Model into the platform implementation using model-to-text rules. To do this, the WebSA model-to-text transformation rules are written and loaded into the WebTE tool using text files. This step is also optional, because the WebTE tool has a complete set of predefined model-to-text rules for the most important platforms such as J2EE and .NET. However, a user could introduce new rules that are specific for his application.
6. Finally, the T2 transformation is executed and the instances of the WebSA metamodel are queried to generate the implementation of a specific platform (J2EE or .NET).

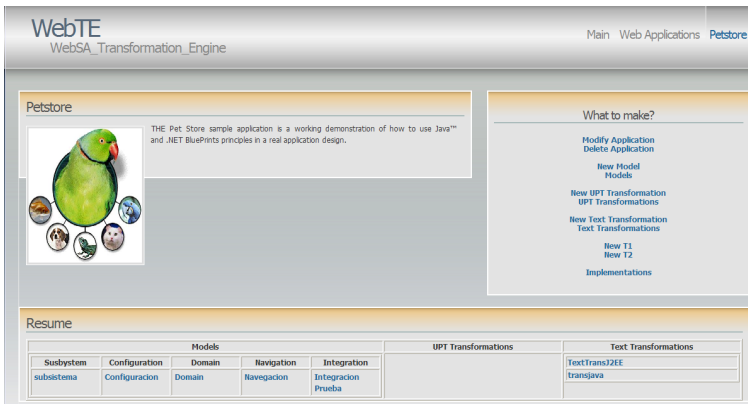


Fig. 2. Main Page of WebTE for Petstore Web Application

Fig. 2 depicts a snap-shot of the WebTE tool. This Web page presents the main page of the well-known Web application Petstore [11] where we find an image of the application, a description and a resume of its artefacts (models, transformations and executions) that have been stored in the WebTE tool up to now.

Besides, on the right side, we can see a table which has a set of links with all the different actions that a user can perform using the WebTE Tool such as: *Modify Application* and *Delete Application* which allow us to manage a created application. *New Model and Models* permit to create and query the models, respectively. *New UPT Transformation and UPT Transformation* create and query the new model-to-model transformations specific for the Petstore application. *New Text Transformation and Text Transformation* create and query the new model-to-text transformations specific for the Petstore application. *New T1* starts the execution of the WebSA T1 transformation as does *New T2* for the WebSA T2 transformation. Finally, the *Implementation* link permits to query the set of Petstore implementations obtained up to now.

Up to now, WebTE only supports the Web functional models defined in the OO-H approach. However, WebTE could be easily extended to other functional approaches

using the Netbeans MDR [9] framework which allows us to obtain automatically a JMI metamodel from a XMI-MOF metamodel defined in any UML tool.

4 Conclusions and Future Works

In this paper has been presented a Web tool called which WebTE is based on three main aspects: (1) support of the models and transformations of the WebSA approach. (2) Extensibility for adding and modifying models and transformations rules specific for a user or a system. (3) Interoperability between different tools in order to obtain different artefacts from them using XMI. WebTE offers a great tool compatibility because the models and transformations can be represented in any UML tool and can also be shared using XMI.

Today, this tool has been applied to an MDA proposal as WebSA and a Web Functional Method as OO-H but it could be applied to any approach which defines its metamodel in the standard MOF. In our future work, we will try to increase the number of possible users of the WebTE tool introducing other Web methodologies meta-models such as UWE [6], WebML [2] and W2000 [1]. It allows us to introduce their functional aspects into the WebSA development process.

References

1. Baresi, L., Garzotto, F., Paolini, P.: Extending UML for Modeling Web Applications. In: Proceedings of the 34th International Conference on System Sciences (2001)
2. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing* 6(4), 20–30 (2002)
3. Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. *IEEE Multimedia* 8(2), 26–39 (2001)
4. Java Metadata Interface (JMI) (2006), <http://java.sun.com/products/jmi/>
5. Kleppe, A., Warmer, J., Bast, W.: *MDA Explained: The Model Driven Architecture, Practice and Promise*. Addison-Wesley, London, UK (2003)
6. Koch, N., Kraus, A.: The expressive Power of UML-based Web Engineering. In: 2nd IWWOST'02, CYTED. June 2002. pp. 105–119 (2002)
7. Meliá, S., Gomez, J.: The WebSA Approach: Applying Model Driven Engineering to Web Applications. *Journal of Web Engineering*, © 5(2), Rinton Press, 121–149 (2006)
8. Meliá, S., Gomez, J.: UPT. A Graphical Transformation Language based on a UML Profile. In: Proceedings of European Workshop on Milestones, Models and Mappings for Model-Driven Architecture (3M4MDA 2006). 2nd European Conference on Model Driven Architecture (EC-MDA 2006) (July 2006)
9. Metadata Repository Project HOME (MDR) [architecture.html](http://mdr.netbeans.org/architecture.html) (2006), <http://mdr.netbeans.org/>
10. OMG. MDA Guide, OMG doc. ab/2003-05-01
11. TM J2EE Blueprint. Java Petstore 1.1.2 (November 2004) http://developer.java.sun.com/developer/releases/petstore/petstore1_1_2.html
12. WebTE., <http://ebusiness.dlsi.ua.es:8080/WebSATool/>

Noodles: A Clustering Engine for the Web

Giansalvatore Mecca, Salvatore Raunich,
Alessandro Pappalardo, and Donatello Santoro

Dipartimento di Matematica e Informatica
Università della Basilicata
Potenza – Italy

Abstract. The paper describes the Noodles system, a clustering engine for Web and desktop searches. By employing a new algorithm for document clustering, based on Latent Semantic Indexing, Noodles provides good classification power to simplify browsing of search results by casual users. In the paper, we provide some background about the problem of clustering search results, give an overview of the novel techniques implemented in the system, and present its architecture and main features.

1 Background

Web and desktop search services are nowadays essential tools for any Internet user. However, keyword-based, boolean-style search engines like Google usually fall short when asked to answer rather broad queries – those that are often posed by less-experienced users – like, for example, to find documents about the term “*power*” or the term “*amazon*”. The poor quality of results in these cases is mainly due to two different factors: (a) polysemy and/or synonymity in search terms (b) excessively high number of results returned to the user. As a consequence, less skilled users are often frustrated in their search efforts.

The Semantic Web promises to solve most of these problems by adding semantics to Web resources. In fact, there have been some proposals in the literature towards a semantic Web search engine [3]; these proposals assume that documents can be classified based on their RDF or OWL annotations, and therefore are not immediately applicable. As a consequence, at the moment the most promising efforts along the way to Semantic Web search services are the so-called *clustering engines*.

The idea of clustering search results is not new, and has been investigated quite deeply in Information Retrieval, based on the so called *cluster hypothesis* [10] according to which clustering may be beneficial to users of an information retrieval system since it is likely that results that are relevant to the user are close to each other in the document space, and therefore tend to fall into relatively few clusters. Several commercial clustering engines have recently emerged on the market. In fact, even Google has experimented for a while with forms of clustering of their search results [11], and has recently introduced a “Refine Your Query” feature¹ that essentially allows to select one of a few topics to

¹ <http://www.google.com/help/features.html#refine>

narrow a search. Similar experiments are also being conducted by Microsoft [8]. Other well known examples of commercial clustering engines are Vivisimo [9] and Grokker [7]. These systems share a number of common features with research systems introduced in literature, mainly the Grouper system [12,13] and Lingo/Carrot Search [5,6]. We summarize these features in the following.

First, these tools are usually not search engines by themselves. On the contrary, when a user poses a query, the clustering engine uses one or more traditional search engines to gather a number of results; then, it does a form of post-processing on these results in order to cluster them into meaningful groups. The cluster tree is then presented to the user so that s/he can browse it in order to explore the result set. It can be seen that such a technique may be helpful to users, since they can quickly grasp the different meanings and articulations of the search terms, and more easily select a subset of relevant clusters.

Being based on a post-processing step, all of these clustering engines work by analyzing *snippets*, i.e., short document abstracts returned by the search engine, usually containing words around query term occurrences. The reason for this is performance: each snippet contains from 0 to 40 words, and therefore can be analyzed very quickly, so that users do not experience excessive delays due to the clustering step. However, snippets are often hardly representative of the whole document content, and this may in some cases seriously worsen the quality of the clusters.

2 The Noodles System

The Noodles system is a clustering engine for Web and desktop searches. It is based on a novel algorithm for clustering search results [4]. Figure 1 shows clusters produced by the system for a Google search on term “amazon”.

One of the main features of the system is that it is not based on snippets, but it requires access to the whole document contents and uses a different compression techniques to improve performance. A key objective of the Noodles clustering technique is to achieve a high level of quality in terms of its ability to correctly *classify* documents, i.e., to dynamically build a bunch of clusters that correctly reflect the different categories in the document collection returned by the search engine. Similarly to [1], we believe that this ability may assist less-skilled users in browsing the document set and finding relevant results.

The clustering algorithm that has been developed is called *Dynamic SVD Clustering (DSC)* [4], and it is based on Latent Semantic Indexing [2]; the novelty of the algorithm is twofold: (a) first, it is based on an incremental computation of singular values, and does not require to compute the whole SVD of the original matrix; (b) second, it uses an original strategy to select k , i.e., the number of singular values used to represent the “concepts” in the document space; differently from other proposals in the literature, our strategy does not assume a fixed value of k , neither a fixed approximation threshold.

In [4], based on experimental results, we show that the algorithm has very good classification power; in many cases it is able to cluster pre-classified documents

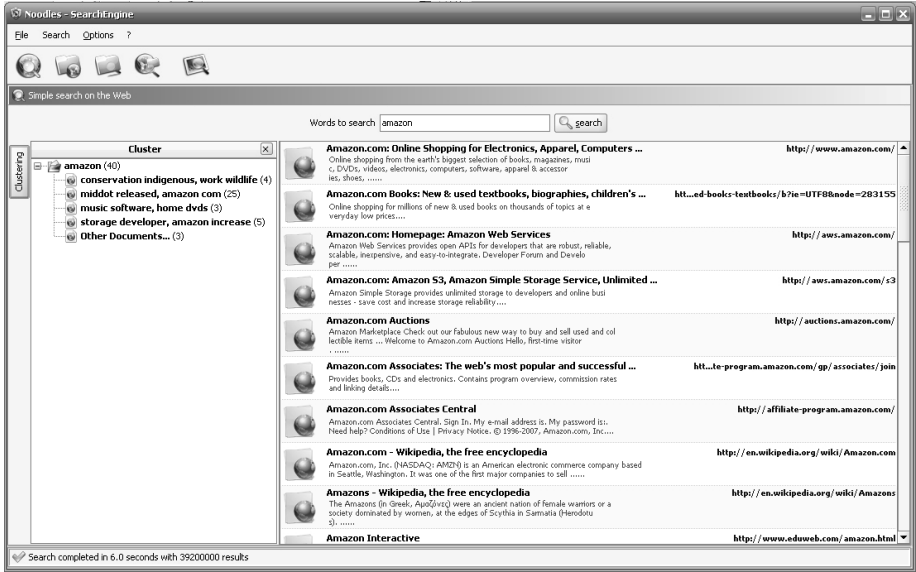


Fig. 1. A Snapshot of the Noodles Search Engine

collections with very good accuracy; it is worth noting that the quality of the classification severely degrades when snippets are used in place of the whole document content, thus providing further evidence that snippets are often too poor and not sufficiently informative.

Note that such good classification power has a cost in terms of computing times, since the SVD-based computation has higher complexity with respect to typical phrase analysis algorithms based on snippets. However, the algorithm has been designed in such a way that in practice it has good performance, and lends to a very natural clustering strategy based on the minimum spanning tree of the projected document space. A detailed report on experimental results is available in [4].

3 Architecture of the System

The system is fully written in Java. It comprises several modules and some well-known Java libraries, as shown in Figure 2.

Spring² has been adopted as a dependency-injection framework. This greatly helped the development in several respects; on the one side it improved the overall system modularity; on the other side, it helped to experiment different strategies for various aspects of the clustering algorithm. For the development of the desktop user interface, we adopted the Spring Rich Client Platform, which

² <http://www.springframework.org>

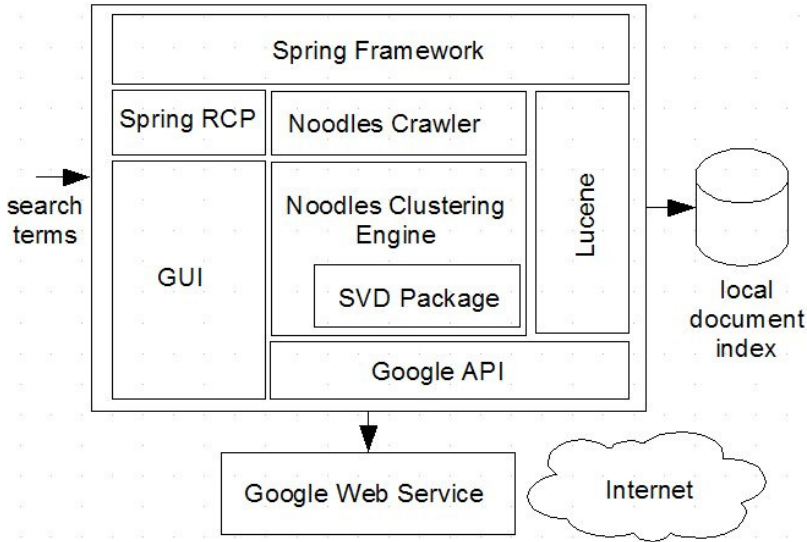


Fig. 2. Architecture of the System

is tightly coupled to the dependency-injection framework, and provides further ease of configuration and an effective binding framework for forms and wizards.

The system supports both Web and desktop searches. Web searches use Google via its web service API. To simplify the experimental phase, it also incorporates a module to sample categories in the DMOZ directory (dmoz.org).

Desktop searches use Apache Lucene (lucene.apache.org) as an indexing engine. The Noodles crawler runs as a daemon and incrementally inspects the local disk to feed files to Lucene for indexing purposes. Its behavior is fully customizable: users may select what portion of the local disk they intend to index.

At the core of the system stands the Noodles clustering engine, which implements the dynamic SVD clustering algorithm. In order to perform SVD computations, COLT (dsd.lbl.gov/~hoschek/colt) was selected as a matrix manipulation package, although we are performing further experiments using JScience (www.jscience.org), the newest math package for the Java language; in our experiments, JScience – which is based on a high performance library called Javolution (javolution.org) – showed significantly better performance both in terms of computing time and heap usage with respect to COLT and other similar packages. Unfortunately, JScience currently does not offer support for SVD, although this might be added to future version. Nevertheless, we believe that JScience might significantly improve computing times once support for SVD is implemented.

In implementing the system, special care was devoted to reduce computation times. For example, we developed our user interface in such a way to minimize the latency associated with the display of clusters. More specifically, when a user runs a query, the system very quickly returns the ranked search results. At the same time, it starts to cluster top-ranked results on a background thread, and

shows a “Clustering” button on the screen. If the user wants to see the clusters, s/he has to select this button. Considering typical user reaction times, the net effect of such an organization of the user interface is that, in the user perception, the clustering is almost immediate.

4 Features of the Prototype

We will demonstrate the main features of the prototype by running several searches, both on the Web and on the local disk, and more specifically: (a) by performing searches based on typical polysemic terms, like “power”, “amazon”, and “life”, and showing clusters produced by the system; (b) by performing searches based on pre-classified documents, sampled from DMOZ, and showing how the system is typically able to correctly reproduce the DMOZ classification. Free Web and desktop searches will be used to assess the quality of the clustering based on user feedbacks.

References

1. Chekuri, C., Raghavan, P.: Web Search Using Automatic Classification. In: Proceedings of the World Wide Web Conference (1997)
2. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Sciences* 41(6), 391–407 (1990)
3. Guha, R., Mc Cool, R., Miller, E.: Semantic Search. In: Proceedings of the World Wide Web Conference (2003)
4. Mecca, G., Raunich, S., Pappalardo, A.: A New Algorithm to Cluster Search Results. Data and Knowledge Engineering, Available as Noodles WR-01-2006 (to appear) at <http://www.db.unibas.it/projects/noodles>
5. Osinski, S., Stefanowski, J., Weiss, D.: Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition. In: Proceedings of the International Conference on Intelligent Information Systems (IIPWM) (2004)
6. Osinski, S., Weiss, D.: A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems* 20(3), 48–54 (2005)
7. The Grokker Search Engine: <http://www.grokker.com>
8. The SRC Search Engine: <http://rwm.directtaps.net/>
9. The Vivisimo Search Engine: <http://www.vivisimo.com>
10. van Rijsbergen, C.J.: *Information Retrieval* (2nd edn.) London, Butterworths (1979)
11. Web 2.0 – Exclusive Demonstration of Clustering from Google. <http://www.searchengineardown.com/2004/10/web-20-exclusive-demonstration-of.html>
12. Zamir, O., Etzioni, O.: Web Document Clustering: A Feasibility Demonstration. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) (1998)
13. Zamir, O., Etzioni, O.: Grouper: A Dynamic Clustering Interface for Web Search Results. *Computer Networks* 31(11–16), 1361–1374 (1999)

WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications

Roberto Acerbis¹, Aldo Bongio¹, Marco Brambilla², and Stefano Butti¹

¹ WebModels S.r.l.

Piazzale Gerbetto, 6. I22100 Como, Italy

{roberto.acerbis, aldo.bongio, stefano.butti}@webratio.com

² Dipartimento di Elettronica e Informazione, Politecnico di Milano

Piazza L. Da Vinci, 32. I20133 Milano, Italy

mbrambil@elet.polimi.it

Abstract. The goal of this work is to present the software WebRatio 5, which is a good representative of a new generation of CASE tools for model-driven design of Web applications. WebRatio 5 supports the WebML language and methodology, and exploits the implementation experience of previous versions of the software for providing user-friendly application design paradigms and reliable code generation engines. The tool is developed as a set of Eclipse plugins and takes advantage of all the features of this IDE framework. Moreover, it provides new capabilities in terms of support of extensions to the models, project documentation, and coverage of new phases of the development lifecycle. The overall approach moves towards a full coverage of the specification, design, verification, and implementation of Web applications.

1 Introduction and Motivation

Although new paradigms of Web applications are arising, data-intensive Web applications still constitute the most diffused class of applications found on the Web. Since their size and complexity are typically high, the ideal software development process for this kind of applications should meet two goals: (i) incorporate requirements and model driven design in the development lifecycle; (ii) delivering a software architecture that meets the non-functional requirements of performance, security, scalability, availability, maintainability, usability, and high visual quality. Such process should also be amenable to automation, to let developers concentrate on functional requirements and optimization, and delegate the repetitive tasks (such as code implementation) to software tools.

The model-driven design of this kind of Web applications should start from well established requirement specifications and involves the definition of a data model (to specify the data used by the application), a hypertext model (to describe the organization of the front-end interface) and a presentation model (to personalize the graphical aspect of the interface). Afterwards, model verification and model transformations (e.g., for generating the running code) should be provided to complete the development process. Unfortunately, no existing CASE tool can claim to support all these aspects of the development.

In this work we present the CASE software WebRatio 5 [10], representing a new generation of model-driven CASE tools for Web applications. WebRatio 5 supports the WebML language and methodology, and exploits the implementation experience of previous versions of the software for providing user-friendly application design paradigms and reliable transformation engines. The tool is developed as a set of Eclipse plug-ins and takes advantage of its features. Moreover, it provides new capabilities in terms of support of extensions to the models, project documentation, and coverage of new phases of the development lifecycle.

The main advantages of the new Eclipse version of WebRatio are the following:

- All the design and development activities are performed through a common interface. This includes the modeling of the Web application, the definition of the visual identity, and the development of new business components;
- All the design items (models, components, documentation, and so on) are stored into a common area (the so-called Eclipse workspace) and can be easily versioned into a versioning system, such as CVS;
- All the existing Eclipse widgets can be reused and integrated in the toolsuite;
- New and existing editors for model and code design can be easily integrated.

The following sections outline the main features of the tool, provide some GUI examples and describe the overall philosophy of the WebRatio toolsuite, more and more moving towards a full coverage of development process of Web applications.

2 Supporting the Design of WebML Models

WebRatio 5 fully supports the WebML metamodel [9] [3], including the most recent extensions for workflow-driven Web applications [2] and Web services [8]. WebML is a high-level notation for data-, service-, and process- centric Web applications. It allows specifying the data model of a Web application and one or more hypertext models (e.g., for different types of users) used to publish and manipulate the underlying data. Each hypertext is a graph of *pages*, consisting of connected *units*, representing at a conceptual level the primitives for publishing contents into pages. Units are connected by *links*, that define navigation paths and carry data to allow computation of the hypertext. Hypertexts also include *operations* that specify business actions, such as content management operations on the data or other kinds of tasks.

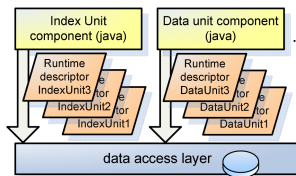


Fig. 1. Java components and XML descriptors for WebRatio units

3 WebRatio 5 Architecture

The *design-time* part of WebRatio 5 is a GUI for application design comprising a set of editors, a set of transformers and model validators, and some workspace management components. Models are saved as XML documents. The model is transformed into a running Web application through a code generator, which is developed using the ANT, XSLT, and Groovy technologies. Groovy [7] is an agile language using a Java-like syntax and fully integrated in the Java Platform. It provides many features that are inspired by scripting languages. The tool is integrated with CVS [1] for collaborative design and visual synchronization of project versions. The architecture is fully extensible, since it allows to specify new components (units) and include them in the application design and code generation framework. At design time, the components are described through a Java class that implements the service of the component and by a set of XML descriptor, defining its interface.

The *run-time* framework exploits a set of off-the-shelf object-oriented components for organizing the business tier:

- *Smart service creation*: services that implement units or business actions are created upon request, cached, and reused across multiple requesters;
- *Activity log*: a set of pre-built functions for logging each service is provided;
- *XML parsing and access*: access to the information stored in the XML unit descriptors is granted by standard parsing tools;
- *Connection pooling*: pre-built functions for dynamically managing a pool of database connection allow to optimize performance and reliability.

At runtime one single service class is deployed for each type of component (which is then instantiated with the smart service creation approach). Moreover, one runtime XML descriptor is deployed for each component used in the design (Fig. 1).

4 WebRatio 5 GUI

WebRatio 5 has been implemented as a set of Eclipse [5] plug-ins. Eclipse is a framework for IDEs, in the sense that, besides being an IDE itself, it provides the infrastructure for defining new IDEs, i.e., new plug-ins for a particular programming language or model. For instance, plug-ins exist for Java, C/C++, Python, Perl, UML, and many others. Eclipse is an open source multi-platform framework, executable on Linux, Windows, and Mac OS X.

The WebRatio GUI defines a special Eclipse perspective designed to better suit the needs of visual modelling. It comprises several panels, which include:

- *Model diagram editors* for the design of the WebML data model and hypertext models. The diagram editors are based on the GEF [6] framework and libraries. GEF is a very powerful framework for visually creating and editing models. Fig. 2 (a) shows a snapshot of the data model editor, comprising the project tree, the main diagram editor, the component panel (bottom left), the property panel (center), and the error panel (bottom right);

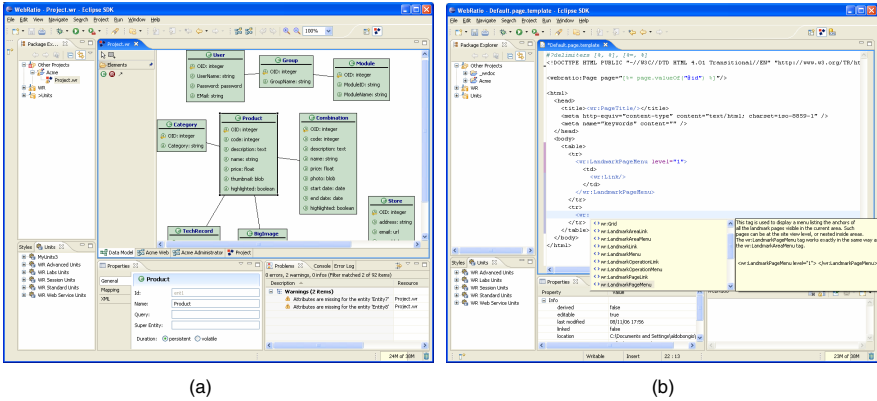


Fig. 2. Data model editor and HTML editor (with code completion) in WebRatio 5

- *Advanced text editors* for the design of XML descriptors, Java components, and so on. The editors provide typical features like syntax highlighting, auto-completion, and so on. An example is shown in Fig. 2 (b);
- *Form-based editors* for the specification of new components and for the properties of components instances. Fig. 3 (a) shows the editor of Indexes;
- *Wizards* for the support of the most common design tasks (e.g., new projects);
- *Documentation editors* for refined and customized project documentation generation. For instance, Fig. 3 (b) shows the editing and the generated documentation for the Index unit component.

The long-term focus of the WebRatio 5 is oriented towards the full coverage of the development process. In this sense, some new *beta* pieces are being developed, for project documentation generation and coverage of new design steps. For instance, a new fully-integrated Business Process editor and WebML model generator [1] is now available (Fig. 4 shows a sample snapshot). It generates skeletons of WebML models that comply with the BP specification and can be refined by the designer.

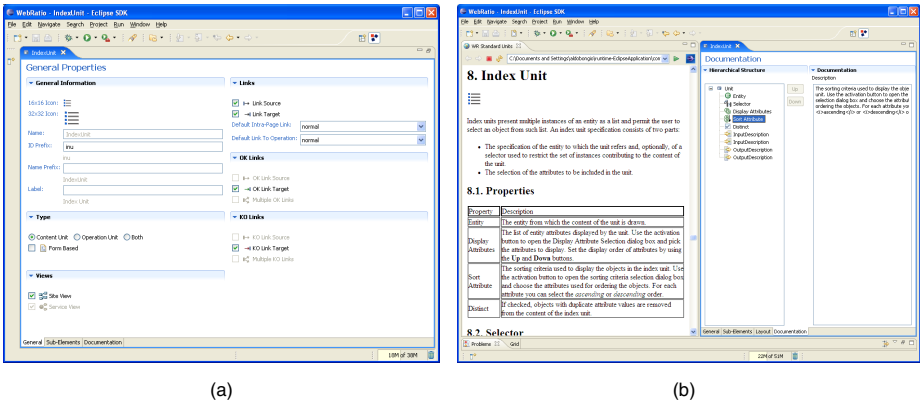


Fig. 3. Component definition editor and project documentation editor in WebRatio 5

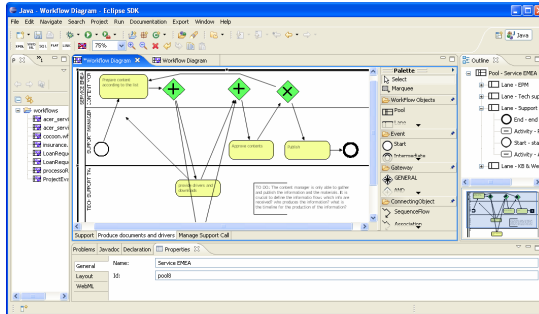


Fig. 4. The BPMN business process editor for Eclipse

5 Conclusions

The proposed demonstration illustrates WebRatio 5, a CASE tool based on the Eclipse framework that allows the model-driven specification of a complex Web application, including process management primitives, calls to Web services, and integration of heterogeneous data sources. The demonstration shows that application developers can concentrate only on the requirements of the application and on its high-level design, because code and project documentation are automatically generated by the CASE tool and correctness is automatically verified.

References

- [1] Brambilla, M.: Generation of WebML Web Application Models from Business Process Specifications. Demo at ICWE2006, pp. 85–86. ACM Press, New York (2006)
- [2] Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. In: ACM TOSEM, vol. 15(4), pp. 360–409 (2006)
- [3] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, San Francisco (2002)
- [4] CVS, Concurrent Versions System (2007) <http://www.nongnu.org/cvs/>
- [5] Eclipse (2007) <http://www.eclipse.org/>
- [6] Eclipse GEF (2007) <http://www.eclipse.org/gef/>
- [7] Groovy (2007) <http://groovy.codehaus.org/>
- [8] Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications. ACM TOIT 5(3), 439–479 (2005)
- [9] WebML.org. (2007) <http://www.webml.org>
- [10] WebRatio 4.3. (2007) <http://www.webratio.com/>

Extending Ruby on Rails for Semantic Web Applications

Cédric Mesnage¹ and Eyal Oren²

¹ Faculty of Informatics
University of Lugano, USI
Lugano, Switzerland

`cedric.mesnage@lu.unisi.ch`

² Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland
`eyal.oren@deri.org`

Abstract. We extend the Ruby on Rails framework towards a more complete Semantic Web application framework. The SWORD plugin provides developers with a set of tools and libraries for managing Semantic Web data and rapid Semantic Web Application development. We describe the functionality of the SWORD plugin and demonstrate its use for rapid development of a social networking application.

1 Introduction

The Semantic Web is a web of data that can be processed by machines, enabling them to interpret, combine and use Web data [1,2]. The Semantic Web offers a uniform way of identifying and describing resources in a distributed environment, and thus increases the interoperability between applications.

On the other hand, web development has evolved recently to a more structured and abstract way of engineering web applications with the appearance of frameworks such as Struts [1], Ruby on Rails [2] and Django [3]. These frameworks overcome the problem of the separation of concerns regarding data manipulation, user interaction and business logic by constructing web applications on the model–view–controller design pattern [7,3].

Although these frameworks ease the development of web applications, they offer only limited support for interoperability. Since Web applications are mostly data-driven, we see increasing interoperability and data reuse through “mash-ups” that combine data from multiple Web applications into new functionality. Such “mash-ups” are supported only to a limited extent by existing frameworks. We improve the possibilities of data reuse in Web application frameworks by using Semantic Web data as a basis for application development.

¹ <http://struts.apache.org>

² <http://rubyonrails.org>

³ <http://www.djangoproject.com>

We focus on the Ruby on Rails, an agile development platform for Web applications. Ruby on Rails provides solutions for rapid prototyping and is supported by an active community. Ruby on Rails is currently a popular platform for development of Web 2.0 applications; we show how it can be extended to support Semantic Web application development. In this paper, we review the related work, we present the functionalities and architecture of the SWORD⁴ plugin and demonstrate how SWORD enables rapid prototyping of a social networking application.

2 Related Work

In [5], Lima and Schwabe present the SHDM (Semantic Hypermedia Design Method) design approach, an extension of their earlier OOHD (Object Oriented Hypermedia Design Method) approach. SHDM is targeted for the design of Web applications for the Semantic Web, replacing the conceptual models of OOHD with Semantic Web ontologies. To our understanding, applications designed using the SHDM method keep a relational database in the backend and use ontologies to structure the metadata used for navigating the data. Our solution is different as we rely only on semantic store as a data backend.

In [9], Vdovjak *et al.* present Hera, a methodology which supports the design and engineering of Semantic Web Information Systems. They decompose an application into three layers, the semantic layer, the application layer and the presentation layer which correspond to the model–view–controller design pattern we use in SWORD. Hera focuses mainly on presentation and the interaction with users is mainly through navigation. In our solution, users can navigate the semantic web as well as adding information, editing information and deleting.

Corcho *et al.* introduce a Semantic Web portal using the MVC design pattern [4], but, similarly to Hera and SHDM, do not integrate it with an existing framework. We extend an existing popular framework (Ruby on Rails), thus connecting to an existing Web development community, leveraging the existing ecospace of plugins and extensions, and reducing the adoption barrier. Using Ruby on Rails and its ecospace leverages its standard functionality such as authentication, Web Services, AJAX support, Web 2.0, templating etc.

3 Semantic Web on Rails Development

We add the following functionality to the Ruby on Rails framework:

Prototyping: we provide a set of generators, inspired by the Ruby on Rails “scaffolding” [8], which generates models, views and controllers based on a database schema. In our case, we generate the model, views and controller based on a given RDF(S) ontology.

Interaction: the generated MVC handles showing, fetching, editing, searching, and versioning of Semantic Web resources.

⁴ <http://wiki.activerdf.org/SWORD/>

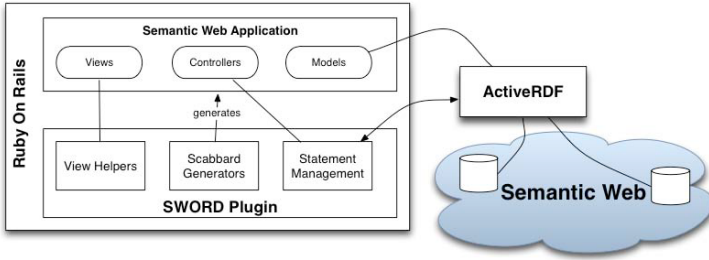


Fig. 1. SWORD architecture

Versioning and provenance of data: we track the versioning history and provenance of each statement, allowing human readers to include or exclude certain sources from their application views.

Semantic Web resource management: we provide libraries to manipulate Semantic Web resources and statements.

Figure 1 shows the relationships between SWORD and different components. SWORD integrates with Ruby on Rails as a plugin, and uses the ActiveRDF⁵ Ruby library⁶ to maps RDF(S) resources to Ruby objects.

4 The FOAF Browser Example

We demonstrate the functionalities of SWORD by prototyping a social networking application. This application uses the FOAF⁶(friend of a friend) ontology which defines a vocabulary for representing people and their relationships. The application provides views to search, show, browse and edit personal profiles, while maintaining the history and provenance of information. The application integrates data from various, arbitrary, sources using on-demand data collection, and enables interoperability with other Web applications through the FOAF ontology and other RDF vocabularies.

We generate and run the web application as follows:

```
rails social_networking ; cd social_networking
./script/generate scabbard person foaf http://xmlns.com/foaf/0.1/
./script/server
```

Fig. 2 shows the generated files, namely a controller for `people`, some helpers, a `person` model, a shared `people` layout and several views for displaying the various actions (the underscored files are partial views used in AJAX actions).

In Figure 3, the generated application is displayed, showing the profile of Cédric. This personal profile is actually an RDF file on his web page which was automatically fetched by the application and integrated into the knowledge base.

⁵ <http://www.activerdf.org/>

⁶ <http://foaf-project.org/>

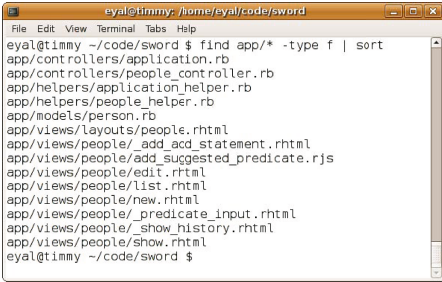


Fig. 2. Generated scaffolding files

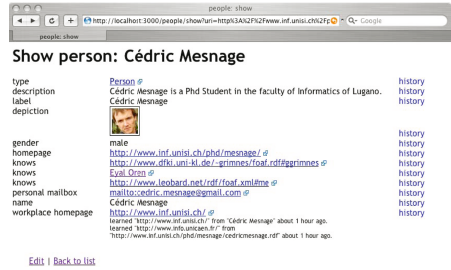


Fig. 3. Showing Cédric’s FOAF profile

When showing this person, all known statements about him are displayed, using human-readable labels, defined in an external ontology which is also integrated on-demand.

As shown at the bottom of the screenshot, the user can display the history and provenance (source) of any piece of information; in this particular example, the history of the “workplace homepage” is shown.

The generated application allows users to browse the social network by following any relationship between people, such as the “foaf:knows” relationship between acquaintances. When clicking on such a relationship, the information about that person is automatically fetched from the Semantic Web, and the display is updated to show the particular information of that person. In this example, users browse the social network; in general, the application allows users to browse the Semantic Web.

Fig. 4 displays the automatically generated overview page, showing a list of people. From this page users can search for all people (resources) related to a certain term, and for each person the standard CRUD (create, read, update, delete) actions are available, through the user interface actions: “show”, “edit”, and “delete”.

Editing as shown in Figure 5 is provided by a view which contains a form dynamically created according to the available information on the resource to be edited. Once submitted the update action of the controller is called. It adds

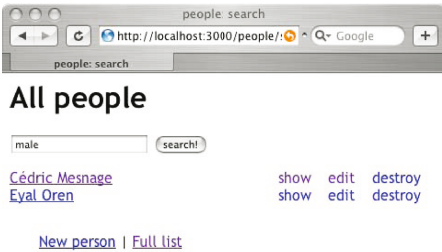


Fig. 4. Searching

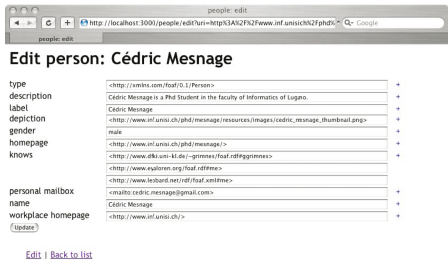


Fig. 5. Editing Cédric’s foaf profile

new statements if new information is entered (using the + button in the form) or edition statements if some existing statements have been edited.

All this behaviour is generated automatically, and can be customised by the application designer to adapt the generic behaviour to his needs, by changing actions or by adding new actions and new views.

5 Conclusion

In this paper, we presented SWORD, a plugin for Ruby On Rails which together with the ActiveRDF Ruby library transforms the popular Web application framework in a Semantic Web application framework. We demonstrated the use of SWORD to create an interoperable social networking application based on the FOAF ontology.

References

1. Berners-Lee, T.: Weaving the Web – The Past, Present and Future of the World Wide Web by its Inventor. Texere (2000)
2. Berners-Lee, T., Hall, W., Hendler, J.A.: A Framework for Web Science (Foundations and Trends(R) in Web Science). Now Publishers Inc., (2006)
3. Burbeck, S.: Applications Programming in Smalltalk-80: How to use Model–View–Controller (1987)
4. Corcho, O., López-Cima, A., Gómez-Pérez, A.: A platform for the development of semantic web portals. In: ICWE '06. Proceedings of the 6th international conference on Web engineering, pp. 145–152. ACM Press, New York (2006)
5. Lima, F., Schwabe, D.: Application modeling for the semantic web. In: Web Congress, 2003. Proceedings. First Latin American, pp. 93– 102 (2003)
6. Oren, E., Delbru, R., Gerke, S., Haller, A., Decker, S.: ActiveRDF: Object-oriented semantic web programming. In: Proceedings of the International World-Wide Web Conference (May 2007)
7. Reenskaug, T.: Models, views, controllers. Tech. rep., Xerox PARC (1979)
8. Thomas, D., Hansson, D., Breedt, L., Clark, M., Davidson, J.D., Gehrtland, J., Schwarz, A.: Agile Web Development with Rails. Pragmatic Bookshelf (2006)
9. Vdovjak, R., Frasincar, F., Houben, G., Barna, P.: Engineering semantic web information systems in hera. Journal of Web Engineering, pp. 3–26 (2003)

Personalized Faceted Navigation in the Semantic Web*

Michal Tvarožek and Mária Bieliková

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
{Name.Surname}@fiit.stuba.sk

Abstract. This paper presents the prototype of an adaptive faceted semantic browser – Factic. Factic implements our novel method of navigation in open information spaces represented by ontologies based on an enhanced faceted browser with support for dynamic facet generation and adaptation based on user characteristics. It is developed as part of a modular framework that supports personalization based on an automatically acquired ontological user model. We describe software tool design and implementation together with discussion on several problems mostly related to the general immaturity of current Semantic Web solutions.

1 Concept Overview

The Semantic Web as envisioned by Tim Berners-Lee aims to solve some problems of the current Web related to its constant change and growth by incorporating shared semantics i.e. “meaning” thus e.g. significantly improving interoperability between systems [1]. Although this idea was proposed several years ago, it still remains largely unrealized due to various reasons such as the lack of standards or appropriate software tools. As the need for shared semantics and (web) data integration grows, the demand for common conceptualizations referred to as ontologies is also increasing. Some authors argue that the use of ontologies in the e-science community presages ultimate success for the Semantic Web [1].

Consequently, proper software tools for navigation in the Semantic Web i.e. for navigation in ontologies (e.g., RDF/RDFS, OWL) are required. While these will include new types of tools, adding support for ontologies to “classical” tools is also imperative as these are already widely used in different scenarios. Examples of existing tools include search engines, web portals or faceted browsers [2].

Furthermore, the size and changeability of the Web and consequently the Semantic Web together with their diverse user base make them prime candidates for adaptive web-based systems that take advantage of (automatic) user adaptation in order to increase overall effectiveness, productivity and user orientation.

* This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104 and the State programme of research and development under the contract No. 1025/04.

The concept of the adaptive faceted semantic browser was proposed in [3]. As such, it is an enhanced faceted browser with support for:

- Ontological representation of the application domain by means of a domain ontology (e.g., in OWL).
- Logging of user actions with semantics within the browser as defined by an event ontology. The created user action logs are subsequently used for automatic user modeling, which is performed by external user modeling tools [4].
- Personalization based on an ontological user model derived from the domain ontology and created and maintained by the aforementioned user modeling tools. The personalization includes the adaptation of facets, facet restrictions and search results as well as the recommendation of relevant concepts.

Based on its properties, the adaptive faceted semantic browser is suited for effective viewing and navigating in large open information spaces represented by an OWL ontology. It can also be used as an information retrieval tool where the search query is visually created by means of navigation – selecting restrictions in the set of available facets, which are dynamically adapted to users’ needs.

2 Faceted Browser Design and Implementation

We designed and implemented the adaptive faceted semantic browser in the form of a software tool called *Factic*. It is a presentation tool that allows users to navigate and search in an information space represented by an OWL ontology. Therefore, we integrated *Factic* into the personalized presentation layer [5] of a web-based information portal [6] (see Figure 1).

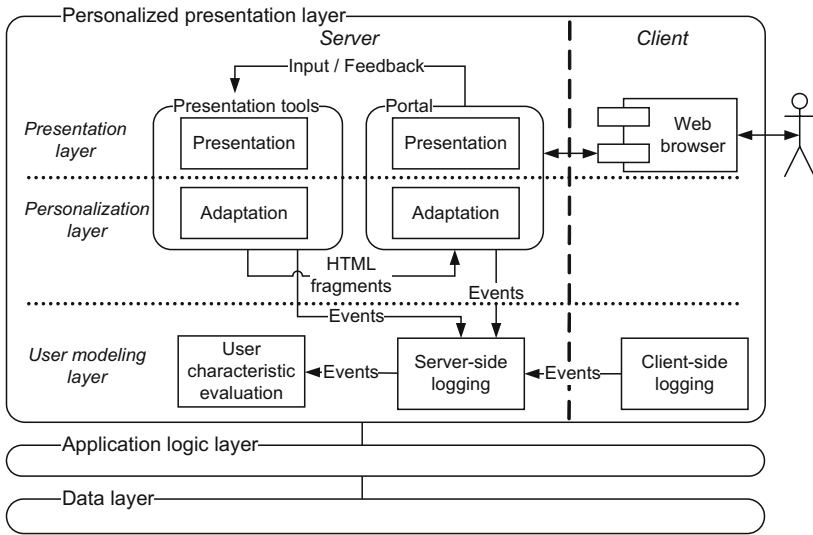


Fig. 1. Architecture of the personalized presentation layer of a web-based portal

The *Factic* presentation tool is depicted on the top left and can be divided into two parts – the presentation and the adaptation part. As input, *Factic* takes user input/feedback from the Portal module, to which it also sends the results of its processing in the form of (X)HTML fragments. The portal serves for the integration of individual presentation tools (e.g., *Factic*) and acts as a proxy towards the client web browser depicted on the right.

Presentation tools as well as the Portal tool perform user action logging with semantics by means of the user modeling layer depicted at the bottom, which performs both client-side and server-side logging and user characteristic analysis.

Factic is implemented in Java as an *Apache Cocoon* coplet (i.e. Cocoon portlet) and uses *Sesame* to store ontological data, *MySQL* to store relational data and as a back-end for *Sesame*. For the logging service we use *Apache Axis* as a web service container and *Apache Tomcat* as a servlet container.

Cocoon is based on XML and the pipes and filters architectural pattern where every request is processed by a given pipeline. Each pipeline consists of a single generator, zero or more transformers and a serializer. *Factic* itself as a *Cocoon* generator takes full advantage of its XML processing capabilities.

Figure 2 depicts the design and request processing of the *Factic* tool, which employs a two-step transform view, where the initial logical XML output description is transformed by a set of XSL transformations into the final XHTML document (top) and sent to the client web browser (right). Individual user requests are handled as described by the Sequence 1.

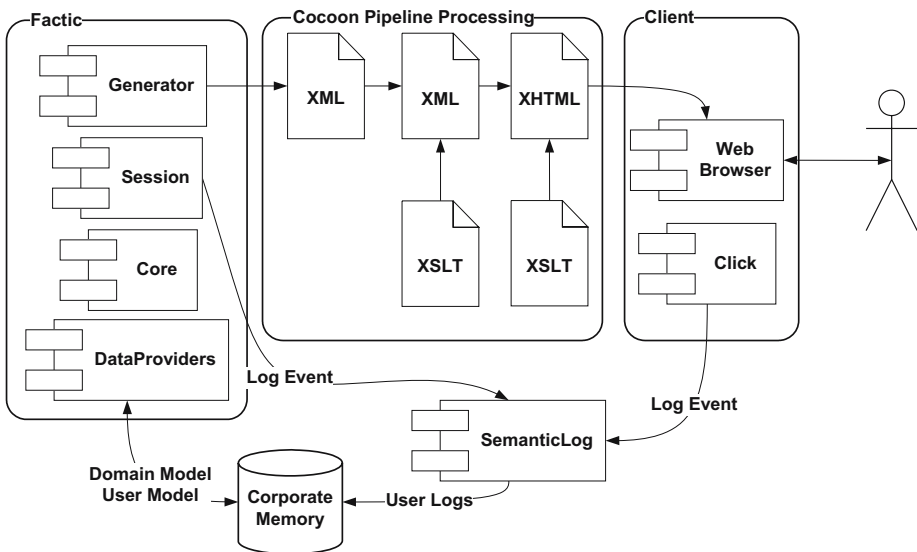


Fig. 2. Design of the Factic presentation tool

Sequence 1. HandleRequest *Input:* URL request *Output:* XHTML response

- | | |
|--------------------------|---|
| 1. <i>Session:</i> | Preprocess request, update session state |
| 2. <i>Core:</i> | Process request, create and execute query |
| 3. <i>DataProviders:</i> | Retrieve domain and user data |
| 4. <i>Core:</i> | Process results, evaluate adaptation and annotation |
| 5. <i>Session:</i> | Log event via the SemanticLog logging service |
| 6. <i>Generator:</i> | Generate logical output description in XML |
| 7. <i>Cocoon:</i> | Transform XML output to formatted XHTML response |
-

3 Discussion and Conclusion

We evaluated *Factic* in the domain of online job offers¹ and in the domain of scientific publications². In both cases, we used *Factic* for the presentation of and navigation in the respective domain ontology as well as for information retrieval.

Figure 3 shows the experimental results that we achieved. The time and number of clicks represent the total user effort that was necessary to complete a given scenario i.e. to find a certain set of ontological instances. The results indicate that adaptive selection of active facets can significantly improve total processing time, which depends linearly on the number of displayed facets. Furthermore, recommendation of suitable ontological concepts based on the user model further reduces the number of necessary clicks as well as overall task completion time.

While the results that we achieved are promising we also encountered several problems in the form of several performance bottlenecks:

- The logging of user actions together with the display state of the GUI via web services was very slow because a lot of data had to be serialized/deserialized via SOAP. We solved this by using a hybrid logging approach where some of the data are logged via web services (e.g., client-side logging) and some data are logged directly by means of an API (e.g., display state of the GUI).
- The cost of ontological queries is high and consequently, the processing of ontological queries is slow. We were unable to resolve this problem although we improved overall performance by caching data in *Factic*. Furthermore, the ontological repository *Sesame* is rather immature – it is slow, unoptimized and contains several bugs, which prevent correct evaluation of queries.
- SeRQL – the recommended query language for *Sesame* and thus *Sesame* lack several important features such as COUNT() or ORDER BY. These must thus be emulated by our application which further reduces performance.

The primary advantage of our approach lies in the use of ontologies. The shared conceptualization provided by ontologies improves tool interoperability. E.g., our automatic user modeling is not hard-coded to the output of specific presentation tools nor does it require extensive preprocessing as seen in traditional analysis of web server logs. Furthermore, the stored semantics of user actions are directly used in the user modeling process to estimate user characteristics.

¹ NAZOU Project, <http://nazou.fiit.stuba.sk>

² MAPEKUS Project, <http://mapekus.fiit.stuba.sk>

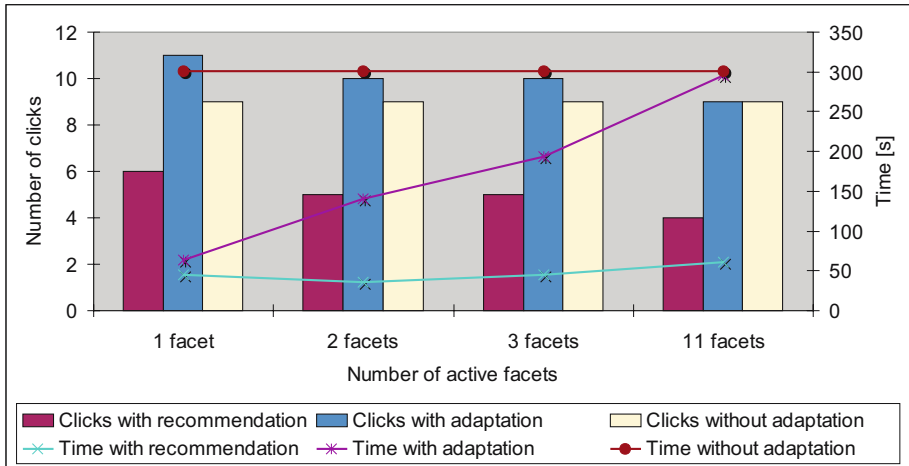


Fig. 3. Evaluation results for different adaptation modes (non-adaptive, with adaptation, with recommendation)

The acquired user characteristics are used in the adaptation process to improve efficiency and overall user experience without the need for direct conscious user involvement. As a result, the user can focus on the tasks at hand without the need to perform tedious system and/or user model settings.

Lastly, the *Cocoon* presentation framework allows us to easily integrate additional presentation tools as well as to further customize the processing pipeline by using additional transformers. Thus future work will include the integration with additional presentation/navigation/information retrieval tools and the implementation of new adaptation functions, such as dynamic facet generation.

References

1. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. *IEEE Intelligent Systems* 21(3), 96–101 (2006)
2. Instone, K.: How user interfaces represent and benefit from a faceted classification system. In: *SOASIST* (2004)
3. Tvarožek, M.: Personalized Navigation in the Semantic Web. In: Wade, V., Ashman, H., Smyth, B. (eds.) *AH 2006*. LNCS, vol. 4018, pp. 467–471. Springer, Heidelberg (2006)
4. Andrejko, A., Barla, M., Bieliková, M.: Ontology-based User modeling for Web-based Information Systems. In: *ISD 2006*, Budapest, Springer, Heidelberg (2006)
5. Tvarožek, M., Barla, M., Bieliková, M.: Personalized Presentation in Web-Based Information Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) *SOFSEM 2007*. LNCS, vol. 4362, pp. 796–807. Springer, Heidelberg (2007)
6. Barla, M., Bartalos, P., Sivák, P., Szobi, K., Tvarožek, M., Filkorn, R.: Ontology as an Information Base for Domain Oriented Portal Solutions. In: *ISD 2006*, Budapest, Springer, Heidelberg (2006)

WebVAT: Web Page Visualization and Analysis Tool

Yevgen Borodin, Jalal Mahmud, Asad Ahmed, and I.V. Ramakrishnan

Dept. of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{borodin, jmahmud, asada, ram}@cs.sunysb.edu

Abstract. WebVAT is an open-source platform-independent visualization tool designed to facilitate Web page analysis. The tool, built on top of the Mozilla Web browser, exposes Mozilla's internal representation of Web pages, *Frame Tree*, reflecting HTML rendering information. Compared to HTML DOM analyzers, WebVAT provides access to a cleaner, fuller, and more accurate data structure, which contains layout information, reflecting changes made by CSS and some types of dynamic content. WebVAT provides a framework for experiments and evaluations of algorithms over the Frame Tree. WebVAT also captures user interaction with the browser and can be used for data collection. WebVAT is a working tool actively used in the HearSay [10] project. This paper describes the architecture, design, and some of the applications of WebVAT.

1 Introduction

The expansion of the Web created a large venue for research. A big niche is taken by Web content analysis, including Web page segmentation, classification, summarization, etc. Like any other research area, Web content analysis requires tools to help with experimentation and evaluation. A number of existing tools allow viewing and editing Web pages, exploring their structure, etc.

Web page analysis often involves examining the internal structure of Web pages, usually represented by HTML DOM trees [2]. DOM trees are widely used for Web information extraction [1]. A number of software tools enable DOM inspection [9,3]. However, a DOM tree does not specify how to render a Web page, leaving the implementation to Web browsers. HTML DOM trees do not capture the layout information, unless it is explicitly specified in the HTML code. They also do not reflect changes made by JavaScript or Cascading Style Sheets (CSS), limiting the information available to Web engineers and researchers.

At the same time, Web browsers (e.g. FireFox, Internet Explorer, etc.), which are perfect for rendering Web pages, do not easily expose the layout of Web pages. Mozilla extensions, such as Web Developer [12] and Firebug [4], can access and visualize layout information of FireFox. However, because FireFox extensions are written in XUL and JavaScript, they cannot directly use libraries written in other languages, e.g. Java. Therefore, using FireFox extensions to test various

algorithms is not feasible. Some Web page segmentation algorithms use Web browser API's to obtain page layout (e.g. VIPS [13]), but we are unaware of any open-source tools that make use of visual layout of Web pages.

In this paper we describe WebVAT, a tool developed specifically for Web page visualization and analysis in the framework of the HearSay [10] project. WebVAT, based on the Mozilla Web browser, provides visualization capabilities and a flexible infrastructure for Web page analysis. WebVAT enables users to analyze the structure and layout of Web pages as rendered by the browser. We are actively using WebVAT for visualization and evaluation of our algorithms, data collection, capturing user interactions - all contributing to the development of the state-of-the-art non-visual Web browser, HearSay. We next present the architecture and design of WebVAT in Sections 2 and 3 respectively, followed by some of the applications of the tool in Section 4. We close the paper with concluding remarks, future work, and acknowledgements in Section 5.

2 WebVAT Architecture

WebVAT, written in Java, is built on top of Mozilla, an open-source cross-platform Web browser. Thus, WebVAT works on a variety of platforms including Windows, Linux, and OsX. The architecture of WebVAT is shown in Figure 1.

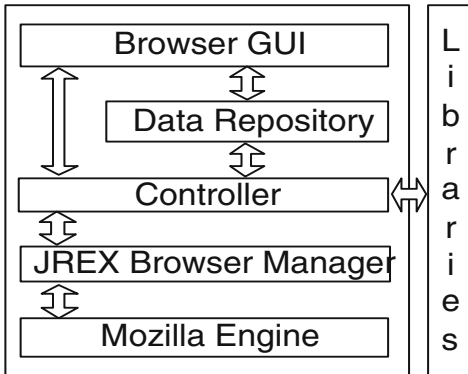


Fig. 1. WebVAT Architecture

Users interact with WebVAT through an event-driven graphical user interface provided by the Mozilla Web browser. Any user interaction with the Browser GUI is captured and can be processed, modified, and recorded by the WebVAT Controller module. The Controller also interfaces with a number of Libraries, containing various Web content analysis modules. The Browser GUI and the Controller share a common Data Repository. Besides using the already implemented functionalities of WebVAT, advanced users can easily extend the Controller and Browser GUI, and test their own algorithms.

WebVAT Controller interacts with Mozilla engine through the JREX Browser Manager [5]. JREX provides Java interface to the Mozilla engine, allowing to call the engine's APIs, define custom event handlers, etc. Mozilla engine supports standard browser functionalities, such as support of cookies, secure connection, etc. We have extended the Mozilla engine and JREX to expose and extract a *Frame Tree*, Mozilla's internal representation of a Web page, after the Web page has been rendered by the browser. This way, Mozilla takes care of any

dynamic content, cascading style-sheets, malformed HTML, and other rendering problems. This relieves users from having to deal with heavy DOM-tree objects, while giving them fuller and more accurate information about the style and layout of Web pages. While we are using Mozilla for rendering HTML, other browsers will produce similar data structures.

When the user enters an address or navigates a link, the Controller extracts the Frame Tree of the Web page from the Mozilla engine. Figure 2 (b) shows a frame tree corresponding to the Amazon.com Web page. A *Frame Tree* is a tree-like data structure that contains Web page content, along with its 2-D coordinates and formatting information, that specifies how the Web page has to be rendered on the screen. Frame coordinates refer to the *upper-left corners* of the corresponding Web page segments displayed in the browser, independent of the screen resolution or the size of the browser window.

A frame tree is composed of nested *frames*¹, so that the entire page is a root frame, containing other nested frames down to the smallest individual objects on the page. The browser window in Figure 2 (a) shows the Amazon.com Web page with some of the frames highlighted. The corresponding frame tree nodes are selected in windows (b) and (c). The frame-trees are partially expanded to demonstrate the types of frames. We distinguish between the following classes of frames: text, links, images, image-links, form-elements, XHTML, and non-leaf frames. We next describe the design of WebVAT.

3 WebVAT Design

WebVAT is designed around the Mozilla Web browser interface, which displays the browser window with a standard menu extended with *Tools*, *Trees*, and *Highlight* (see Figure 2).

The *Tree* menu contains the list of all frame-tree windows that can be displayed on the screen. Different frame-tree windows can be used to visualize the results of experimental algorithms. For example, Figure 2 (b) shows the original frame tree produced by the Mozilla engine, while window (c) shows a frame tree that was processed and segmented into blocks (3-D icons) by our geometrical clustering algorithm [7].

With minimal code changes, WebVAT can support any reasonable number of frame-tree windows, synchronized by the observer handler (part of Browser GUI module). Selecting any node in any tree also selects the corresponding nodes in all other active frame-tree windows, and highlights the corresponding frame in the browser window, as can be seen in Figure 2 (a), (b), and (c). The *Highlight* menu items give additional control over highlighting functionality by allowing to clear highlighting, use different colors to highlight frames, etc. The experimental algorithms executed by WebVAT can activate frame-tree windows and highlight frames to visualize the results.

The *Tools* menu contains a growing number of useful tools. Among them there are: search, Figure 2 (e), which allows to find frames by the contained text, or

¹ Note, this is different from HTML frames.

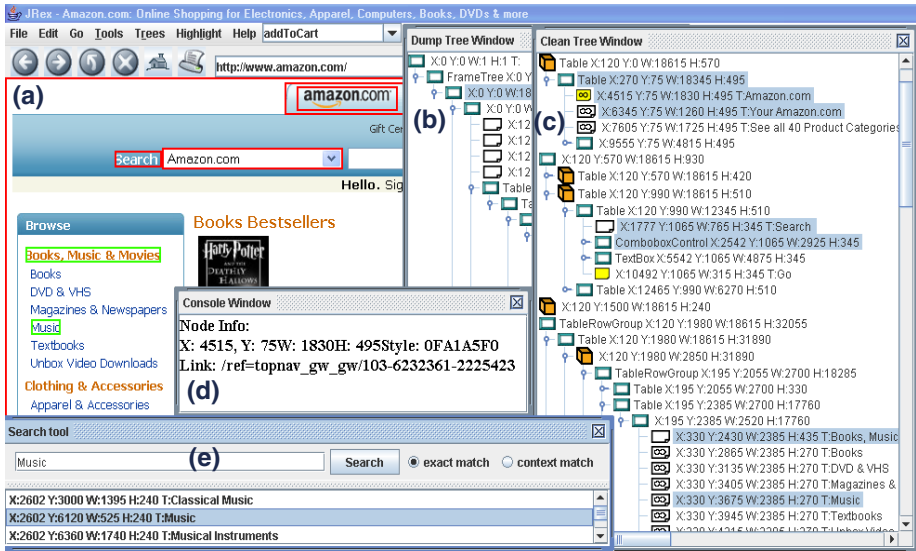


Fig. 2. WebVAT in action

can use other experimental search algorithms; console window that can be used for any output, Figure 2 (d); evaluation tool to record questionnaire answers, user-evaluation results, etc.; and, a data collection tool that allows to save HTML pages and the corresponding XML frame trees with selected frames. The data collection tool can also save sequence of pages, recording followed links, and the action labels (e.g.: *addToCart*), which can be selected from the combo box in the menu panel of the main window in Figure 2 (a).

4 WebVAT Application: The HearSay Experience

WebVAT can be used in a number of applications. Here we describe its role in the research and development of the HearSay non-visual Web browser [10].

WebVAT was used to verify the correctness of the Frame Trees while we were modifying the Mozilla engine code. The HearSay browser is also based on Mozilla; HearSay uses a number of algorithms and techniques to clean the frame trees, analyze their content, and convert them into audible dialogs. WebVAT helped verify all of the algorithms used in HearSay.

The frame-tree window in Figure 2 (c) displays the results of our Web page segmentation algorithm [7], which identifies geometrically aligned blocks as semantic clusters of information (marked as 3-D blocks). We are working on expanding our partitioning algorithm to find repeating patterns within blocks [8].

We also used WebVAT as a *data collection* tool. The participants were asked to identify and select some links and the information pertaining to the same topic around them in a number of Web pages. They were also asked to identify the

information relevant to the links on the pages, to which the links were pointing. The data was, then, used to test our context collection algorithm. The same data was used in training an SVM-based statistical model to identify relevant information in Web pages while following links from one page to another [6,7]. WebVAT helped us visualize and evaluate the results of the algorithms. We are now using WebVAT to construct a process model for Web transactions [11].

5 Conclusion and Future Work

In this paper we described the architecture, design, and applications of our Web content visualization and analysis tool, WebVAT, which will be soon publicly released with the HearSay Web browser (www.cs.sunysb.edu/~hearsay). We identify several directions to further enhance this tool.

We used WebVAT to collect the data for off-line training of statistical models. It may be possible to integrate different machine learning modules with WebVAT to train statistical models online, while using the tool. For example, we plan to use WebVAT to collect data to train Bayesian models for transactional concept detection, such as taxonomy, search results, etc. WebVAT can also be enhanced to create ontologies – the knowledge underlying the semantic Web. Users will be able to highlight sections of Web pages, or frame tree nodes, and specify the corresponding concept name. This will help learning ontologies from examples.

Acknowledgements. This research is supported by NSF Award IIS-0534419.

References

1. Chakrabarti, S.: Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In: WWW'01 (2001)
2. <http://www.w3.org/DOM/DOMTR>
3. <http://www.dubbeldam.com/DOMSpy.html>
4. <http://www.getfirebug.com/>
5. <http://jrex.mozdev.org/>
6. Mahmud, J., Borodin, Y., Das, D., Ramakrishnan, I.: Combating information overload in non-visual web access using context. In: UII, Short paper (2007)
7. Mahmud, J., Borodin, Y., Ramakrishnan, I.: Csurf: A context-driven non-visual web-browser. In: Proceedings of WWW (to Appear)
8. Mukherjee, S., Yang, G., Ramakrishnan, I.: Automatic annotation of content-rich html documents: Structural and semantic analysis. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, Springer, Heidelberg (2003)
9. <http://www.sharewareconnection.com/pagespy.htm>
10. Ramakrishnan, I., Stent, A., Yang, G.: Hearsay: Enabling audio browsing on hypertext content. In: WWW (2004)
11. Sun, Z., Mahmud, J., Mukherjee, S., Ramakrishnan, I.V.: Model-directed web transactions under constrained modalities. In: WWW '06. Proceedings of the 15th international conference on World Wide Web, pp. 447–456 (2006)
12. <http://chrispederick.com/work/webdeveloper/>
13. Yu, S., Cai, D., Wen, J.-R., Ma, W.-Y.: Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In: WWW (2003)

Smart Tools to Support Meta-design Paradigm for Developing Web Based Business Applications

Athula Ginige, Xufeng Liang, Makis Marmaridis, Anupama Ginige,
and Buddhima De Silva

University of Western Sydney, Locked Bag 1797, Penrith South DC, 1719, NSW, Australia
a.ginige@uws.edu.au, bdesilva@scm.uws.edu.au

Abstract. Many Small and Medium Enterprises (SMEs) tend to gradually adopt Web based business applications to enhance their business processes. To support this gradual adoption we need a framework that supports iterative development. Further processes that have been supported by web based business applications can change and evolve requiring applications to be changed accordingly. To support these needs we have extended the **Component Based E Application Development and Deployment Shell; CBEADS[®]**. We analysed many business applications and derived a meta-model. We implemented this meta-model with in CBEADS[®] and developed a set of Smart Tools to take the instance values of the meta-model and generate the web based business applications. When a new business application is required, a business analyst can create a new instance of the meta-model. To change an implemented business application the appropriate values of the meta-model instance that corresponds to the particular application can be changed.

Keywords: Meta-Design, Web Engineering, Smart Tools, rapid development, model driven software development.

1 Introduction

The AeIMS research group at the University of Western Sydney has been working with Small to Medium Enterprises (SMEs) in the Western Sydney region to investigate how Web based business applications can be used to enhance their business processes to become competitive in a global economy [1, 2]. In this research one challenge was to find a way to develop web based business applications rapidly and in a cost-effective manner [3]. Also it was necessary to have the ability to change these applications with evolving business needs [4]. The development approach should also reduce the gap between what the users actually wanted and what is being implemented in terms of functionality [5].

In a Business organisation there are many software applications to support its business processes such as ordering, inventory management, leave processing, invoicing, production planning, customer relationship management (CRM) etc. Today most of these applications tend to be web based. Some of the data used in these processes such as employee details, customer details, product or service details etc will be common to many business processes. Thus organisations that have invested in separate systems to

support their business applications are finding it hard to keep the common data stored in separate systems in sync. Unlike large organisations, SMEs have not got the money or the time to invest in obtaining a single ERP system to support its business processes as a one-off project. We have observed often in SMEs one process at a time being enhanced by the use of web technologies based on some priority criteria. Also these web applications used to enhance the business processes need to evolve when business processes change with time.

To meet the above requirements we adopted the concept that software is a medium to capture knowledge rather than a product [6]. This led us to change our thinking from looking at methodologies to develop web based business applications to developing a framework within which web based business applications can evolve. We adapted the **Component Based E-Application Development and Deployment Shell (CBEADS[®])** as our framework [3, 7]. CBEADS[®] has the ability to create new functions within its own framework so that it can evolve. These functions can be grouped together to form various business applications.

This resulted in a hierarchical model; an organisation can have a shell, the shell will have many applications, applications consist of many functions (use cases). We analysed many business applications and found that most of these can be modeled as a form being routed based on some rules and different actors can have access to different views of the underlying data objects. Thus rather than developing “the application” we developed a meta-model of the application and a set of tools within which the Business Analyst can create the applications that they want [8].

2 Meta-model for Web Based Business Applications

Our meta-model consists of three levels of hierarchical abstraction called Shell, Application and Function as shown in figure 1. The Shell provides the common

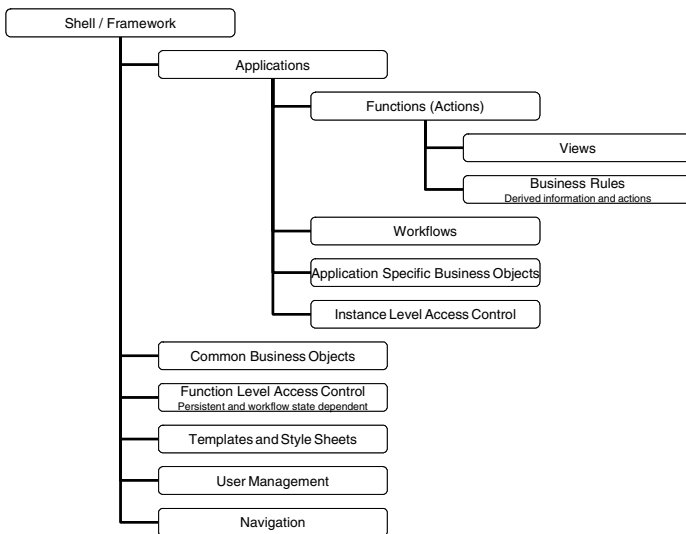


Fig. 1. Hierarchy of Abstraction Levels in Business Application Meta-Model

functionality required for any application such as user management, authentication, access control and overall navigation. The Application provides a set of functions required to perform a business process and sequencing of functions as required to create the necessary workflows. Functions provide views or user interfaces required to perform actions in a business process.

To implement this meta-model within CBEADS[®] framework we had to first develop a way to specify Business Objects, Views, Business Rules, Function level and Instance level access control, Workflows, Over all layout and, Look and feel. Then we had to create a way to generate the physical objects (i.e. databases), functions and workflows required for the application based on instance values of the meta-model. For this we developed a set of Smart Tools. We incorporated some computer domain knowledge into these tools so that a Business Analyst can use these tools to develop parts of the application without needing to develop detail code as well as databases.

3 Smart Tools

We developed an architecture for Smart Business Objects and two tools; Builder and UI Generator to create the objects and to generate the required views incorporating various business rules [9]. We developed a state machine based approach to model business processes that can evolve with changing business needs and a workflow engine to enact these business process models [10]. We identified that navigation and access control are tightly coupled and implemented these two together.

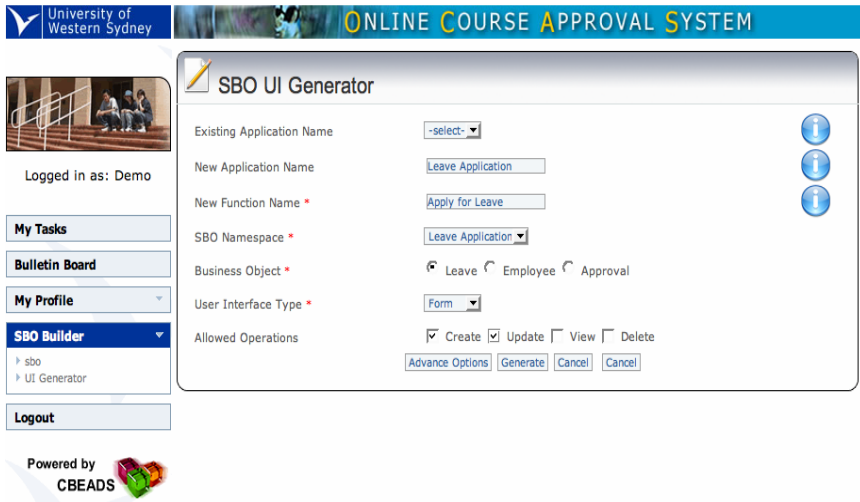


Fig. 2. Generating UIs for leave processing application

3.1 Smart Business Objects (SBO)

Using the SBO builder tool in CBEADS[®] we can specify business objects using the SBOML language [9]. Example of a business object used in a leave processing

application is shown below. The SBO builder then generates the Business Objects for the application.

SBOML Specification of the Leave Object
In leavesystem, leave has date, applicant, from (date), to (date), type (which could be sick or annual or no pay), status (which could be approve or reject), many approval (has date, approved by, comment)

UI Generator tool allows us to select the business object and necessary attributes to create different views required for the leave processing application. This tool is shown in figure 2.

3.2 Workflow Engine

Next we need to sequence the views. This we do using a state table. The workflow specification for the leave processing application is shown in Table 1.

Table 1. State Table for Leave processing workflow

Current State	Actor	Function	Buttons	Do Action	Next State
1	Employee	Apply_Leave	Submit	Email Division_Manager	2
2	Division_Manager	Approve_Leave	Approve	Email HR_Manager, Employee	3
			Reject	Email Employee	4
3	HR_Manager	Process_Leave	Processed	Email Employee, Division_Manager	4
4	HR_Manager	View_Applications	-		4

3.3 Navigation and Access Control

We have identified the need to provide 2 types of access to various functions; functions that can be accessed any time (workflow state independent access) and functions that can be accessed based on workflow state such as a manager getting a link to access approve leave function only when there is a pending leave application. We also need to manage instance level access control.

We used role based access control mechanism in CBEADS[®] to provide state independent navigation to functions at shell level. We developed an access control mechanism based on setting and revoking capabilities to provide access to workflow state dependent functions. We implemented the notion of a “Project Team” that owns an object instance to achieve instance level access control.

4 Conclusion

We have developed a set of Smart Tools to rapidly develop Web based Business Applications based on meta-design paradigm. This approach is well suited to develop

web based business applications for SMEs as it supports iterative development. We are now in the process of developing another set of tools to empower end users to develop their own business applications.

References

1. Ginige, A.: Collaborating to Win - Creating an Effective Virtual Organisation. In: International Workshop on Business and Information, Taipei, Taiwan: Shih Chien University and National Taipei University (2004)
2. Ginige, A.: From eTransformation to eCollaboration: Issues and Solutions. In: 2nd International Conference on Information Management and Business (IMB 2006) Sydney, Australia (2006)
3. Ginige, A.: Re Engineering Software Development Process for eBusiness Application Development. In: Fifteenth International Conference on Software Engineering and Knowledge Engineering. San Francisco Bay, USA (2003)
4. Ginige, A.: New Paradigm for Developing Software for E-Business. In: IEEE Symposia on Human-Centric Computing Languages and Environments, Stresa, Italy, IEEE, New York (2001)
5. Epper, M.: Poor Project Management Number-One Problem of Outsourced E-Projects, in Research Briefs, Cutter Consortium (2000)
6. Armour, P.G.: The Case for a New Business Model: Is software a product or a medium. In: Communication of the ACM (2000)
7. Ginige, A.: New Paradigm for Developing Evolutionary Software to Support E-Business. In: Chang, S.K. (ed.) Handbook of Software Engineering and Knowledge Engineering, pp. 711–725. World Scientific, Singapore (2002)
8. Ginige, A., De Silva, B.: CBEADS©: A framework to support Meta-Design Paradigm. In: (HCII 2007). 12th International Conference on Human-Computer Interaction, Beijing, P.R. China. LNCS, vol. 4554, pp. 107–116, Springer, Heidelberg (2007)
9. Liang, X., Ginige, A.: Smart Business Objects: A new Approach to Model Business Objects for Web Applications. In: 1st International Conference on Software and Data Technologies. Setubal, Portugal (2006)
10. Ginige, J.A., Uma, S., Ginige, A.: A Mechanism for Efficient Management of Changes in BPEL based Business Processes: An Algebraic Methodology. In: ICEBE 2006. IEEE International Conference on e-Business Engineering, Shanghai China, IEEE, Los Alamitos (2006)

Next-Generation Tactical-Situation-Assessment Technology (TSAT): Chat

Emily W. Medina¹, Sunny Fugate², LorRaine Duffy³, Dennis Magsombol⁴,
Omar Amezcua⁵, Gary Rogers⁶, and Marion G. Ceruti⁷

Space and Naval Warfare Systems Center, San Diego (SSCSD),
¹ Code 2734, ² Code 2725, ³ Code 246207, ^{4,5,6} Code 246204, ⁷ Code 246206
53560 Hull Street, San Diego, CA 92152-5001
ewilson@spawar.navy.mil, fugate@spawar.navy.mil,
lorraine.duffy@navy.mil, dennis.magsombol@navy.mil,
omar.amezcua@navy.mil, gary.rogers@navy.mil,
marion.ceruti@navy.mil

Abstract. This paper presents concepts, content, status, applications and challenges of chat as used in the military context of secure net-centric command and control. It describes the importance of chat as it contributes to situation assessment and the common operating picture, which presents current collective knowledge of the battle space. The paper discusses future chat capabilities and outlines the road ahead for the TSAT project.

Keywords: Chat, collaborative applications, Common Operational Picture, network-content management, social-network applications.

1 Introduction

Chat is an important tool commonly used on the Web. Chat also is one of the many tools, including web services that enable net-centric military operations. Chat for net-centric communications in defense-related situation assessment is one of the primary activities that drive command and control (C²) or battle-space management. The Department of Defense uses mostly Internet-Relay Chat (IRC), which is an internet-based technology. The TSAT research-and-development project is focused on improving chat using new technology. Chat's value and maximum benefit can be realized only with the context in which it was developed.

In distributed C², the primary tool for consistency in operational situation assessment is the Common Operational Picture (COP) and the daily briefings and updates that define relevant objects, their relationships, their intent and the commander's response to the situation or threat. The COP is geographically oriented, distributed, shared space for maintaining and presenting the current status and collective knowledge of the battle space. The tactical war fighter usually provides the context for situation assessment, the maintenance of which depends on moment-to-moment, daily, and long-term updates of the variables of interest and the time evolution of these variables. The primary method for capturing this information resides in voice and written communications, with an increasing reliance on tactical-text chat [3], [9].

At the moment, opportunities to improve its policy or practice seem remote, given the current political and economic outlook.

In this same vein, transfer of understanding relies heavily on improper, unwieldy, and low-information content methods of communicating context: the written word in text chat; Defense Message System (DMS) communications; or embedded within typesetting and presentation software such as Microsoft Word and Powerpoint® documents. It is alarming that our daily tactical communication updates occur through the use of high-overhead, low information density, non-interoperable, or proprietary information formats such as archaic message formats, document typesetting software, or bullet-list presentation software. The simplest of these text formats generally require the isolation of simplistic and jargon heavy content and the complete disposal of relevant context [11]. Whereas these technologies have valid uses: the broadcast of official and formal announcements; the publication of documents; or the presentation of simple concepts to an audience, they will never be suitable medium for tactical communications.

The operational Navy has an urgent requirement for better organization of the large amount of electronic information. Unorganized and content-heavy information, like that found in United States Message Text Format (USMTF) messages, chat room logs, or websites, is altogether too much for one user to assimilate, compare, analyze and use to initiate an action in a timely manner. A capability needs to be developed to support information transfer in the form of written context in situation assessment. New technology provides better ways to transfer information rapidly than the inefficient text chat and DMS paragraph or text document formats obscured by inane typesetting information and features.

We can improve the content and methods of delivery of collaborative communications for command and control in a network-centric environment. Chat is an important part of the technical approach to improve the efficiency of knowledge sharing through communication of situation-assessment context. The importance of chat is evident considering that the content of chat messages drives real-time targeting and battle-space management [9]. This content can be enhanced through linguistic research and analysis, thus improving methods for determining chat context themes. Linguistic research also can help develop a novel approach to acquiring the situational assessment themes of chat content from their human writers by building a prototype “chatbot.”

2 Chat Status

Today’s chat is a technology for sharing context. Chat is the primary and often the only means of communicating situation updates on intermittent and discontinuous networks, superseding radio communications with text-only descriptions. For example,

(1) Strike-group Concept of Operations (CONOPS) establishes 500 to 800 chat rooms with between 2,000 and 4,000 users in each “channel” based on functional roles.

(2) Joint CONOPS establishes hundreds of chat rooms with joint-service participation based on mission objectives. It enables joint access to service-specific chat rooms to maintain non-intrusive situation assessment of service-specific activities [9].

Chat functions include support of the following activities:

- (1) Real-time targeting;
- (2) Edge users with limitations of low and/or intermittent bandwidth;
- (3) Immediate updates to COP context;
- (4) General information sharing of updates on a regular basis continuing for months to establish the management of operational tempo and battle rhythm;
- (5) Cross-domain operations.

An example has been observed of Mid-East dynamic target relocation based on chat interchange between a pilot and a ground-reconnaissance team. This is not an isolated or unusual case.

3 Chat Development Benefits, Issues, and Challenges

Fleet usage of chat is widespread for current operations but vastly inadequate for information retrieval and management. As important as it is, chat is too text driven, slow, and ambiguous. For these reasons, chat cannot provide the best accommodations to edge (tactical) users operating in extreme and hostile environments, such as:

- (1) Chemical-biological war fighters in protective suits called “MOPP gear;”
- (2) Special operations personnel in hostile situations;
- (3) Clean-up crews and search-and-rescue (SAR) personnel in oil spills, fires, earthquakes, and tsunamis;
- (4) Extreme-edge users on minesweepers and in submarines;
- (5) Other users in oxygen-deprived or toxic-atmospheric environments.

Table 1. Benefits and issues of TSAT chat technology development

Technology	Properties of each medium	Technology shortfalls
COP	<ul style="list-style-type: none"> • Comprehensive • Visually integrative 	<ul style="list-style-type: none"> • Unavailable in field • Poorly communicable
Text	<ul style="list-style-type: none"> • Persistent • Immediate • Easy to implement & use • Expressive • Prevalent operational acceptance 	<ul style="list-style-type: none"> • Poor field interface • No visual or symbol integration • Ambiguity issues • Slow communication rate • Not stealthy
Voice & Video	<ul style="list-style-type: none"> • Many solved problems • Immediate • Ostensive/direct • Intuitive interfaces • Mature technology • Video is gesture capable 	<ul style="list-style-type: none"> • Not persistent • Not stealthy • Symbol integration difficult • Textual integration difficult • High bandwidth required • Limited archival access

Observation and statistical analysis of trends in chat-room text constitute a subset of the larger overall issue of identifying and managing large collections of unstructured text. The Internet and email are prime examples of such collections. Chat entries hold vast amounts of information that easily can be misconstrued or forgotten

altogether [3]. Therefore, chat-room text itself must be scrutinized and the domain well defined before any further improvements on military chat commence. Issues and benefits that relate to secure net-centric chat are summarized in Table 1, showing how development of augmented chat capabilities can resolve outstanding technology shortfalls for COP, text, and through multi-modal tactical applications, voice and video.

Chat users need to integrate their information with other information sources at their disposal: geographic land-based terrain maps, the COP, and non-geographic computer-based “terrain” (e.g. global network operations and network topology). They also must be able to incorporate object relationships and time into a precise context that can be fused with other data to present an accurate COP and situation assessment, to serve as a basis for current command decisions, and for later study in operations analysis. Multiple criteria need to be established to institute rigor in chat-room text analysis. Metrics must be identified and measured against one another, and even compared with existing benchmark methods for chat analysis that are used in academia [1], [2], [4], [5], [6], [7], [10]. A few methods initially seem promising. For example, a user might be observed in a real-time environment to measure the amount of time taken to process a message and disseminate its contents to involved parties. Alternately, a user could be observed in attempts to locate topical information in a chat database. Another method might involve comparing the time a user takes to process a message with English text to the amount of time the user takes if visual symbols are substituted for text.

4 Chat Content

At the onset of Operations Iraqi Freedom and Enduring Freedom, the joint tactical community resorted to Internet Relay Chat (IRC) to transmit contextual information. This represents a late 1980s Internet capability. IRC has seen little or no improvement in the past two decades [3], [9]. With chat becoming more widespread in operational and tactical communities, the need is growing to understand the following.

- (1) The content that is communicated in chat sequences for situation updates;
- (2) How to categorize and parse the information more efficiently to various other repositories, such as databases; and
- (3) How to communicate quickly the nuances of evolving situations to distributed forces in an improved (and more automated) format.

Concomitantly, knowledge of context resides first in the warfighter’s mind. Today, this knowledge is communicated in gestures that are outside of computer networks and, therefore, are not stored in digital format. The present work includes the intent to acquire that contextual information using conversational query (in the guise of a “chatbot”) that constitutes an interesting and novel approach to information gathering. Traditional data-acquisition technologies have focused on data storage sites, whereas a key focus of the present work in TSAT is in the human network of incidental information, or “color commentary” that often is disregarded and rarely is preserved with other historical accounts unless one can recall and record a personal conversation with the information source. Whereas few metrics exist for chat-room

text analysis, to improve the current situation with chat, a statistical analysis of chat structure, topical organization, and user trends first must be performed.

5 Future Chat Capabilities

The goals of unstructured text chat research are twofold. First, linguistic analysis must discover statistically the topics that are most likely to reoccur in chat rooms. After these themes are identified, further study will focus on the best means of managing large databases of chat text. A few approaches have gained popularity when observing chat-room data, among them topic thread detection, and user-thread detection [1], [5], [6], [8], [10]. Topic thread detection would aid a user in identifying important entries, whereas user-thread detection would facilitate error detection and the identification of anomalous behavior. In the near term, metrics will be identified, tested, and evaluated. Thus, several near-term efforts need to be accomplished to advance chat beyond its current state, including short-term goals such as:

- (1) Select a chat domain as the focus the study.
- (2) Access secure-net-centric chat for content analysis.
- (3) Obtain a statistically significant data set.
- (4) Identify and test chat metrics.
- (5) Perform statistical analysis of chat-room topical content, user trends, and battle-field themes.
- (6) Construct a prototype for a chat-user database
- (7) Construct a prototype for topic-thread detection.

In ten years, chat will augment text with an Icon/Visual/Symbolic-based language. This language will be easy to understand, visual, international, and capable of depicting geographic and non-geographic, network-based battlefields. Future chat will use very low bandwidth (<300bps) and operate robustly in conditions of intermittent connectivity. The chat of the future will allow the user to portray complex concepts more efficiently than that of today. These complex concepts will be linguistically derived from then-current chat content to discipline language development. Tomorrow's chat will be keyboard independent because war fighters will use it for chemical and biological defense in extremely contaminated environments and in any other place where typing on a keyboard is not an option. For example, environmentally challenged astronauts will use it in space. Future rescue crews will use future chat in toxic, fire, or oxygen-deprived environments. Special Forces require stealth and immediate, persistent, line-of-sight independent communication tools, ease of use, and comprehensive situation updates from a COP in minimum time with minimum effort. This technology will provide key capabilities for special-operations personnel who must maintain a low probability of communication interception and detection in very noisy or stealth environments. The success of chat is tied to other technological advances. For example, future improvements to graphic user interface, textual representation, and method of use rely solely on an accurate portrayal of the military-chat domain.

6 The Road Ahead for the TSAT Project

Research goals include the following.

- (1) Determine metrics for ease of use, efficiency, and ambiguity resolution.
- (2) Answer the following key research question: “Does TSAT net-centric chat increase information capacity or communications efficiency?”
- (3) Linguistically analyze current chat content for linguistic “themes.”
- (4) Build a prototype candidate language.
- (5) Integrate linguistic themes with visual language prototype (new symbols).
- (6) Introduce new symbols and visual language elements into chat.
- (7) Perform “Visual Language” experiments testing human factors.
- (8) Achieve a technology transition in a long-range program in linguistic analysis technology for tactical communications.
- (9) Incorporate some Web-based technologies into chat.

7 Summary

The objective of the TSAT project is to improve the contextual information interface that accompanies traditional situation assessment. The program focus is to revolutionize the tactical text-chat interface including content, technique, application, network management, and hardware interface. Another important goal is to improve the transfer of contextual information to the COP and among distributed operational and tactical war fighters, regardless of the specific context. When completed, this research and development project will offer a significant improvement in the ability of war fighters at all echelons to share knowledge rapidly and accurately. Not only will it improve the communication of current contextual information, it will extend the range of capabilities both in terms of how information is exchanged.

Acknowledgements

The authors thank the Defense Threat Reduction Agency and the SSCSD Science and Technology Initiative for their support of this work. This paper is the work of U.S. Government employees performed in the course of employment and no copyright subsists therein. It is approved for public release with an unlimited distribution.

References

1. Bengel, J., Gaulch, S., Mittur, E., Vijayaraghavan, R.: ChatTrack: Chat room topic detection using classification (2004) <http://citeseer.ist.psu.edu/bengel04chattrack.html>
2. Elnahrawy, E.: Log-based chat room monitoring using text categorization: A comparative study. In: Proceedings of the International Association of Science and Technology for Development Conference on Information and Knowledge Sharing (IKS) (November 2002)

3. Eovito, B.: An assessment of Joint chat requirements from current usage patterns. Thesis, Naval Postgraduate School (NPS) (June 2006)
4. Hearst, M.: Multi-paragraph segmentation of expository text. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (June 1994)
5. Kolenda, T., Hansen, L.K., Larsen, J.: Signal detection using ICA: Application to chat room topic spotting (2001) <http://citeseer.ist.psu.edu/kolenda01signal.html>
6. Lin, C.Y.: Knowledge-based automatic topic identification. In: Proceedings of the Association for Computational Linguistics (ACL) (June 1995)
7. Salton, G., Singhal, A., Buckley, C., Mitra, M.: Automatic text decomposition using text segments and text themes. In: Proceedings of the 7th ACM Conference on Hypertext (March 1996)
8. Schmidt, A.P., Stone, T.K.M.: Detection of topic change in IRC chat logs. <http://www.trevorstone.org/school/ircsegmentation.pdf>
9. Simpson Jr, M.L.: A user's epistle on TextChat tool acquisition. In: Proc. of the Command and Control Research and Technology Symposium (CCRTS 2006) (June 2006)
10. Wayne, C.L.: Topic detection and tracking in English and Chinese. In: proc. of the 5th International Workshop on Information Retrieval with Asian Languages (IRAL) (September-October 2000)
11. Tufte, E.: Beautiful Evidence. Graphics Press (July 2006)

Tool Support for Model Checking of Web Application Designs*

Marco Brambilla¹, Jordi Cabot², and Nathalie Moreno³

¹ Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. Da Vinci, 32. I20133 Milano, Italy
mbrambil@elet.polimi.it

² Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya
Rbla. Poblenou 156. E08018 Barcelona, Spain
jcabot@uoc.edu

³ Departamento de Lenguajes y Sistemas Informáticos, Universidad de Málaga
Complejo Tecnológico, Campus de Teatinos. E29071 Málaga, Spain
vergara@lcc.uma.es

Abstract. In this work we report our experience in applying model checking techniques to the analysis of static and dynamic properties of Web application models. We propose a mix of tools that facilitate model driven design of Web applications, automatic code generation, and automatic property verification. As recommended by current tendencies in the academic field, we bridge the gap between the tools by devising a set of MDA transformations between the different models. We show that such approach is feasible although we also highlight how current state-of-the-art industrial tools are still partially inadequate for providing seamless support to MDA approaches for industrial Web applications.

1 Introduction

The design of a web application involves the definition of a data model (to specify the data used by the application), a hypertext model (to describe the organization of the front-end interface) and a presentation model (to personalize its graphical aspect).

In industrial web applications, data and hypertext models easily become huge and very complex. Consequently, their definition is an inherently error prone process yet their correctness is especially important when web designs are used to automatically derive the implementation of the web application (as in [4]).

Unfortunately, support for web designs verification is rather limited. Common tools and methods for web application development present poor verification facilities, mainly purely syntax consistency analysis, as the reachability of all pages from the home page or the existence in the data model of all entity types and attributes referred in the hypertext model.

* This work has been partial supported by the Italian grant FAR N. 4412/ICT, the Spanish-Italian integrated action HI2006-0208, the grant BE 00062 (Catalan Government) and the Spanish Research Projects TIN2005-09405-02-01 and TIN2005-06053.

A complete verification of web designs requires more formal model checking techniques. Currently, there are very few formal verifiers for web applications. Furthermore, using them properly requires deep knowledge of temporal logic (model checking techniques formalize the design to be verified as a finite state machine while the properties to check are expressed in temporal logic). The syntax and semantics of these concepts are completely alien to web designers, who are used to think in terms of pages, links, forms, and so on. This impairs the applicability of model checking tools and prevents their wide adoption in industrial web development projects.

We believe that current state-of-the-art technologies and tools makes now feasible to align Web application modelling and formal model checking so that we can get the best of both worlds: the usability and expressivity of graphical web modelling languages together with the verification capabilities of model checking tools. Clearly, such alignment could boost the adoption of model driven web development methods in the software industry, cutting down the cost of web application development and, at the same time, improving its quality.

As a first attempt to fully integrate formal verifiers with web modelling tools we have devised an MDA framework for the transformation of web models into the formal models expected by model checking tools. In this sense, the aim of this paper is to report our experience and, based on its results, to analyze the main issues that still remain to be solved in the tools for design and verification of Web applications.

Our framework comprises a set of tools that support several standards defined by the OMG. These standards include UML (Unified Modeling Language), MOF (Meta-Object Facility), XMI (XML Metadata Interchange), and MOF/QVT (Query/View/Transformations), among others. The choice of this transformation framework brings a set of benefits with respect to other more direct transformations (as XSLT or Java implementations): (1) it overcomes the limitations on the complexity of the XSLT transformations; (2) it provides the flexibility (reusability) to easily include other types of hypertext models in the framework and (3) it brings the possibility of formally verifying the transformation process (since in our approach both the source and target languages as well as the own transformation are specified by means of formally specified metamodels)...

The rest of the paper is structured as follows. Sections 2 and 3 present Wave and WebML, respectively. Section 4 shows the set of tools we combine to support our approach for web development. Finally section 5 draws some conclusions and outlines future work.

2 Wave: A Web Application Verifier

Wave [6] is a tool in Java for verifying temporal properties of data-driven Web applications. Wave takes as inputs the specification of a Web application (written in a declarative, rule-based specification textual language) and a property (expressed in first-order temporal logic), and outputs whether the property is true or false for that Web application together with useful verification information (such as a counter example in case of a false property).

A property is true when all possible runs (i.e. run-time executions) of the web application satisfy the property. In this sense, Wave goes beyond purely syntactic analysis

and permits to verify all common temporal verification properties (e.g. see [7]). As an example, with Wave we can check that before shipping an order the user has paid that same order in some previous web page. Instead, with a syntactic analysis we can just check that the shipping page is not reached before the payment page; we cannot ensure that the shipped order is the order paid in a previous payment page.

3 WebML

WebML [12][4] is a high-level notation for data-, service-, and process- centric Web applications. It allows specifying the data model of a Web application and one or more hypertext models (e.g., for different types of users or for different publishing devices) used to publish and manipulate the underlying data.

Each hypertext is called site view and is defined as a graph of pages, consisting of connected units, representing at a conceptual level the primitives for publishing contents into pages: the content displayed by a unit is extracted from an entity type, possibly filtered. Units are connected by links carrying data from a unit to another, to allow computation of the hypertext and definition of navigational paths for users. Hypertext models also include content-management units to specify modification operations (insert/update/delete) on the data underlying the site.

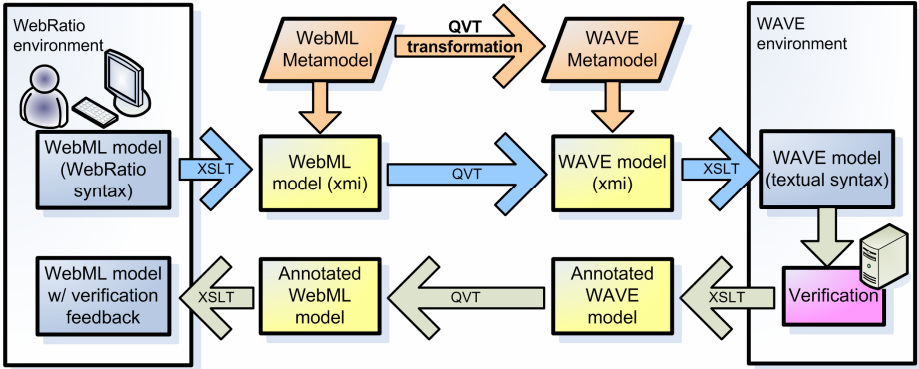
WebML is supported by the CASE tool called WebRatio [13], which provides facilities for designing WebML models and for automatically generating the running code of the application. However, WebRatio provides no support for the verification of generic user-defined properties for the hypertext. The only checking that is performed is related to the correctness of the Web application with respect to the WebML syntax and semantics.

4 Tool Framework

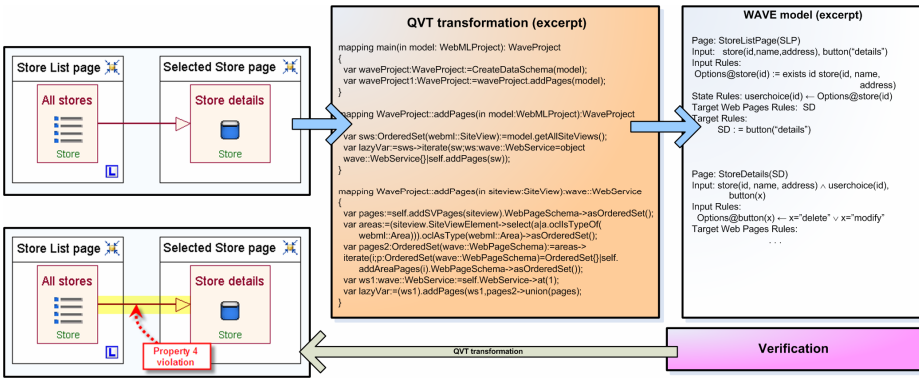
Our formal verification of web designs is implemented through collaboration among several tools. In this section we comment the different steps of the web development process when using our framework and describe the tools employed in each step. An overview of the whole process is shown in Fig. 1. The upper part (Fig. 1a), shows the interactions among the tools and their input and output models. The bottom part (Fig. 1b) sketches the application of the framework over a very simple WebML model.

Step 1. *Specification of the web application models with the WebRatio CASE tool* [13]. In our approach, the design of a web application starts with the specification of the data and hypertext models in WebML [4] using WebRatio. WebRatio stores all specified models in a single XML file. Bottom left part of Fig. 1 shows a simple hypertext model comprising a page with an index (*All stores*) and a page with the details of the selected object (*Store details*), connected by a link.

Step 2. *Transformation of the WebRatio syntax into an XMI representation.* The XML file exported by WebRatio must be transformed into an XMI representation [9] to fit into the MDA transformation framework. Such transformation is implemented through XSLT rules. The XML Schema of the XMI representation follows the structure of the WebML metamodel (see the next step).



(a)



(b)

Fig. 1. Overall view of the proposed framework and its application over an example model

Step 3. QVT-Transformation of the WebML models into Wave models. This step bridges the gap between WebRatio and the Wave model checker. The transformation is formally defined as model-to-model transformation [5], with rules specified using the standard QVT-Operational language [10]. The execution of the transformation is performed with the help of the Borland Together Architect tool [3]. Together supports the QVT-Operational language in the rule specification. Given a transformation definition and an input model, Together produces the output model according to the specified transformation rules. Before applying the transformations in Together, we need to previously perform the following tasks:

- Definition of new metamodels for the WebML and Wave languages. Model-to-model transformations assume the existence of a metamodel (in particular a MOF metamodel [11]) for the initial and target languages of the transformation. Based on the concrete syntax of both languages we have developed and integrated in Together new WebML and Wave metamodels. The WebML metamodel we use is a slightly revised version of the one presented in [8].

- Definition of the transformation rules to map WebML concepts to Wave concepts. Each rule transforms a WebML element into its corresponding Wave element.

In Fig. 1b we show a small transformation excerpt and the Wave model resulting from applying it over the initial WebML model.

Step 4. *Transformation of the Wave XMI representation into the Wave textual syntax.* As a result of the previous transformation, Together stores the generated Wave model in an XMI file. The structure of this XMI file is compliant with the (Together) Wave metamodel. In order to load the model into the Wave tool we must first transform this XMI representation into the (textual) syntax expected by Wave.

Step 5. *Model checking of the resulting Wave model.* Once the model is loaded in Wave, the designer can start checking that the model verifies the properties he/she is interested in. Such properties must be specified in LTL (Linear Temporal Logic). To facilitate their definition we developed a tool [2] that provides a visual notation to express LTL formulas. Then, these formulas are translated into LTL textual syntax.

Step 6. *Improvement of the WebML models according to the verification feedback.* If the verification determines that some properties are invalid, the designer can refine the WebML model to solve the issues. Obviously, the designer must be able to understand the feedback. To this purpose, the feedback should be expressed in terms of the elements of the original WebML models and not in terms of the generated formal Wave models. This can be achieved through reverse model transformations that highlight and annotate the original WebML models, to make the issues evident into the WebRatio tool. Therefore, the designer will see annotations on the WebML elements corresponding to the Wave elements violating the verified properties. In Fig. 1b, *Property 4* appears to be violated by the link connecting the two WebML units.

5 Conclusions and Further Work

This work presented a framework for the design and verification of Web applications, based on the WebML visual notation and on the Wave formal language and verifier. We described the transformation between them and we discussed how the framework could exploit the benefits of both languages. Indeed, the usability and readability advantages of WebML are coupled with the formal specification and verifiability properties provided by Wave. Thanks to Wave, general properties specified by the user can be checked, included rules on the execution and on the navigation of the user. Refer to [1] for additional material (as the full transformation rules, the meta-models description, etc.) regarding this tool framework.

As mentioned in the introduction, our approach is based on a set of tools and standards defined by the OMG. However, several issues were encountered in using these OMG standards: poor support for QVT transformations in current CASE tools; compatibility problems between the different XMI representations expected by each tool (i.e., UML tool vendors fail to generate fully XMI-compliant specifications of the

models); a transformation architecture more complex than we expected at the beginning (a lot of steps are necessary to go from the xml generated by WebRatio to the textual input of Wave), partially due to incompatibilities between the tools.

From our experience with this framework, we draw two main conclusions: (1) it is feasible to align web modelling languages and formal verifiers; and (2) much effort is still necessary to simplify the whole process. Otherwise, it will be difficult to foster the use of our framework (or similar ones) among industry partners, despite its clear benefits to improve the web development process. Fortunately, tool support for the OMG standards is continuously growing, so we can expect that some of the drawbacks encountered will be lessened in a near future.

Further work goes in the direction of improving the usability of the framework, by means of a seamless integration with the WebRatio tool in a way that allows the designer to be unaware of the intermediate transformations, thanks to proper visual interfaces for specifying the properties to be checked and for presenting the obtained results directly within the CASE tool. Moreover, predefined properties should be provided to the designer in the form of patterns that when applied over a concrete hypertext model, generate automatically the appropriate LTL formulae for that model. Another approach for simplifying the property definition is to allow visual specification of properties too, like in the solution presented in [2].

References

- [1] Brambilla, M., Cabot, J., Moreno, N.: Tool Support for Model Checking of Web Application Designs (2007) <http://www.elet.polimi.it/upload/mbrambill/webmlwave>
- [2] Brambilla, M., Deutsch, A., Sui, L., Vianu, V.: The Role of Visual Tools in a Web Application Design and Verification Framework: A Visual Notation for LTL Formulae. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 557–568. Springer, Heidelberg (2005)
- [3] Borland Together Architect (2007) <http://www.borland.com/us/products/together/>
- [4] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, Seattle, Washington (2002)
- [5] Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches, IBM Systems Journal, vol. 5(3) (2006)
- [6] Deutsch, A., Marcus, M., Sui, L., Vianu, V., Zhou, D.: A Verifier for Interactive, Data-driven Web Applications. In: SIGMOD'05 (2005)
- [7] Holzmann, G.: The Spin Model Checker Primer and Ref. Manual. Addison-Wesley, London, UK (2003)
- [8] Moreno, N., Fraternali, P., Vallecillo, A.: A UML 2.0 Profile for WebML Modeling, Model-Driven Web Engineering. In: ICWE'06 Workshops (2006)
- [9] OMG. XML Metadata Interchange (XMI) Specification v.2.0. (formal/03-05-02) (2003)
- [10] OMG. MOF QVT. Final Adopted Specification (ptc/05-11-01) (2005)
- [11] OMG. Meta Object Facility (MOF) Core Specification, (formal/06-01-01)(2006)
- [12] WebML.org (2007) <http://www.webml.org>
- [13] WebRatio 4.3 (2007) <http://www.webratio.com/>

Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA

Roberto Acerbis¹, Aldo Bongio¹, Marco Brambilla²,
Massimo Tisi², Stefano Ceri², and Emanuele Tosetti³

¹ Web Models

Piazzale Gerbetto, 6. 22100 Como, Italy

{roberto.acerbis, aldo.bongio}@webratio.com

² Politecnico di Milano

Piazza L. Da Vinci, 32. 20133 Milano, Italy

{mbrambil, tisi, ceri}@elet.polimi.it,

³ Acer Europe Services

Via Lepetit, 40. 20020 Lainate (MI), Italy

emanuele_tosetti@acer-euro.com

Abstract. This paper addresses the problem of developing enterprise-class eBusiness solutions in a more economically viable and time-effective way, by adopting Model Driven Development (MDD). Specifically, we report on an experience of more than six years of collaboration between Acer Inc. (the 4th branded PC vendor worldwide) and Web Models, an Italian startup company spinoff of Politecnico di Milano, innovator in the market of software tools and methodologies for MDD. The results clearly demonstrate that MDD can shorten the development of complex eBusiness solutions, improve the quality and conformance to requirements, and increase the economic profitability of solutions, by lowering the total cost of ownership and extending the life span of systems.

Keywords: Model Driven Development, WebML, industrial case study.

1 Introduction

The advent of the Web as a universal platform has initially facilitated the shift towards enterprise integrated applications, by offering a standard means of distributing data and functions. However, the unprecedented speed of evolution typical of the Web and the fierce competition on technologies among the major ICT players challenges the long-term sustainability of IT projects, due to a number of cross-cutting complexity factors: the spectrum of relevant standards and architectures constantly increases; the learning curve of technologies is higher than their evolution rate; outsourced or distributed development challenges classical software project management.

A solution to the growing complexity of IT projects requires a thorough innovation of the approach to software development, as advocated by the modern research on software engineering, which proposes Model Driven Development (MDD) as a means of improving the governance and end-to-end productivity of software [10]. In essence, MDD promotes a novel approach to software development centered around the

notions of: *platform independent model (PIM)*, which is a representation of the system's functionality independent of any specific technology, and *model transformation*, which is the process of progressively refining high-level models into lower-level ones, until an executable model on a concrete platform is reached.

This paper elaborates on a six-years experience of applying MDD to a set of enterprise-scale applications, developed by the EMEA branch of Acer Inc. using WebRatio, an innovative MDD methodology and tool suite based on the WebML meta-model[2]. The milestones of the work can be summarized as follows:

1. 2000-2002: The introduction of MDD methods and tools in the company, as a means of mastering the deep internal reorganization and technology change.
2. 2002-2004: The consolidation of the MDD approach as a key success factor in mission-critical applications, which led to its diffusion outside the marketing, e.g., to the distribution channel management, sales, and financial services sectors.
3. 2004-today: The integration of the MDD approach within the design of a global Service Oriented Architecture, with the aim of managing large-scale projects, involving not only internal roles, but also distributors and partners.

Model Driven Development seemed the most adequate methodology for bringing software development to non-IT business units. The idea was to exploit the knowledge about the business processes of the marketing personnel, delegating as much as possible of the construction of the implementation code to suitable development tools.

The reference model chosen by Acer is WebML, a model-driven methodology that builds on several previous proposals for hypermedia and Web design notations, including HDM, HDM-lite, RMM, OOHDM, and Araneus[1]. The design principles, notations, and development procedures of WebML are described at large in [2]; HDM [6] pioneered the model-driven design of hypermedia applications and influenced several subsequent proposals like RMM [7], Strudel [4], and OOHDM [8]; while other approaches (e.g., [3]) were inspired by object oriented models. All these methods offer powerful built-in navigation constructs, as opposed to WebML, which exploits simple, yet orthogonal, composition and navigation primitives.

2 Case Studies

Acer-Euro. The first version of the Acer-Euro application (Acer-Euro 1.0) aimed at establishing a software infrastructure for managing and Web-deploying the marketing and communication contents of 14 countries out of the 31 European Acer national subsidiaries. The Acer-Euro 1.0 system supported the two main functions of Content Publishing and Content Management in a seven-steps distributed workflow, illustrated in Figure 1, involving Local and Central Product Managers (LPMs and CPMs), Central Marketing Managers (CMMs), the central IT department, and the Internet Service Providers (ISPs). In this way, Acer could completely renovate the content and workflow of the marketing and communication functions, while reusing the existing national Internet infrastructures and contracts.

Acer-Euro 1.0 had a very tight schedule. Only seven weeks elapsed from the approval of the new site map and visual identity to the publishing of the 14 national web sites (plus the CMS). In this period, two distinct prototypes were formally approved

by the management: prototype 1, with 50% of functionality (end of week 2), and prototype 2, with 90% of functionality (week 5). Overall, 9 prototypes (2 formal, 7 for internal assessment) were developed in 6 weeks. The development team consisted of four persons: one business expert and one junior developer from Acer, and one analyst and one Java developer from Politecnico di Milano.

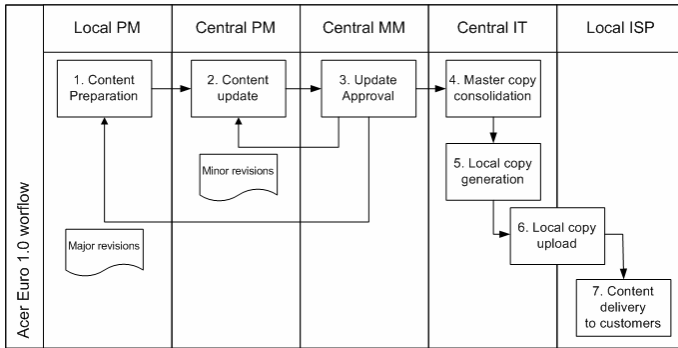


Fig. 1. Acer-Euro 1.0 workflow

As Table 1 clearly shows, the most relevant aspect of the development of Acer-Euro 1.0 is the short time-to-market with respect to the complexity of the application. Such result has to be ascribed to: (i) the high degree of automation of the process thanks to the model-driven approach (90% of the code were synthesized automatically); (ii) the overall productivity rate of 131,5 *function points/staff month*, which is 30% greater than the maximum value expected for traditional programming languages in the SPR Tables [9]; (iii) the velocity in focusing the requirements, thanks to the rapid production of realistic prototypes; (iv) immediate stress test and architecture tuning thanks to code directly generated for the actual delivery platform. Moreover, the benefits of MDD were even more sensible in the maintenance and evolution phase, leading to four major releases between 2001 and 2003 and to 13 multi-language intranet and internet applications, serving additional corporate processes.

Acer Connect. In June 2001, a spin-off project, called Acer Connect was scheduled, to address the delivery and management to the sales channel operators (Acer partners). This is a multi-actor extranet application characterized by: segmentation of the users into a hierarchy of user roles; different access privileges and information visibility to roles; one centralized and several local administration roles, able to perform advanced administrative and monitoring tasks; several group-tailored Web applications (e.g., sales, marketing) targeting contents to corporate-specific or partner-specific communities; a security model for managing group and individual access rights on single pieces of contents; full-fledged content personalization.

The first version of Acer Connect was deployed in Italy and UK in December 2001, after only seven elapsed months of development and with an effort of 24 man months. Today, Acer Connect is rolled out in 25 countries, delivering content and services to a community of over 80.000 users. Acer Connect and Acer-Euro share part of the marketing and communication content, and therefore the former project was

realized as an evolution of the latter. The model-driven approach greatly reduced the complexity of integration, because the high-level models of the two systems were an effective tool for reasoning about the functionality to reuse and develop.

Besides Acer-Euro and Acer Connect, several other projects were spun-off, to exploit the customer and partner communities gathered around these two portals, which collectively totalize over 10.800.000 visits per month. As a remark on the long-term sustainability of MDD, we note that, despite their complexity and multi-national reach, both Acer-Euro and Acer Connect are maintained and evolved by one junior developer each, working on the project at part time.

Table 1. Main dimensional and economic parameters of the Acer-Euro project

Class	Dimension	Value
Time & effort	Number of elapsed workdays	49
	Number of development staff-months (analysts and developers)	6 staff-months
	Total number of prototypes	9
	Average elapsed man days between consecutive prototypes	5,4
	Average number of development man days per prototype	15,5
Size	Number of localized web applications	14 B2C, 4 CMS
	Number of supported languages	12
	Number of data entry masks	39
	Number of automatically generated database tables	46
	Number of automatically generated database views	82
	Number of automatically generated database statements	279 queries, 89 updates
	Number of automatically generated JSP page templates	48
	Number of automatically generated or reused Java classes	250
	Number of automatically generated Java lines of code	12500
	Number of manually written SQL statements	17 constraints
Degree of automation	Percentage of automatically generated SQL code	96%
	Number of manually written/adapted Java & JSP components	10% JSP
	Percentage of automatically generated Java and JSP code	90% JSP, 100% Java
	Total cost of software development of first version	75.000 €
Cost, ROI, and productivity	HW, SW licenses, and connectivity cost of first version	70.000 € (db)
	Return on investment of first version	12-15 months
	Average effort of extension to one additional country	0,5 staff-months
	Average cost of extension to one additional country	7.500 €
	Average ROI of extension to one additional country	2 months
	Number of function points	789
	Average number of function points delivered per staff-month	131,5

3 Results and Critical Considerations

In this section, we summarize the main lessons learned in the application of the MDD principles to web development. The major effect of MDD is to shift the focus of development from implementation to requirement analysis. Almost 80% of the delivery effort concentrates in the phases of data design, hypertext design and prototyping. This means that more development time is spent with the application stakeholders, to refine design models and evaluate prototypes. The result is a better quality of the delivered applications and a higher rate of acceptance, because design errors and requirements misinterpretations are eliminated as early as possible. MDD also benefits the more technical tasks of testing, maintenance, and evolution, because reasoning on the system is far more effective at the conceptual level than at the physical level.

MDD lowers the technical barriers for developing complex Web applications, allowing a more flexible distribution of responsibilities between the IT department and the business units. When business goals are rapidly evolving, and quick adaptation to changing environment is a critical factor, the possibility of developing, monitoring and adjusting the systems directly by the business units greatly improves efficiency.

The deployment consisted of J2EE standard architectures, with the integration of heterogeneous systems taking place by means of Web services. A well-defined architectural protocol can be established to integrate systems autonomously developed in different business units, avoiding the duplication of software functions and data.

Last but not least, MDD has proven an economically profitable and sustainable way of developing Web systems. The peak productivity rates experienced in the Acer projects has reached five times the number of delivered function points per staff-month of a traditional programming language like Java [9].

On the negative side of MDD, the initial training costs must be considered. MDD requires non-technical knowledge on the modeling of software solutions, which must be acquired with a mix of conventional and on-the-job training. Acer estimates that it took from 4 to 6 months to have fully productive developers with MDD, WebML, and WebRatio. However, as Figure 2 shows, the initial investment in human capital required by MDD pays off in the mid term. The number of applications developed and maintained per unit of personnel increases with the developers' expertise and exceeds ten fully operational, complex and distributed Web applications per developer.

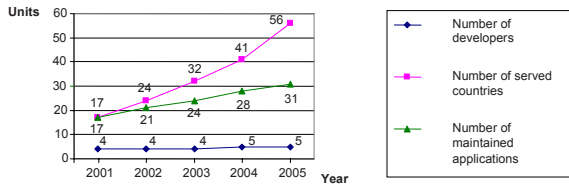


Fig. 2. Evolution of manpower versus number of maintained applications and served countries

4 Conclusions

In this paper we have reported on a long-term experience in applying Model Driven Development to the construction of mission-critical eBusiness solutions, jointly conducted by Acer and WebModels, by exploiting the WebML model. After more than five years of applying MDD with WebRatio, Acer has gained sufficient experience to draw some general conclusions. Today, the use of WebRatio has spread from the Acer-Euro project to most of the Web-based B2C, B2E, and B2B platforms of Acer EMEA and has been exported from Europe also to Acer PanAmerica. The developed solutions cover all the most critical sectors of Acer's business and have given tangible benefits over the years. The abovementioned portfolio of solutions has been deployed, and is continuously being maintained, by an internal team consisting of five developers only. With a traditional development methodology and using conventional programming-oriented tools, the company estimates that the construction of the deployed systems would have required at least three times the resources that have been

invested. None of the developed systems has been retired or has become obsolete; new requirements have been smoothly incorporated into the running versions and rolled out by iteratively extending the deployed systems.

The Acer experience has demonstrated the feasibility of MDD and the efficiency it introduces into the development lifecycle, largely anticipating the current debate on the Model Driven Architecture. However, the adoption of a model-based approach must also extend to the maintenance and evolution steps (which account for over 60% of the total lifecycle cost), where they provide the best advantages in terms of cost and productivity.

In conclusion, MDD appears to be a powerful tool for renovating businesses and taking advantage of the advent of low-cost distributed network infrastructures. However, the transition requires innovation not only in the business strategies but also in the IT departments.

Our future work will concentrate on improving and extending the quantitative assessment of the benefits of Model Driven Development in the Web application sector. A novel software tool for automatically performing the evaluation of software projects size and effort is under development, which will support the measurement of different project parameters related to size, effort, and cost.

References

- [1] Atzeni, P., Mecca, G., Merialdo, P.: Design and maintenance of data-intensive Web sites. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 436–450. Springer, Heidelberg (1998)
- [2] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, Seattle, Washington (2003)
- [3] Conallen, J.: Modeling Web Application Architectures with UML. *Communications of the ACM* 42(10), 63–70 (1999)
- [4] Fernandez, M., Florescu, D., et al.: Catching the boat with Strudel: Experiences with a Web-site management system. In: 24th ACM SIGMOD, Seattle, WA (1998)
- [5] Fraternali, P.: Tools and approaches for developing data-intensive Web applications: A Survey. *ACM Computing Survey* 31(3), 227–263 (1999)
- [6] Garzotto, F., Paolini, P., Schwabe, D.: HDM - A Model-Based Approach to Hypertext Application Design. *ACM TOIS* 11(1), 1–26 (1993)
- [7] Isakowitz, T., Stohr, E., Balasubramanian, P.: RMM: A Methodology for Structured Hypertext Design. *CACM* 38(8), 34–44 (1995)
- [8] Rossi, G., Schwabe, D., Lyardet, F.: Web Application Models are More than Conceptual Models. In: Kouloumdjian, J., Roddick, J.F., Chen, P.P., Embley, D.W., Liddle, S.W. (eds.) *Advances in Conceptual Modeling*. LNCS, vol. 1727, pp. 239–252. Springer, Heidelberg (1999)
- [9] Software Productivity Research: SPR Programming language Table – Version PLT2005a (2005) <http://www.spr.com>
- [10] Warner, J., Bast, W., Pinkley, D., Herrera, M., Kleppe, A.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley, London, UK (2003)

The Challenges of Application Service Hosting

Ike Nassi, Joydip Das, and Ming-Chien Shan

SAP America
3410 Hillview Avenue, Palo Alto, CA 94304
ming-chien.shan@sap.com

Abstract. In this paper, we discuss the major issues associated with the new model of software delivery – service on demand – and explain why it alters the economics of software. As this model is expected to deliver fundamental leaps in cost efficiency, operation performance, infrastructure orchestration and application control, we describe the supporting technology required to achieve these goals. We also highlight those crucial operational processes for enhancing the quality of software delivery under the service on demand model. We then briefly outline our research roadmap to develop an on demand operating environment based on the fundamental principles: standardization, repetition, and, ultimately, automation.

Keywords: Service hosting, service on demand, software as a service, service provisioning, resource virtualization.

1 Introduction

Buying and operating enterprise applications is not like buying traditional durable goods such as TVs or refrigerators – customers are not just buying a discrete product and simply plugging it in for use. Instead, they are buying into an R&D stream of maintenance and operation. Today, the total spent on enterprise IT exceeds one trillion dollars per year. However, amazingly, 75% of that is used for managing existing systems. IDC cites five major areas that cost companies millions of dollars annually: “CIOs and their departments really focus on five key aspects to the management and maintenance of their software systems: availability, performance, security, problems and change management.” For example, Wachovia Bank has been spending \$700,000-\$800,000 a year on consultant fees just to keep their old system up and running while, similarly, the amount of money spent by 200,000+ Oracle customers for only managing their Oracle software ranged between \$40 and \$80 billion per year. Maintenance and administration costs over time have been a major culprit in the high spending required to maintain the status quo. “Customers can spend up to four times the cost of their software license per year to own and manage their applications,” says Gartner Inc. Furthermore, the cost of any outage can range from \$100 to \$10,000 per minute of downtime, according to AMR Research.

As a consequence, customers don’t want to spend money managing their software environment and prefer someone to take over this responsibility but still provide the

same facility to support their business operations. This led to the advent of new models that delivers *software as a service* (SaaS).

In the 1990s, traditional client-server architectures were adapted to the SaaS model by Application Service Providers (ASPs) with limited success. The trend has continued with off-shoring of IT operations to take advantage of cheaper globalized resources provided by Business Process Outsourcing (BPO) vendors. However, these approaches focus on delivering traditional packaged software applications as services. While economies of scale lower cost, solution implementation and operation cycles continue to be complex and time consuming and primarily suitable for large enterprises. More recently, a new breed of software platforms and applications have emerged to take advantage of internet scale architectures to provide *software on demand*. New on-demand vendors are taking advantage of this model to drive significant adoption of software in the high-volume small and mid-size enterprise markets. In effect the *software on demand* model has become synonymous with the *software as a service* moniker.

Gartner, IDC and Fortune Magazine have all cited *on demand software* as one of the key megatrends and predicted that up to 40% of all applications will be delivered over the Internet within the next 2-3 years. As a matter of fact, people have already been using service on demand in their daily life, without being aware of it, e.g., even private individuals think nothing of using eBay and PayPal to sell goods to people half a world away, essentially making them small businesses outsourcing their IT needs to a sophisticated global vendor. This is an example of delivering enterprise applications on a needed-base via the Web for the masses.

The on demand model fundamentally alters the economics of software delivery through operational efficiency and performance, infrastructure orchestration and application control. In addition, it changes both the relationships between the software vendor and the customer, and the economics of purchasing and owning software. However, the profitable transition to such an on-demand model will require incumbent client-server application vendors to adapt and migrate to a new breed of technical solutions as well as fundamentally different operation models.

2 Basic Issues and Approaches

Service on demand addresses the cost of software delivery at several levels. First, standardized technology platforms that support internet scale architectures reduce the initial infrastructure costs. Second, a very high degree of automation with focus on availability, security and fault tolerance reduces the human cost of delivering software. Finally, applications that lend themselves to a high degree of modularization and non-disruptive change management allow seamless incremental deployments and pay-as-you go service acquisition.

The technology array used to support service on demand is classified into three categories:

1. **Resource virtualization.** This category of technologies aims to provide a single, consolidated, logical view of and easy access to all available resources in an environment. It is the process of presenting computing resources in ways that users

and applications can easily ascertain values from many types of integrated implementation rather than presenting them in a way dictated by a specific implementation, geographical location, or physical package. A meta-level of implementation views can in some sense describe the overall concept of virtualization. However, resource virtualization, specially at the hardware level alone is insufficient in driving down the cost of delivering service on-demand to create profitable margins.

2. **Service provisioning.** This category of technologies makes the right resources available to the right processes at the right time. The execution resource allocation and application management should support the adequate level of quality of service to meet the minimum SLA requirements contracted with customers. It is the end-to-end capability to automatically deploy and dynamically optimize resources, including servers, storages, network bandwidth, operating systems, middleware, applications, and third-party connections. It requires tools to manage service levels, meter system usage and performance, and billing for service usage, as well as monitor and optimize service provisioning processes.
3. **Application adaptation.** This category of technologies reshapes the existing/new software applications to be able to leverage the virtualized execution environment and internet scale architectures, achieving the maximum degree of performance and flexibility. For example, vendors like RightNow have started to build their software to be delivered as a service. As a result of this effort, the average deployment time of RightNow's call center solution from customer commitment to being live is less than 40 days, including integration and customization. Meanwhile, vendors like Oracle and SAP are both on the way to break up their application packages into modular components to facilitate service provisioning.

However, the service management and resource virtualization technologies alone cannot deliver profitability in the on demand service model. As indicated in a Gartner report, human operations were responsible for more than 50% of process operational failures. Additionally, numerous studies point to the cost of operations as the primary contributor to overall IT costs. As a result, in order to ensure a sustainable level of service quality while maintaining profit margins, we must do three things: First, we must standardize the fundamental infrastructure and building blocks. Second, we must develop specialization and repetition of the key availability, security, performance, problem, and change management processes. If a process is repeated, it can be automated. The next generation of complex software management cannot be based on people; it must be based on processes and automation, which is the key to higher quality and lower cost. Finally, the process can be optimized, meaning data from the process is used to change and improve the process. In summary, an on demand operating environment is an open standards-based, heterogeneous world, integrated, automated and freely enabled with self-managing capabilities.

In addition, we need to develop the operational processes focusing on the support of key service business operations, including configuration management, availability management, performance management, security management problem management, and change management. At SAP, based on our experience of managing thousands of software systems, a set of key management processes has been identified, including an event correlation and root cause analysis process, a disaster recovery management

process, a capacity management process, a resource management process, a production assessment process, an escalation management process, an update management process and a proactive problem management process.

To evolve all of the above-mentioned capabilities and deliver more choice in tailoring services to our customers, we at SAP Research Lab are developing a prototype researching an execution and development framework supporting the business of service on demand. We will also study how to codify changes to the existing application packages facilitating the service on demand operation.

3 Additional Challenges and Directions

The basic business model for service on demand tends toward fixing functionality to put an upper limit on cost. But businesses change. In today's dynamic business environment, without the ability to innovate, companies will risk watching more nimble competitors eat into their established markets. Therefore, the business objectives for customers to move to the service on demand model is not just trying to reduce the total cost of ownership, but also more critically, to reduce the time of implementation of new composite services to support their end-to-end business operations. This is because, in the past, most vendors had constrained the ability to change the software, providing very basic offering of their services. As the service business continued to mature, there is a realization that, for larger mid-size companies, there is a need to extend and tailor the applications to the customer's business.

In addition, large global service providers always attempt to define what they called "killer applications." These killer applications were competitive services predicted to be highly utilized by a wide variety of customers. Oftentimes, these applications were time-intensive to create and somewhat costly to maintain. The on demand business ecosystem must afford creators the ability to quickly create and push out to their users, virtually eliminating risk from the large enterprise in determining what best killer application was the target, so as to provide to their customer base the most robust service.

Based on the observations mentioned above, it becomes clear that the ability of a service provider to enable the quick response to ever-changing customer demands will determine who survives in today's rapidly changing marketplace. It requires a new set of composite service development technology fabric that can adapt on demand business to these ever-changing requirements. Currently, SAP is exploring a new category of services, called *tool on demand*, to provide a hosted development environment for composite service development, deployment, monitoring and maintenance in its hosted environment in addition to its hosted service on demand environment.

Last, as adoption of its offerings grows, SAP may also consider expanding its services to include offering its NetWeaver application server as a service. This next level of outsourcing, called *execution environment on demand*, offers customers the flexibility to not only use the framework to tailor the software, but also use independent software vendors and custom-written applications that run on SAP technology to provide more flexibility to support their business needs.

References

1. Chou, T.: The hidden cost of Software. ASPNews (May 2003)
2. Fellenstein, C.: On Demand Computing, IBM Press (2005)
3. Murphy, B., Gent, T.: Measuring system and software reliability using an automated data collection process. Quality and reliability engineering international.
4. Oppenheimer, D., Ganapathi, A., Patterson, D.: Why do Internet services fail, and what can be done about it?. In: Proc. Of the 4th USENIX Symposium on Internet Technologies and Systems, Seattle, WA (March 2003)
5. Rowell, J.: A step-by-step guide to starting up SaaS operations. OpSource (2004)

Securing Code in Services Oriented Architecture

Emilio Rodriguez Priego and Francisco J. García

Departamento de Matemáticas y Computación
Universidad de La Rioja
Edificio Vives, Luis de Ulloa s/n
E-26004 Logrono (La Rioja, Spain)
{emilio.rodriquez, francisco.garcia}@unirioja.es

Abstract. SOA proposed security mechanisms are only centered in the data transmitted between service provider and consumer. However, it's well known that the biggest threats to the integrity of the information are precisely focused not on the data directly but on the code that manages it. Our main statement is that it will only be possible to reach an acceptable level of security if the protection mechanisms cover not only the data but also the code that process these data. In this paper we present a new approach about mobile code security based on the Services Oriented Architecture Reference Model and Web Services technology. This new model allows the development of systems with end-to-end security, where all elements (code and data) are secure.

Keywords: Web services security, mobile code security, Service Oriented Architecture.

1 Introduction to the Problem

Nowadays, there is a growing interest in Web Services technology and Service Oriented Architecture (SOA), whose Reference Model has been recently approved as an OASIS standard [2]. In this model, a consuming entity requests from a supplier entity, one or more services under a set of conditions (interaction, visibility, execution context, policies and contracts). At the same time, security is one of the aspects that requires more attention due to the application of this technology to environments where information exchange is made through public networks like Internet, in which there potentially exists diverse security threats. Recently different standard-based solutions have been proposed to solve the problem of secure sending and reception of messages. According to SOA, the performed action details of the supplied service are not typically visible by the consumer [1, 2]. Therefore SOA does not address the subjects related to the realization of a service, like, e.g. securing it, delegating the effective resolution of these problems to the supplier.

The proposed security mechanisms of the SOA model and the technology of web services are centered in the transmitted data (message), and they put their focus on end to end integrity, confidentiality, identity and authentication. These mechanisms work well and in practice they reach the objective. However, it's well known that the biggest threats to the integrity of the information are precisely focused not on the data directly but on the code that manages it [8,9].

Therefore, our main statement is that it will only be possible to reach an acceptable level of security if the protection mechanisms cover not only the data but also the code that process these data.

Independently of the web services development and SOA model, the security problem of mobile code and its interaction with the environment in which code is executed has been studied in the last years, particularly for the singular case of mobile agents [6]. Both SOA and mobile code have specific and common aspects of security but until now had not been treated jointly. The main contribution of this article is the proposal of a new approach to the problem of the security of mobile code, named “Web Services based Secure Code” (here-in-after WSbSC) that it’s based on SOA Reference Model and the Web Services Architecture.

WSbSC allows for the improvement of security in a typical interaction between consumer and provider without forcing the participants to necessarily know the details of implementation of the services.

Our model is virtually applicable to any SOA situation in which an integral model of security, involving data and code was required. However, in certain situations code visibility, integrity and/or portability are more important, because code integrity, the code in itself, the source of the code or all of these elements, take part or are exchanged as part of services provided by the supplier. Typical examples of these application environments are distributed processes, hosting or rent of processes, auditing and validation of code by certification entities, and so on.

Once the problem that we want to address has been stated, the rest of the paper is organized in two main sections, each one describing its own objectives, resolution outline and methodology. In Section 2, we provide a general description of the WSbSC reference model. Section 3 presents the application of the model to a basic SOA interaction. The last section summarizes the main contribution, related work, the status of the research and indicates the future work.

2 WSbSC Reference Model

The reference model of WSbSC (here-in-after WSbSC-RM) is an abstract framework for understanding significant relationships among service entities (providers and consumers) that allows an integral (data and code) secure interaction, enabling the development of specific architectures using consistent standards or specifications.

WSbSC-RM relies on SOA-RM and it adds new concepts and relationships to the modeling of data and code exchange based on services.

The central concept in WSbSC-RM is the code, just as it has been defined traditionally but with some specific features: (1) the code can be portable: i.e., it can be sent from one system to another without manual intervention; (2) the code can be executed in any compatible execution environment; (3) transmission, load and execution of the code can be carried out in a safe way, applying the same basic principles of secure transmission of data (identity, integrity and confidentiality); and (4) the code can be verified in a secure manner.

There have been different proposals related to code portability, validation and execution in distributed environments [4,5,7]. Most proposals are based on hardware or software techniques for execution in a local environment.

WSbSC-RM states that the transmission, reception, execution, load, compilation and validity of the code are *services* that can be offered by systems potentially remote and weakly coupled. As we see below, with WSbSC *code is not only externally verifiable* [3], *but also externally compilable and externally executable*.

WSbSC-RM distinguishes the following actors:

- *Author*: it's the owner and creator of the code and its legal owner.
- *Supplier*: provides the code to a consumer and distributes it by author permission.
- *Client*: uses the code provided by a supplier.
- *Verifier*: verifies the code according to a security policy previously established.
- *Compiler*: given a code, it compiles another functional equivalent code.
- *Processor*: possesses an execution environment that executes the code.

All WSbSC-RM actors are service consumers or providers, from the point of view of SOA-RM. Besides WSbSC-RM allows the modeling (recursively) of the actions (local or remote) of a service by composing services offered by these actors and according to code-centric policies and contracts. What is a key added factor of our approach with reference to SOA, is that actors playing the role of consumers in any relationship to a provider may impose a certain security policy to regulate the service that the provider is going to perform. This policy, and here is the contribution, does not only affect the data (message) as SOA does, but also the service implementation. This policy refers to one or several security aspects (such as integrity, confidentiality, validity, and so on) and may specify a mechanism or set of mechanisms that the provider must implement to accomplish the policy. These security requirements can be used by the consumer to select the most suitable provider in each case, depending on the mechanisms the former can implement. The response to each service request will include, as well as the result, metadata about the required, and fulfilled, policy.

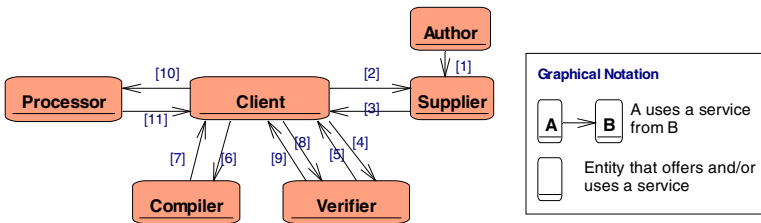


Fig. 1. General WSbSC-RM scenario

At this point, we have not studied yet all relationships between WSbSC-RM actors like service actors, but we can illustrate these relationships by describing a general example that shows some key concepts in SOA-RM: service, interaction, policies and contracts, real world effect, etc. Fig. 1 shows this general scenario. Interactions involve the following steps: (1) An *author* creates the code and sends it to a *supplier* for distribution. (2) A *client* localizes and requests the code that satisfies its needs from the *supplier*. (3) The *supplier* delivers the code. (4) The *client* requests the verification of the code according to the *client* policy from a *verifier*. (5) The *verifier* delivers the validated code. (6) If code is not compiled for the architecture in which is going to be

Applying WSbSC, interactions in this basic scenario are: (I) a *consumer entity* requests a *service* from a *provider entity*. The security policy of *consumer entity* establishes that the *provider entity* must implement the service using WSbSC and, therefore the *provider entity* must get a PSC of the service implementation to achieve this policy. (0-9) The *provider entity*, if it hasn't done so previously, carries out the process of creation, supply, validation and generation of the PSC and executes the service. (II) It returns the result of the service together with the PSC-cert. Note that (0-9) illustrate another case in which the *provider entity* delegates in the *supplier* the process to get the PSC.

By means of PSC-cert, the consumer entity can obtain an extra security level that certifies that the service was created, supplied, verified, compiled and executed by trust entities without being necessary to know the code itself. There can be diverse variations: e.g., the *provider entity* could submit a copy of the code, encrypted with the verifier's public key. If we suppose that both the *consumer* and the *provider* entities trust the *verifier*, then the *consumer* entity could request code validation to the *verifier*. Code confidentiality is guaranteed by encryption, and the *verifier's* public key encryption guarantees that only the *verifier* can decrypt, analyze and evaluate the code validity. Periodical tests and the use of several verifiers can improve security even more. To illustrate the work to be performed, the following listing outlines the structure of a PSC-cert in a simple case.

```
<wsbsec:psc xmlns:wsbsec=.. xmlns:uddi=.. xmlns:ds..>
<wsbsec:code EncodingType="Base64">CHVibG..</wsbsec:code>
<wsbsec:psc-cert>
  <wsbsec:AuthoredCode> ..</wsbsec:AuthoredCode>
  <ds:Signature ..> ..</ds:Signature>
  <wsbsec:SuppliedCode> (a)</wsbsec:SuppliedCode>
  <ds:Signature ..> ..</ds:Signature>
  <wsbsec:CompiledCode>.(a).</wsbsec:CompiledCode>
  <ds:Signature ..> ..</ds:Signature>
  <wsbsec:VerifiedCode>.(a).</wsbsec:VerifiedCode>
  <ds:Signature ..> ..</ds:Signature>
</wsbsec:psc-cert>
</wsbsec:psc>
```

AuthoredCode block is added at point 0 together with autor's signature of that block. The following blocks are added in the same way by each actor when they have finished their tasks. Note that sections marked with (a) consist of two main parts: metadata related to the actor (description, e.g., by means of uddi business entity, credentials, e.g., SAML authorization credentials, etc.) and metadata about its action, e.g., for the compiler: the compiler environment, the target language, and so on.

This section has outlined one of the alternative interaction scenarios among consumers and suppliers. In order to finish developing this section, we will (1) study more alternative interaction scenarios, (2) select the most suitable web services standards to implement PSC, (3) develop the WSbSC security model extending the WS-SecurityPolicy model, and (4) we will develop an actual case relevant enough to illustrate the different alternative scenarios.

4 Contribution, Related Work, Status and Future Work

Several solutions about mobile code has been proposed in the last years: e.g., [4,5,6] focus on execution environment (compiler, verifier and/or processor). [7] and more recently [10] suggest a contract between producer and consumer of mobile code. [11] defines a model-driven approach for service-oriented software development.

The main contribution of this paper is the proposal of a new approach to the problem of the security of mobile code, WSbSC, that it's based on SOA-RM and the Web Services Architecture. WSbSC provides a level of security that covers not only the data but also the code that process these data.

A lot of work must be done, both to specify the demanded policy and the process that must be preformed to implement it (perhaps using WS-SecurityPolicy or even BPEL), and in order to select the standards to be used at each part of the PSC-cert (SAML, WS-Policy, WS-Addressing, UDDI, and so on). As a future line of research we are planning to extend the model to portable objects, i.e., securing the object state as well as the code that manages that state (behavior), making this code PSC.

Acknowledgments. Partially supported by Comunidad Autónoma de La Rioja, project ANGI-2005/19.

References

1. Web Services Architecture (February 2004) <http://www.w3.org/TR/ws-arch/>
2. Reference Model for Service Oriented Architecture v1.0 October 2006 <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>
3. Seshadri, A., Luk, M., Perrig, A., van Doorn, L., Khosla, P.: Externally verifiable code execution. *Communications of the ACM* (September 2006)
4. Franz, M., Chandra, D., Gal, A., Haldar, V., Reig, F., Wang, N.: A portable Virtual Machine target for Proof-Carrying Code. In: *Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators* (June 2003)
5. Yau, S.S., Prasad, A., Zhou, X.: An Object-Oriented Approach to Containing Mobile and Active Codes in Large-Scale Networks, words. In: *Fourth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'99)* (1999)
6. Claessens, J., Preneel, B., Vandewalle, J.: How can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions, *ACM Transactions on Internet Technology (TOIT)* (February 2003)
7. Sekar, R., Ramakrishnan, C.R., Ramakrishnan, I.V., Smolka, S.A.: Model-Carrying Code (MCC): a new paradigm for mobile-code security. In: *Proceedings of the 2001 workshop on New security paradigms* (September 2001)
8. Whitman, M.E.: *Enemy At The Gate: Threats To Information Security*. *Communications of the ACM* (August 2003)
9. Sima, C.: Are your web applications vulnerable? (October 2004) <http://www.securitydocs.com>
10. *Security of Software and Services for Mobile Systems* (March 2006) <http://www.s3ms.org>
11. *SENSORIA* (October 2004) <http://sensoria.fast.de/>

Service Level Agreements: Web Services and Security*

Ganna Frankova

Dept. of Information and Communication Technologies, Univ. of Trento, Trento, Italy

ganna.frankova@unitn.it

Advisor: Prof. Marco Aiello, Co-Advisor: Prof. Fabio Massacci

Abstract. To support the quality of service guarantee from the service provider side, complex web services require to be contracted through service level agreement. State of the art on web services and web service compositions provides for a number of models for describing quality of service for web services and their compositions, languages for specifying service level agreement in the web service context, and techniques for service level agreement negotiation and monitoring. However, there is no framework for service level agreement composition and composition monitoring, the existing design methodologies for web services do not address the issue of secure workflows development. The present research proposal aims to develop concepts and mechanisms for service level agreement composition and composition monitoring. A methodology that allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis would benefit.

1 Introduction

The use of Web Services (WS) requires quality guarantee from the service provider. Taking into account that the guarantee depends on actual resource usage, the service client and provider must agree a priori by specifying an agreement. This allows the provider to allocate the necessary recourses to support the quality of service guarantees. Additionally, the guarantees on service quality must be monitored and the service client must be notified in case of failure to meet the guarantees.

The Service Level Agreement (SLA) opens a wide spectrum of challenges. In this research proposal we address some of them and propose building SLA framework for SLA composition and composition monitoring. The existing design methodologies for web services do not address the issue of developing secure web services, secure business processes and secure workflows. The present research proposal aims to develop a methodology that allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis.

* This work has been partly supported by the IST-FP6-IP-SERENITY project.

The work is structured as follows: Section 2 provides background and specifies the problem area. Section 3 is devoted to the research objectives and directions, i.e., the problem statement. The approaches and methods to be applied in order to achieve the objectives, the results obtained so far and a tentative plan for future work are presented in Section 4.

2 Background and Problem Area

With the term *Quality of Service (QoS)* we refer to non-functional properties of services. Although various approaches for modelling quality of service for web services have been addressed in a number of recent works, in [4] we claim that current models are far from ideal and there is a lot of space for further investigation and innovative research.

To optimally select component services for a quality-aware composite service building Zeng et al. [21] propose to use a global planning approach. In [13] Lin et al. doubt about the approach and propose a fuzzy way to express QoS requirements. The feasibility of using workflow patterns to determine the QoS in a web service composition is demonstrated by Jaeger et al. [8].

There are several proposals addressing the issue of web services design based on early requirements [12], [18], [9]. The design methodologies do not aim to design secure web services, secure business processes and workflows. Some approaches [5], [3] aim to design secure systems. On the negative site, the approaches do not support the design of software and business processes based on a service-oriented architecture.

Covering web services with SLA opens new research directions such as SLA specification and monitoring. Although such languages as SLAng [11], WSOL [19] and WSLA [15] for specifying SLA have been proposed, specification of SLA is still a research topic [20,16]. Fundamental concepts of non-functional SLA monitoring are presented in [17]. Web Service Level Agreement framework for defining and monitoring SLAs is presented in [10]. CREMONA, an architecture for creating and monitoring agreements in a context of web services is proposed in [14]. But all the approaches do not support monitoring SLA with the goal of anticipating terms violations.

Emergent Questions: SLA Composition and Monitoring

The cornerstone of web services success lies in the ability to compose web services on the fly in order to build complex added-value services. Dealing with quality aware web service composition requires the study of a number of problems, namely:

1. definition of a quality of service model to provide quality of service information at the level of individual web services;
2. building complex web services with multiple QoS constraints;
3. ensuring that a promised quality of service is actually provided during execution;

4. developing secure business processes and secure workflows in the web services context;
5. finding and monitoring global SLA of a composition of web services.

In [4], we have addressed the first and second issues. A solution to the third issue was proposed in [1] by using a methodology to handle changes during the interactions of web services and to prevent the violation of QoS constraints. In this work, we consider the above problems and focus on the fifth and fourth issue.

3 Research Objectives and Directions

Problem Statement

Covering web service aggregation with SLAs has opened a number of critical issues regarding SLA composition and monitoring, namely:

(1) **global SLA for web service composition calculation**

Assume that a SLA is defined for each web service from a set of n services, i.e., $SLA_1, SLA_2, \dots, SLA_n$. The services are used to build a web service composition (see the orchestration unit) and the client needs to know the global SLA of the whole composition, i.e., SLA_G . The scheme depicted on Figure 1 shows the corresponding scenario.

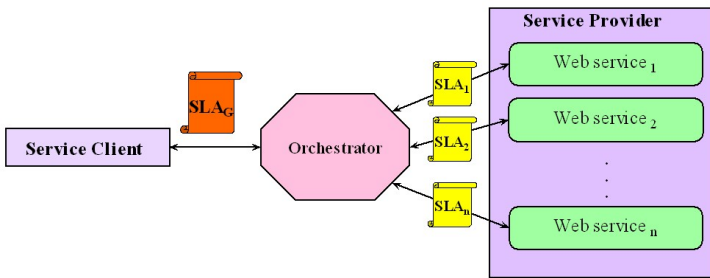


Fig. 1. “Service Client-Service Provider” interaction scheme

Depending on whether a goal of the clients is specified or not, one can consider two cases. *Case 1:* The client’s goal is not specified. A global SLA SLA_G of web service composition is calculated based on the SLAs SLA_i coming from the individual web services $S_i, i=1..n$ to be composed. Calculating the global SLA should consider the business process $BP(S_1, \dots, S_n)$. *Case 2:* The client’s goal is specified. Considering the specified SLA_G , such a set of web services S_i and SLAs $SLA_i, i=1..n$ covering the web services are received that the business process $BP(S_1, \dots, S_n)$ guarantees the specified goal. There is an optional input of a set of web services S_j and SLAs $SLA_j, j=1..m, m \leq n$ covering the web services in the scenario.

Our research will be focused on calculation of global SLA for web service composition. Technique for composing SLA development are seen as a challenge.

(2) SLA for web service composition monitoring

Usually SLA monitoring techniques consider guaranteeing the fulfillment of a service term. We claim that **how** the guarantee is fulfilled should be taken into account. The answer to the question “Is the guarantee close or far from being violated?” should be given by the SLA framework. Our SLA framework will provide monitoring based on the client’s goal described in SLA. It will not only predict and notify terms violations, but also discover the components of the composition that are responsible for the violation.

(3) secure workflow design based on early requirements

Requirements engineering has little counterpart in the development of business processes for web services. The existing design methodologies for web services do not address the issue of developing secure web services, secure business processes and secure workflows. This part of the work is devoted designing of secure business processes and secure workflows for web services. We aim to develop a methodology that allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis.

4 Approaches and Methods

In order to realize the research objectives, a number of issues has been addressed and the results that fit in the framework of the thesis are obtained, namely:

SLA Analysis

The WS-Agreement [6] protocol has been studied and the earlier missed formal definition of SLA in a context of web services has been provided. The proposed set of formal rules ties together agreement terms and the life-cycle of an agreement. From the analysis some shortcomings of the protocol have been discovered. In particular, (i) there is no checking of how close a term to be violated and (ii) breaking one term of agreement results in terminating the whole agreement, while more flexibility is needed. The possible evaluations of agreements have been found and ways in which to make an agreement more robust and long lived have been identified. Two extensions to the specification and supporting environment have been proposed. The first is used to anticipate violations, while the second is devoted to run-time renegotiation.

SLA Composition

In the proposed methodology, we use hypergraphs [2] to capture the structure of business process. We see the problem of finding the global SLA of a business process as the Business Process Hypergraph (BPH) decomposition path evaluation. Suppose that a BPH is a weighted hypergraph. Each decomposition arc is associated a set of weights that show contribution of a source node to the target one. Each leaf node is assigned with a QoS value that corresponds to the QoS of atomic service. Each decomposition arc is assigned with an aggregation function which calculates the value of a target node taking as arguments

source nodes and the set of weights. In order to evaluate a BPH decomposition path, one has to design the aggregation functions. In our approach the design of the aggregation functions depends on the structural activity that represent each compound service in BPH. The structural activity represents the basic structural elements of a composition as sequence, parallel execution, switch. Then, it is possible to perform the aggregation of numerical QoS dimensions in the spirit of Jaeger et. al [8]. The global SLA of a business process is the cost of decomposition path of BPH. As an extension of the suggested methodology, we propose to find the “minimal” decomposition path leading to the global SLA of the business process fulfilment. At the current moment we are finalizing the methodology and working with an industrial partner on the aggregation functions design and the mathematic model justification from the business point of view.

SLA Monitoring

The results of our work [1] will be used to build a monitoring system with anticipate terms violation goal. The proposed requirements for the rules specifying warning issuing are the following: easy to compute to avoid overloading of the monitoring system and be fast to provide warnings. In addition they should provide good performance in detecting as many violations as possible generating the minimum number of false positives. The linear least squares method was chosen as a forecasting method. We have conducted preliminary experimentation to show the feasibility of the anticipate violations strategy. In the experimentation based on synthetic data, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings.

Secure Workflow Design

We address the issue of secure workflows design based on early requirements analysis, namely, SI^* /Secure Tropos [7], by presenting a methodology that bridges the gap between early requirements analysis and secure workflows for web services development. The methodology allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis. The proposed refinement methodology, aims to obtain an appropriate coarse grained secure business process that can be further refined into workflows. We introduce a specification language for secure business processes Secure BPEL, which is a dialect of WS-BPEL for the functional parts and abstracts away low level implementation details from WS-Security and WS-Federation specifications. At this point we have an open “lack of permission” problem. The “lack of permission” situation appears when there is a chain of delegation/trust of execution with no corresponding chain of delegation/trust of permission. In the Secure BPEL language both delegation and trust are modeled by invocation. In order to address the “lack of permission” problem one needs of introducing special types of invocation that allow the data to be protected, i.e., allows message confidentiality and integrity. Currently, we plan to extend the methodology to derive SLA from early requirements analysis.

References

1. Aiello, M., Frankova, G., Malfatti, D.: Whats in an agreement? An analysis and an extension of WS-Agreement. In: Proceedings of the 3rd International Conference on Service-Oriented Computing (2005)
2. Ausiello, G., Italiano, G.F., Nanni, U.: Optimal traversal of directed hypergraphs. Technical Report TR-92-073 (1992)
3. Cheng, B.H.C., Konrad, S., Campbell, L.A., Wassermann, R.: Using Security Patterns to Model and Analyze Security Requirements. In: IEEE Workshop on Requirements for High Assurance Systems (2003)
4. Frankova, G.: Web service quality composition modelling. In: Proceedings of the PhD Symposium at 3rd International Conference on Service-Oriented Computing (2005)
5. Georg, G., Ray, I., France, R.: Using Aspects to Design a Secure System. In: Proceedings of IEEE International Conference on Engineering of Complex Computer Systems, Greenbelt, Maryland, USA (December 2 - 4, 2002)
6. GGF. Web Services Agreement Specification (WS-Agreement) (September 2005)
7. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Requirements Engineering for Trust Management: Model, Methodology, and Reasoning. *International Journal of Information Security* 5(4), 257–274 (2006)
8. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: QoS Aggregation for Service Composition using Workflow Patterns. In: Proceedings of the 8th International Enterprise Distributed Object Computing Conference (2004)
9. Kazhamiakin, R., Pistore, M., Roveri, M.: A Framework for Integrating Business Processes and Business Requirements. In: Proceeding of the Enterprise Distributed Object Computing Conference (2004)
10. Keller, A., Ludwig, H.: Defining and monitoring Service Level Agreements for dynamic e-business. In: Proceedings of the 16th USENIX System Administration Conference (2002)
11. Lamanna, D.D., Skene, J., Emmerich, W.: SLAng: A language for defining Service Level Agreements. In: Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems (2003)
12. Lau, D., Mylopoulos, J.: Designing Web Services with Tropos. In: Proceedings of the IEEE International Conference on Web Services (2004)
13. Lin, M., Xie, J., Guo, H., Wang, H.: Solving QoS-driven web service dynamic composition as fuzzy constraint satisfaction. In: Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (2005)
14. Ludwig, H., Dan, A., Kearney, R.: CREMONA: an architecture and library for creation and monitoring of WS-Agreements. In: Proceedings of the Second International Conference on Service-Oriented Computing (2004)
15. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) language specification. Version 1.0. IBM Corporation (January 2003), <http://www.research.ibm.com/wsla/>
16. Molina-Jimenez, C., Pruyne, J., van Moorsel, A.: The role of agreements in IT management software. *Architecting Dependable Systems III*, pp. 36–58 (2005)
17. Molina-Jimenez, C., Shrivastava, S.K., Crowcroft, J., Gevros, P.: On the monitoring of contractual Service Level Agreements. In: Proceedings of the First IEEE International Workshop on Electronic Contracting (2004)
18. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From Stakeholder Needs to Service Requirements. In: Proceeding of International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (2006)

19. Tasic, V.: WSOL Version 1.2. Technical Report SCE-04-11, Department of Systems and Computer Engineering, Carleton University (July 2004)
20. Trienekens, J.J.M., Bouman, J.J., van der Zwan, M.: Specification of Service Level Agreements: Problems, principles and practices. *Software Quality Journal* 12(1), 43–57 (2004)
21. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: *Proceedings of the 12th International conference on World Wide Web* (2003)

Risk Management for Service-Oriented Systems

Natallia Kokash

DIT - University of Trento, via Sommarive, 14, 38050 Trento, Italy
natallia.kokash@dit.unitn.it

Abstract. Web service technology can be used for integrating heterogeneous and autonomous applications into cross-organizational systems. A key problem is to support a high quality of service-oriented systems despite vulnerabilities caused by the use of external web services. One important aspect that has received little attention so far is risk management for such systems. This paper discusses risks peculiar for service-based systems, their impact and ways of mitigation. In the context of service-oriented design, risks can be reduced by selection of appropriate business partners, web service discovery, service composition and Quality of Service (QoS) management.

Advisors. Vincenzo D'Andrea.

1 Introduction

Rapidly evolving web service technology is a key mechanism for simplifying large-scale business operations by consumption of ready-to-use services. More and more software is becoming available as web services, loosely-coupled functional entities accessible via well-defined user interfaces. These interfaces are published in registries and can be discovered by potential customers. Several web services from different providers may have to be integrated to implement real-world business information systems. Such systems, in their turn, can be available as web services for invocation by end-users or further integration.

Web service implementation details are normally hidden. Their potential clients reason about service functionalities being unaware of their internal structure. This spawns a significant challenge for those who want to use existing services as parts of new web systems. For attracting customers, these systems must be responsive, robust, and always available. They should support concurrency demands and deal gracefully with load variations. Such requirements are commonly referred to as Quality of Service (QoS). Issues related to QoS support for composite web services has found considerable research interest.

A composite web service is reactive to any changes in the behavior of constituent services. Service-Oriented (SO) systems are subject to risks caused by architectural vulnerability (e.g., technical problems, security-level threats) and by conflicting interests of the involved partners. A mechanism to support designers of service-based applications in vulnerability assessment of their systems is needed. A way of gathering the requisite data to make a good business or

technical judgement is provided by *risk management*. Among the main steps of risk management are *risk identification* (surfacing risks before they become problems), *risk analysis* (converting identified risk data into decision-making information), and *risk control* (monitoring the status of risk and actions taken to mitigate them). Regarding SO design, risk management can be applied to decide whether to entrust a part of functionality to an existing web service, which of the existing services to choose, how to protect data exchanged between partners, which additional controls must be implemented, how many alternative services are required for a particular task, and so on.

The rest of the paper is organized as follows. Section 2 provides main definitions and basic information about risk management. Section 3 discusses the distinctive features of risk analysis for SO systems. In Section 4, a high-level framework for design and reconfiguration of SO systems is proposed. The last section concludes the paper and outlines future work.

2 Software Risk Management

Risk management operates with notions of *assets* (objects of the protection efforts such as system components or data), *threats* (danger sources), *vulnerabilities* (defects in system design, implementation, or internal controls), *threat probabilities* (likelihoods that given negative events will be triggered), and their *impacts* on the organization (tangible or intangible, e.g., monetary loss or breach of reputation, law, regulation, or contract). In order to mitigate risks, *countermeasures* (management, operational, and technical controls that adequately protect the system) are applied [1].

Risk r is defined as a probability p of a threat e multiplied by a respective magnitude q of its impact: $r(e) = p(e)q(e)$. Real-world systems are unlikely to have a single risk factor. Risk of a set of independent threats can be calculated as a sum of risks for each particular threat [2]. In more general case, analysis of conditional dependencies among threats (their probabilities and impacts) is required. Often probability calculation and impact estimation are extremely rough, but still help to handle technical vulnerabilities of the system.

The goal of risk management frameworks [1][3] is to help designers to manage software projects within established time and budget constraints. Several works examine risk management for business processes [4][5]. A *business process* is a structured set of activities designed to produce a specified output for an organization or a consortium. Business processes are subject to errors in each of their components: to enable the successful completion of a business process it is important to manage the risks associated with each sub-activity. A company can rely on someone else to run certain business functions. A survey of current practices in risk management for project *outsourcing* (a formal agreement with a third party to perform a service) can be found in [6].

There exist a bulk of potential threats for cross-organizational systems. In the same time, there can be no stakeholders with a full knowledge about the system. Among the most common risk factors are tasks involving third parties, use of

Table 1. Risks in service-oriented applications

Event	Assessment	Mitigation
Loss of service	Likelihood and implications of service unavailability and network-related problems	Use for non-critical tasks and tasks that can be suspended. Use of alternative services for critical tasks. Establishment of trusted relations and service level agreements. Redesign (own code or locally deployed components).
Loss of data	Likelihood and implications of data loss	Data replication. Exchange of non-critical data. Establishment of trusted relations. Service level agreements.
Loss of users	Analysis of user expectations.	Warn users about possible problems. Provide different quality layers. Collect feedback from users.
Unexpected service behavior	Analysis of consistency and completeness of published service specifications.	Service testing. Clarification of the specifications. Service level agreements.
Specification changes	Evaluation of implications of format changes and loss of service.	Use of services on a small-scale and plan for redesign. Discovery of alternative services.
Performance problems	Service testing. Likelihood and implications of inadequate service performance	Service performance monitoring. Use for non-critical tasks. Service level agreements.
Lack of interoperability	Likelihood and implications of application lock-in and integration loss.	Evaluation of integration capabilities. Use of alternative services. Redesign.
Contract violation	Reputation of a service.	Run-time monitoring. Use of web services for non-critical tasks. Use of alternative services for critical tasks.

unfamiliar or emerging technologies, organizational problems, unclear goals, absence of quality controls and effective management. Being a state-of-the-art in business process integration techniques, SO systems are subject to the mentioned risks as well.

3 Risk Management for Service-Oriented Systems

Among a set of risks peculiar for SO systems are risks caused by service providers (disposal of a service, changes in interface and behavioral logics of a service, contract violation, obtrusion of a new contract with worse conditions, disclosure of user data) and technical aspects (network or service failures, problems with semantic interoperability). Table 1 summarizes some common threats for service-based applications. Security threats (information disclosure, spoofing and tampering, downgrade, repudiation, denial of service, etc.) and methods for their prevention are discussed in WS-Policy and WS-Security specifications.

Techniques for risk management fall into five categories: (i) *risk avoidance* involves altering the original system design to remove particularly risky elements; *risk reduction* employs methods that reduce the probability or impact of a risk occurring (e.g., use of alternative services to reduce the probability of a service loss, data replication to avoid a loss of data); (iii) *risk transfer* moves the ownership of the risk to a third party by contract (e.g., contracts stipulating penalties to web services for information disclosure); (iv) *risk deferral* entails deferring decisions to a date when a risk is less likely to happen or less severe; (v) *risk retention* implies that certain risks have to be accepted.

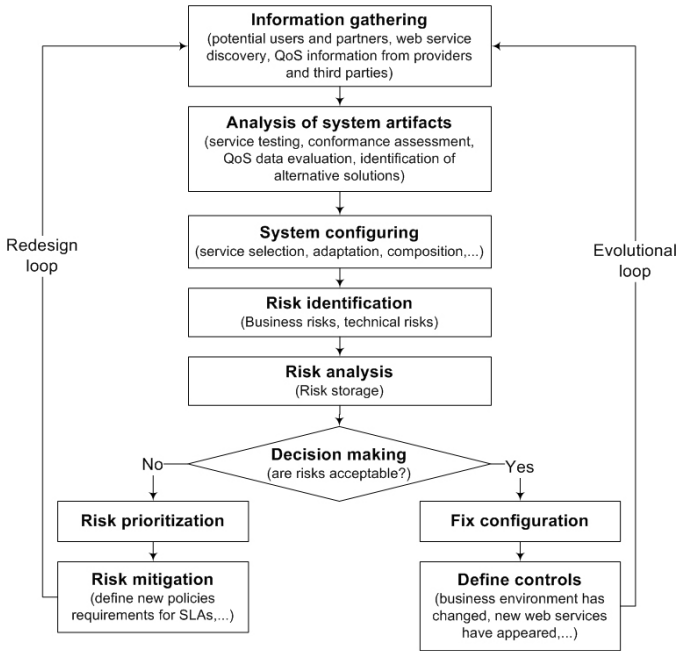


Fig. 1. Risk management framework for service-oriented systems

For every accepted risk, a designer must put controls in place that detect the corresponding event when it triggers. The idea of automatic service composition implies that service-based applications are created without (or with minimum of) human intervention: according to the principle of *late binding*, abstract service descriptions are mapped to real services on demand. Following this trend, a tool for automatic risk analysis should be developed.

4 Risk Management Framework

To enable design of reliable SO systems, an appropriate methodology and supporting tools are needed. It is the matter of risk analysis to decide whether a service-oriented computing can be applied for some business application. Zimmermann et al. [7] identify risk mitigation strategies before kicking off any premature implementation work as one of the most important lessons learnt from the project involving SO architectures, process choreography, and web services in the telecommunications industry. In our vision, risk management is a context sensitive and iterative process. Organizations have to perform systematic risk assessment both for systems being developed and for deployed systems that support runtime reconfiguration [8].

The proposed risk management framework is shown in Fig. 1. Once the business process has been modelled at the abstract level, the further analysis is

needed to gather information about potential business partners and useful web services. Principal users and their expectations should be examined and QoS information about services acquired. The latter can be provided by service owners, other clients or certification agencies. Then, the analysis of system artifacts is performed. In particular, web services discovered at the previous step must be tested and their conformance with system requirements, both functional and non-functional, assessed. At the next step, the initial system configuration is selected or the system is reconfigured. Among possible actions are: *service replacement*, which means that a component of a composite web service is changed; *structural change*, which means the systems logical structure is changed; and *geometrical change*, which means that the logical application structure remain fixed, but the physical structure (hardware, network topology, communication protocols, etc.) is changed [8]. Risks for the chosen configuration are identified using a predefined ontology of threats affecting the system.

Probabilities of threats can be estimated quantitatively based on monitored data. For example, probability of a loss of service is defined by service unavailability rate, percentile of incorrect past executions to total number of invocations defines probability of unexpected service behavior, normalized incompliance between a required web service and a discovered one can be seen as a probability of a lack of semantic interoperability. For assessing an impact of each particular threat on the system, dependency diagrams, which outline interrelationships of various functional elements, can be involved. The information about risk assessment (unique ID, description, reason, warning signs, probability, impact, timescale, cost of mitigation, expected level of risk after mitigation, etc.) is stored to allow for the comparative analysis of different system configurations.

If the overall risk level is not acceptable and/or there exist a potential for its mitigation, the designers must try to reduce it. Risk prioritization implies that the identified risks can be segmented into categories (high, medium, and low) and managed accordingly. Risk mitigation can be achieved either through system reconfiguration or through contract management. New functional requirements, such as necessity of data encryption, additional controls, etc. can be unveiled. Finally, if the overall risk is acceptable, the system configuration is fixed and the service-based application can be exploited. Predefined controls can detect important changes in the system or in the business environment and initiate an evolutionary loop when some parts of the system are replaced or adopted to meet new business requirements.

5 Conclusions and Future Work

Risk management is an essential component for SO design affecting structure of a system, eliciting requirements for service discovery, selection and contracting. It can be partially automated to enable design of reliable service-based systems using automatic service composition techniques.

In our recent work [9] a detailed description of risk-based service evaluation and selection strategies is given. In [10] an extensible monitoring service able to

collect client reports on service invocations and suggest reliable services to users with similar needs is presented. It provides a basis for gathering QoS statistics (e.g., types of possible web service exceptions) and aims at improving quality of web service discovery [11]. This will allow for the easier integration of existing functionalities into new software applications.

In our future work we are planning to elaborate the above ideas, supplementing design of service-based business processes with ability to model risks and infer necessary preventive strategies and controls.

References

1. Verdon, D., McGraw, G.: Risk analysis in software design. *IEEE Security and Privacy*, 33–37 (2004)
2. Roy, G.G.: A risk management framework for software engineering practice. In: ASWEC. Australian Software Engineering Conference, pp. 60–67. IEEE Computer Society Press, Los Alamitos (2004)
3. Freimut, B., Hartkopf, S., Kaiser, P., Kontio, J., Kobitzsch, W.: An industrial case study of implementing software risk management. In: ESEC/FSE, pp. 277–287. ACM Press, New York (2001)
4. zur Muehlen, M., Rosemann, M.: Integrating risks in business process models. In: Australasian Conference on Information Systems (ACIS) (2005)
5. Neiger, D., Churilov, L., zur Muehlen, M., Rosemann, M.: Integrating risks in business process models with value focused process engineering. In: European Conference on Information Systems (ECIS) (2006)
6. O’Keeffe, F., Vanlandingham, S.: Managing the risks of outsourcing: a survey of current practices and their effectiveness. White paper, Protiviti (2004) http://www.protiviti.com/downloads/PRO/pro-us/product_sheets/business_risk/Protiviti ORM WhitePaper.pdf
7. Zimmermann, O., Doubrovski, V., Grundler, J., Hogg, K.: Service-oriented architecture and business process choreography in an order management scenario: Rationale, concepts, lessons learned. In: OOPSLA. Conference on Object-oriented Programming, Systems, Languages and Applications, pp. 301–312. ACM Press, New York (2005)
8. Li, Y., Sun, K., Qiu, J., Chen, Y.: Self-reconfiguration of service-based systems: A case study for service level agreements and resource optimization. In: ICWS, pp. 266–273. IEEE Computer Society Press, Los Alamitos (2005)
9. Kokash, N., D’Andrea, V.: Evaluating quality of web services: A risk-driven approach. In: BIS. International Conference on Business Information Systems. LNCS, vol. 4439, pp. 180–194. Springer, Heidelberg (2007)
10. Kokash, N., Birukou, A., D’Andrea, V.: Web service discovery based on past user experience. In: BIS. International Conference on Business Information Systems. LNCS, vol. 4439, pp. 95–107. Springer, Heidelberg (2007)
11. Kokash, N., van den Heuvel, W.J., D’Andrea, V.: Leveraging web services discovery with customizable hybrid matching. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 522–528. Springer, Heidelberg (2006)

A Framework for Situational Web Methods Engineering

Sebastian Lahajnar

Pris inženiring d.o.o. Ljubljana, Slovenia
sebastian.lahajnar@pris-inz.si

Abstract. In the past ten years many web application development methods with their own or from other methods borrowed models, techniques and activities were proposed in literature. Each of these methods is appropriate for building several types of web applications on different modeling levels and project phases. It's unlikely that a single method will ever be capable to cover all aspects of web application development. The most appropriate approach for web application projects is a construction of an organization-specific base method with the use of reusable method fragments (components) and the adaptation of the base method in order to support specific project characteristics. For this purpose, the basis for an appropriate method engineering framework is proposed, which includes a process for method construction and a repository for methods, method components, configurations, rules and development situations characteristics storage.

1 Introduction

Nowadays, web applications have an important role in a development of information systems. As Kiely and Fitzgerald state [8], 33% of development projects can be classified as E-Commerce, 7% involves building internet sites, while only 25% involves traditional application development (payroll systems etc.). Comparing two empirical researches ([8], [7]) of the information systems development environment it can be noticed, that the use of methods for software development is increasing (from 40% in 1998 to 62% in 2002). Consequently, also the increasing interest in using and building methods, techniques and tools, designed specially for web application development is understandable.

In the past ten years many web application development methods based on traditional and object models, languages and processes such as ER, UML and Unified Process were proposed. With regard to their origins and purposes, they can be categorized as data, hypertext, object or process oriented [11]. The origin, initial scope and later extensions influence the usability of methods for various types of web applications (portals, transactional, document systems etc.) on different levels of modeling (conceptual, logical) and in different phases (requirements gathering, analysis, design, realization etc.). In this sense, WebML [5] is suitable for the specification of complex data intensive web applications at the conceptual level, SOHDM [14] emphasizes a process driven web applications development, Conallen [6] proposes a development process based on RUP and advocates the importance of modeling concrete products in

the design phase with the use of UML WAE profile, UWE [13] focuses on the navigation and presentation aspects of web applications, the primary purpose of WebSA [15] method is to define the architectural view of web applications etc.

2 The Problem Domain

From the analysis of current web development methods it is possible to deduce, that their activities and models are well suited for describing navigational, presentational and architectural views on different levels. For the description of data view at the conceptual level, web development methods use traditional models and techniques such as ER model and UML class diagram. Web development methods mostly don't concern with activities for business modeling, project management, testing, maintenance, portfolio management etc., which are part of traditional development processes. On the other hand, traditional development processes and modeling languages do not comprehend adequate activities and modeling techniques for the description of web applications particularity, such as navigation and presentation, and particularity of other modeling domains as for example business process modeling and data modeling. The solution is provided by number of existing UML profiles and other modeling techniques, which can be included in a development process in order to improve a reliability of developed products.

Web application development in the context of current organization's characteristics, complexity of the solution, the size of the project and other factors, may include a number of more or less important activities with the final goal to deploy a working web application. The structure, activities and produced artifacts must be adapted to concrete situations what means, that the method can not be fully defined in advance. Nevertheless, it is recommended to develop a base organization-specific method, founded on experiences acquired in previous projects, good practices and organization culture, which can be used as a reference for a construction of adapted methods for specific project situations. A base method as all other adapted methods must be composed from a set of best, autonomous and reusable fragments, which will provide good foundations for successful project progress in given circumstances. Since such fragments belong to different software development methods and processes, they have to be re-engineered in order to be successfully included in the new method in construction. Following the reflection in previous paragraphs, two theses are defined: 1) it's unlikely that a single method will ever be capable to cover all aspects of web application development. 2) the most appropriate approach for web application development projects is a construction of organization-specific base method on the basis of reusable method fragments stored in a repository, which is then tailored to support specific project needs.

3 The Aims and Objectives of the Research

The main objective of the research is to build a common framework for web application method engineering. In this context, many traditional and new approaches for situational method engineering have been already investigated. The effort resulted in

the definition of the framework foundations with the corresponding process of re-engineering parts of widely recognized methods in method fragments, constructing base method with method fragments assembly and tailoring base method to project specific demands.

The proposed framework is supposed to include the most important activities, models and techniques which nowadays are used in information systems development, as also their extensions in the field of web engineering, business processes and databases, with the emphasis on modern object oriented development processes and the UML as a main modeling language.

4 The Solution

The discipline to design, construct and adapt methods, techniques and tools for information systems development is called method engineering, while its specific direction, situational method engineering, deals with a development of specific methods, tailored for concrete organization or project circumstances [2]. The construction of new methods is founded on standardized assets called method fragments, defined as coherent pieces of information systems development methods. Brinkkemper [3] classifies method fragments according to three dimensions: perspective (product, process), abstraction (conceptual, technical) and granularity (method, stage, model, diagram and concept). The process and the product view of the method fragment are inseparable interrelated as products are both results and inputs of the method's prescribed actions.

Instead of method fragments, Wistrand [19] proposes the concept of method component, defined as a self-contained part of the system engineering method expressing the process of transforming input artifacts into defined target artifacts and a rationale for such a transformation. In this way, a method component is a special case of a method fragment with the following characteristics: self-contained, internal consistency and coherency, rationality and applicability. Two views describes each method component: the internal view addresses the internal structure of the component while the external view focuses on its relationships with other components which all together contribute to the overall method goal. For our purpose, the concept of method component have been chosen as the foundation for the development of the web methods engineering framework as it emphasizes higher level of granularity, fragments independence and fragments interconnection through interfaces.

The process of building the method base is based on the re-engineering of existing methods or their individual fragments and includes the following steps [17]: defining or reconstructing the initial method process model, identifying method components and defining method components. The construction of the method base is a continuous process at the beginning of which the method base is populated with an initial set of method components. The initial set is further enhanced during the process of building the base method and it's adaptation to specific projects.

After the initial set of method components has been defined, the construction of the base method takes place with the use of the assembly strategy following the next three steps [18]: project characterization, method fragments selection and method fragments assembly. Two basic strategies are defined for method fragments assembly

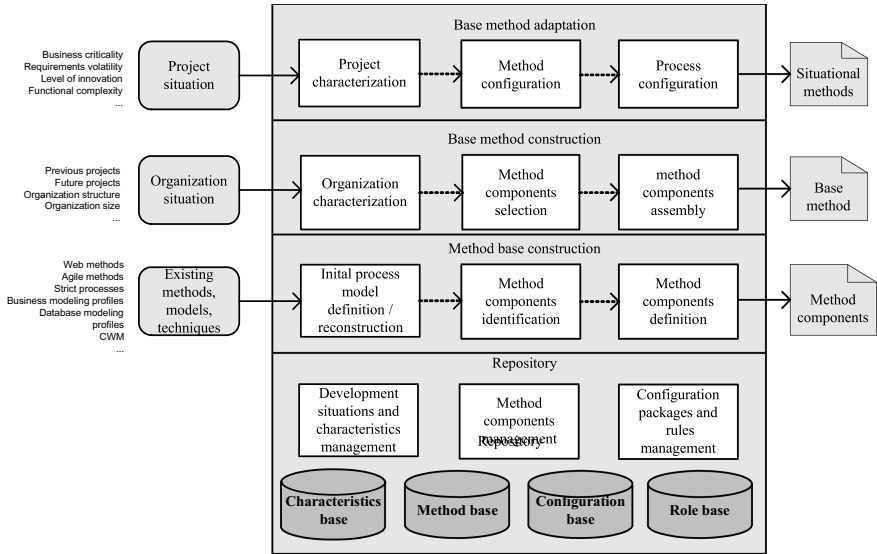


Fig. 1. The structure of the framework

[17]: association and integration. The association strategy is appropriate in a case, when assembled method fragments don't have common elements, while the integration strategy is relevant for the assembly of methods with common engineering objectives, but different ways to achieve them.

The combination of two strategies, method configuration [12] and process configuration [1] respectively, has been chosen for the adaptation of the base method to specific projects. The reusable concepts, a configuration package and a configuration template form the foundation of this strategy. The configuration package is defined as a configuration of the base method suitable for one specific characteristic's value and the configuration template is defined as a combined method configuration for a set of recurrent project characteristics [12]. The process configuration strategy supplements method configuration by enabling the definition of alternative paths in the context of configuration packages. The process configuration is performed on the basis of a set of rules for structural and process restrictions, depending on the specific circumstances.

5 Related Work

Works resembling the framework proposed in this paper comes from two disciplines: web and method engineering. In the area of web engineering, Koch [13] defines a metamodel for UWE approach, which describes the concepts of the navigation and presentation model. This and other proposed metamodels for various web methods could be seen as a foundation for one or more method components, focusing on the navigational and presentational characteristics of the web application. Caceres [4] advocates MIDAS, a model-driven methodology for web IS development in which he

incorporates several models and diagram technique from other web and non web methods for structural PIMs and PSMs. Moreno [16] on the other hand, distinguishes three main PIMs: user interface, business logic and data. For each of them, he proposes a set of feasible models with appropriate concepts. Both approaches share many similarities with the proposed framework, as many different models and technique from different fields are used, and there's also some possibility to tailor the two methods to suit the project's specific needs. However, the framework proposed in this paper takes a broader perspective to web application development as it enables the integration of almost any desired and useful technique, model, concept, activity or whole method, as long as it's adequately represented following the rules of the method components metamodel (internal and external view).

From the sphere of method engineering, the OPF framework [9] should be mentioned, which besides the common set of elements for the software development process, incorporates also some web specific activities, tasks and roles. Nevertheless, OPF doesn't address adequately all web applications specific aspects such as navigation and presentation and furthermore, for each project a completely new instance of the OPF metamodel has to be instantiated, what increases project overall cost and duration.

6 Future Work and Conclusions

A proper metamodel for the method base as a combination of many existing metamodels (method components [19], SMSDM [10], method configuration [12] etc.) is currently under development. After that an already identified set of valuable activities, work products, models and techniques from different software engineering fields (web, database, business process etc.) will be further re-engineered in method components in concordance to the metamodel specified. Based on the developed set of method components, a new organization-specific method for web application development will be constructed with the application of the assembly strategy on the basics of defined components interfaces. Finally, the proposed framework and the base method will be validated on the concrete project, involving a development of many web portals for different faculties in Slovenia which is already in progress at the time of this writing. The new method for web application development will be evaluated against the current approach on the basis of a set of predefined criteria, using a questionnaire and performing interviews with project team members.

References

1. Bajec, M., Vavpotič, D., Krisper, M.: An approach for creating project-specific software development methodologies. In: *Internet and information technology in modern organizations*, IBIMA, pp. 1–15 (2005)
2. Brinkkemper, S.: Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology* 38(4), 275–280 (1996)
3. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based assembly techniques for situational method engineering. *Information Systems* 24 (1999), 209–228 (1999)
4. Cáceres, P., Marcos, E., Vela, B.: A MDA-Based Approach for Web Information System. In: *Workshop in Software Model Engineering, WisME* (2004)

5. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing* 6(4), 20–30 (2002)
6. Conallen, J.: *Building Web Applications with UML*, 2nd edn. Addison-Wesley, London, UK (2003)
7. Fitzgerald, B.: An empirical investigation into the adoption of systems development methodologies. *Information and Management* 34(6), 317–328 (1998)
8. Kiely, G., Fitzgerald, B.: An investigation of the information systems development environment: the nature of development life cycles and the use of methods. In: *Eighth Americas Conference on Information Systems*, Baylor, pp. 1289–1296 (2002)
9. Haire, B., Lowe, D., Henderson-Sellers, B.: Supporting Web development in the OPEN process: Additional roles and techniques. In: Bellahsene, Z., Patel, D., Rolland, C. (eds.) *OOS 2002. LNCS*, vol. 2425, pp. 82–94. Springer, Heidelberg (2002)
10. Henderson-Sellers, B., Gonzalez-Perez, C.: A comparison of four process metamodels and the creation of a new generic standard. *Information & Software Technology* 47(1), 49–65 (2005)
11. Kappel, G.: *Web Engineering, The Discipline of Systematic Development of Web Applications*. John Wiley & Sons, West Sussex, England (2006)
12. Karlsson, F., Ågerfalk, P.J.: Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology* 46, 619–633 (2004)
13. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. In: Lovelle, J.M.C., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) *ICWE 2003. LNCS*, vol. 2722, pp. 497–506. Springer, Heidelberg (2003)
14. Lee, H., Lee, C., Yoo, C.: A scenario-based object-oriented methodology for developing hypermedia information systems. In: Sprague, R. (eds.) *Proceedings of 31st Annual Conference on Systems Science* (1998)
15. Meliá, S., Gómez, J., Koch, N.: Improving Web Design Methods with Architecture Modeling. In: Bauknecht, K., Pröll, B., Werthner, H. (eds.) *EC-Web 2005. LNCS*, vol. 3590, pp. 53–64. Springer, Heidelberg (2005)
16. Moreno, N., Vallecillo, A.: A Model-based Approach for Integrating Third Party Systems with Web Applications. In: Lowe, D.G., Gaedke, M. (eds.) *ICWE 2005. LNCS*, vol. 3579, pp. 441–452. Springer, Heidelberg (2005)
17. Ralyte, J., Rolland, C.: An Assembly Process Model for Method Engineering. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) *CAiSE 2001. LNCS*, vol. 2068, Springer, Heidelberg (2001)
18. Ralyté, J., Rolland, C.: An approach for method reengineering. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) *ER 2001. LNCS*, vol. 2224, Springer, Heidelberg (2001)
19. Wistrand, K., Karlsson, F.: Method Components - Rationale Revealed. In: *The 16th International Conference on Advanced Information Systems Engineering*, Riga, Latvia (2004)

Author Index

- Acerbis, Roberto 501, 539
Aedo, Ignacio 269
Ahmed, Asad 516
Alalfi, Manar H. 306
Amezcuca, Omar 526
Anfurrutia, Felipe I. 473
Ardissono, Liliana 47
Aroyo, Lora 328
Arrue, Myriam 370
- Bae, Jeong Seop 53
Bellekens, Pieter 328
Benatallah, Boualem 1, 479
Benbernou, Salima 1
Bertolino, Antonia 17
Betermieux, Stefan 137
Bieliková, Mária 511
Bomsdorf, Birgit 137
Bongio, Aldo 501, 539
Book, Matthias 167
Borodin, Yevgen 516
Bova, Rosanna 1
Bozzon, Alessandro 210
Brambilla, Marco 312, 501, 533, 539
Butti, Stefano 501
- Cabot, Jordi 59, 533
Cachero, Cristina 74
Calero, Coral 74
Canfora, Gerardo 457
Casati, Fabio 479
Castanedo, Raúl Izquierdo 442
Ceballos, Jordi 59
Celino, Irene 485
Ceri, Stefano 539
Cerizza, Dario 485
Ceruti, Marion G. 526
Chang, Soo Ho 53
Comai, Sara 364
Cordy, James R. 306
- Daniel, Florian 479
Das, Joydip 545
Dattolo, Antonina 421
Davulcu, Hasan 385
- De Angelis, Guglielmo 17
De Lucia, Andrea 415
De Silva, Buddhima 248, 521
Dean, Thomas R. 306
Della Valle, Emanuele 485
Di Iorio, Angelo 421
Díaz, Oscar 473
Díaz, Paloma 269
Distante, Damiano 152, 457
Dolog, Peter 32, 358
Duca, Silvia 421
Duffy, LorRaine 526
- Ebert, Jürgen 194
- Facca, Federico M. 312
Feliziani, Antonio Angelo 421
Fialho, André T.S. 188
Frankova, Ganna 556
Fuente, Aquilino A. Juan 442
Fugate, Sunny 526
Furnari, Roberto 47
- Gaedke, Martin 427
García, Francisco J. 442, 550
Gelgi, Fatih 385
Giles, C. Lee 121, 285
Ginige, Anupama 521
Ginige, Athula 248, 521
Ginzburg, Jeronimo 152
Gipp, Torsten 194
Gómez, Cristina 59
Gómez, Jaime 491
Goy, Anna 47
Gruhn, Volker 167
Guerra, Esther 269
- Hassas, Salima 1
Houben, Geert-Jan 328
Huang, Tao 105
- Iofciu, Tereza 210
- Jatowt, Adam 343

- Kim, Soo Dong 53
 Kimura, Rui 400
 Kokash, Natallia 563

 La, Hyun Jung 53
 Lahajnar, Sebastian 569
 Lee, Wang-Chien 121
 Leporini, Barbara 254
 Li, Huaqing 121
 Li, Yang 105
 Liang, Xufeng 521
 Linaje, M. 226

 Magsombol, Dennis 526
 Mahmud, Jalal 516
 Majer, Frederic 427
 Marmaridis, Makis 521
 Matera, Maristella 479
 Mecca, Giansalvatore 496
 Medina, Emily W. 526
 Meinecke, Johannes 427
 Meliá, Santiago 491
 Mendes, Emilia 90
 Mesnage, Cédric 506
 Miao, Huaikou 301
 Mitra, Prasenjit 285
 Moreno, Nathalie 533
 Mori, Giulio 182

 Nakamura, Satoshi 343
 Nassi, Ike 545
 Nejd, Wolfgang 32, 210

 Oren, Eyal 506
 Oyama, Satoshi 400

 Paik, Hye-Young 1
 Pappalardo, Alessandro 496
 Paternò, Fabio 182, 254
 Pedone, Paola 457
 Pelechano, Vicente 242
 Petrone, Giovanna 47
 Poels, Geert 74
 Polini, Andrea 17
 Preciado, Juan C. 226

 Ramakrishnan, I.V. 516
 Raunich, Salvatore 496
 Richter, Jan 167

 Risi, Michele 415
 Rodriguez Priego, Emilio 550
 Rogers, Gary 526
 Rossi, Gustavo 152, 457

 Saint-Paul, Regis 479
 Sánchez-Figueroa, F. 226
 Santoro, Carmen 182
 Santoro, Donatello 496
 Sanz, Daniel 269
 Scanniello, Giuseppe 415
 Schäfer, Michael 32
 Schwabe, Daniel 188
 Scordia, Antonio 254
 Segnan, Marino 47
 Serrano, José Luís 491
 Shan, Ming-Chien 545
 Sivasubramaniam, Anand 121
 Stage, Jan 358

 Tan, Qingzhao 285
 Tanaka, Katsumi 343, 400
 Tisi, Massimo 539
 Toffetti Carughi, Giovanni 364
 Toda, Hiroyuki 400
 Tönnies, Sascha 210
 Tortora, Genoveffa 415
 Tosetti, Emanuele 539
 Trujillo, Salvador 473
 Turati, Andrea 485
 Tvarožek, Michal 511

 Urbietta, Matias 152

 Vadrevu, Srinivas 385
 Valderas, Pedro 242
 van der Sluijs, Kees 328
 Vigo, Markel 370
 Vitali, Fabio 421

 Wei, Jun 105

 Yanbe, Yusuke 343
 Yu, Jin 479

 Zeng, Hongwei 301
 Zhong, Hua 105
 Zhuang, Ziming 285
 Zuo, Lin 105