Workshop Proceedings

Editors: Marco Brambilla, Emilia Mendes

Workshops

AEWSE'07: workshop on Adaptation and Evolution in Web Systems Engineering
AWSOR'07: workshop on Aligning Web Systems and Organisation Requirements
IWWOST'07: 6th workshop on Web-Oriented Software Technologies
MDWE'07: 3rd workshop on Model-driven Web Engineering
WQVV'07: workshop on Web quality, Verification and Validation
ICWE’07 Workshops, Como, Italy, July 2007 – Marco Brambilla, Emilia Mendes (Eds.)
Workshop proceedings of the 7th International Conference on Web Engineering.
Copyright © 2007 held by authors.

Workshop list

**AEWSE’07: workshop on Adaptation and Evolution in Web Systems Engineering**
Organizers: S. Casteleyn, F. Daniel, P. Dolog, M. Matera, G.-J. Houben, O. De Troyer

**AWSOR’07: workshop on Aligning Web Systems and Organisation Requirements**
Organizers: D. Lowe, D. Zowghi

**IWWOST’07: 6th workshop on Web-Oriented Software Technologies**
Organizers: M. Winckler, O. Pastor, D. Schwabe, L. Olsina, G. Rossi

**MDWE’07: 3rd workshop on Model-driven Web Engineering**
Organizers: A. Vallecillo, N. Koch, G.-J. Houben

**WQVV’07: workshop on Web quality, Verification and Validation**
Organizers: M. A. Moraga, C. C. Munoz, M. A. Caro Gutierrez, A. Marchetto, A. Trentini, T. Bultan

Sponsors


Publisher:
Dipartimento di Elettronica e Informazione,
Politecnico di Milano.
June 2007.
Milano, Italy.
7th International Conf. on Web Engineering

Como, Italy, July 2007

ICWE 2007 Workshops

Editors
Marco Brambilla, Politecnico di Milano (Italy)
Emilia Mendes, University of Auckland (New Zealand)

Copyright © 2007 held by authors
Preface

Web Engineering is a young discipline containing numerous research challenges being investigated by its research community, some of which more pressing than others. Of these, two of the most important relate to the development and maintenance of Web applications since the frequent pressure of time to market faced by Web companies must co-exist with the delivery of high quality Web applications. To tackle these challenges Web engineering research must inform practice with the necessary mechanisms to enable practitioners to:

- Obtain and fully understand user requirements very early on in the development life cycle
- Design and implement applications using the best set of available models/tools/techniques, the least amount of time, and providing the best possible quality. Quality here is often characterised by high usability and reliability.

It is pleasing to see that this year the five workshops to take place during the 7th International Conference on Web Engineering together encompass the mechanisms abovementioned, thus representing a wide and complementary spectrum of research related to Web development and maintenance.

These five workshops and corresponding organisers are as follows:

- **AEWSE'07** - Second International Workshop on Adaptation and Evolution in Web Systems Engineering (Organisers: Sven Casteleyn, Florian Daniel, Peter Dolog, Maristella Matera, Geert-Jan Houben, Olga De Troyer)
- **IWWOST'07** - Sixth International Workshop on Web-Oriented Software Technologies (Organisers: Marco Winckler, Oscar Pastor, Daniel Schwabe, Luis Oslinal, Gustavo Rossi)
- **MDWE'07** - Third International Workshop on Model-Driven Web Engineering (Organisers: Antonio Vallecillo, Nora Koch, Geert-Jan Houben)
- **AWSOR'07** - 1st Workshop on Aligning Web Systems and Organization Requirements (Organisers: David Lowe, Didar Zowghi)
- **WQVV'07** - First Workshop on Web quality, Verification and Validation (Organisers: Maria Angeles Moraga, Coral Calero Munoz, Maria Angélica Caro Gutierrez, Alessandro Marchetto, Andrea Trentini, Tevfik Bultan)

We received nine workshop submissions of which six were selected. Of these, two were joined, making it a total of five workshops to take place during ICWE’07.

I would like to thank the workshops’ organisers for their excellent work putting together a very exciting selection of papers and invited talks. And of course, to thank all the authors who submitted papers to the workshops.

I hope you all find the workshops thought provoking and engaging, and that the discussions and results presented make a strong contribution to the body of knowledge in Web Engineering and to advancing this discipline further.

June 2007

Emilia Mendes
**Table of Contents**

**AEWSE’07**  
Second International Workshop on Adaptation and Evolution in Web Systems Engineering  
(Organisers: Sven Casteleyn, Florian Daniel, Peter Dolog, Maristella Matera, Geert-Jan Houben, Olga De Troyer) ................................................................. 7

**AWSOR’07**  
First Workshop on Aligning Web Systems and Organization Requirements  
(Organisers: David Lowe, Didar Zowghi) ................................................... 109

**IWWOST’07**  
Sixth International Workshop on Web-Oriented Software Technologies  
(Organisers: Marco Winckler, Oscar Pastor, Daniel Schwabe, Luis Olsinal, Gustavo Rossi) ................................................................. 147

**MDWE’07**  
Third International Workshop on Model-Driven Web Engineering  
(Organisers: Antonio Vallecillo, Nora Koch, Geert-Jan Houben) .......... 209

**WQVV’07**  
First Workshop on Web quality, Verification and Validation  
(Organisers: Maria Angeles Moraga, Coral Calero Munoz, Maria Angélica Caro Gutierrez, Alessandro Marchetto, Andrea Trentini, Tevfik Bultan) .... 315
Second International Workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE’07)
July 19, 2007 – Como, Italy

Organisers
Sven Casteleyn (Vrije Universiteit Brussel, Belgium)
Florian Daniel (Politecnico di Milano, Italy)
Peter Dolog (Aalborg Universitet, Denmark)
Maristella Matera (Politecnico di Milano, Italy)
Geert-Jan Houben (Vrije Universiteit Brussel, Belgium, Technische Universiteit Eindhoven, The Netherlands)
Olga De Troyer (Vrije Universiteit Brussel, Belgium)

Program Committee Members
Jaime Gomez (University of Alicante, Spain)
Nora Koch (Ludwig-Maximilian University München, Germany)
Gustavo Rossi (Universidad Nacional de La Plata, Argentina)
Schahram Dustdar (Technical University of Vienna, Austria)
Peter Plessers (Vrije Universiteit Brussel, Belgium)
Jeen Broekstra (Technische Universiteit Eindhoven, The Netherlands)
Moira Norrie (ETH Zurich, Switzerland)
Preface

Current Web applications are evolutionary in their nature: in several scenarios, such class of systems require (frequent) changes of content, functionality, semantics, structure, navigation, presentation or implementation. Examples of such applications are found in domains as eHealth, eGovernment, eLearning, and Business to Business interactions such as open marketplaces. In all these domains, the Web enables to do business or professional activities on the Internet. However, application services change over time due to new knowledge, practices, processes, and management approaches in the application domains. Moreover, recent advances in communication and network technologies provide users the ability to access content with different types of (mobile) devices, at any time, from anywhere, and with any media service.

Such new needs demand for the development of adaptive Web systems, able to support more effective and efficient interactions in all those situations where the contents and services offered by the Web application are (rapidly) changing, and/or strongly depend on the current environmental situation, users’ (dis)abilities, and/or the actual purpose of the application.

Due to the changes in the application domains, the structure, navigation and presentation of Web applications, the content and its semantics are typically highly volatile, and evolve due to a variety of reasons, such as:

- Changes to the design of the application (e.g. to correct design flaws, or to support new requirements);
- Adaptation to new technologies;
- Changes to maintain consistency with (changing) external sources (e.g. a referenced ontology, externally linked pages);
- Update/change (by the user) of for example content, structure, navigation, presentation (e.g. relevant with the rise of blogs, wikis, etc.);
- Maintenance.

Properly dealing with evolution will clearly influence the quality of a Web site (i.e. incorrect linking due to changes, unreachable pages and their automatic repair, consistency, etc). Similarly, mechanisms to automatically deal with evolution and its consequences will become indispensable in large-scale Web applications (where manual management of changes and their impact is infeasible). Also, when ontologies are used to describe or annotate content on Web sites, their evolution must be managed to avoid any inconsistency between the ontologies and the Web sites.

Although highly relevant due to the intrinsic evolutionary nature of Web applications, the problem of dealing with adaptation and evolution of Web applications (during design, implementation and deployment) and its impact is highly under-estimated; so far few works dealt with adaptation and evolution in Web Engineering research.
AEWSE aims at promoting the discussion on the above issues, bringing together researchers and practitioners with different research interests and belonging to different communities. This year’s edition of AEWSE, held in conjunction with ICWE 2007 (Como, Italy), received a good number of submissions, among which 10 papers have been selected for presentation. The addressed topics mainly focus on:

- Model-driven engineering approaches for adaptive and context-aware web applications (see papers by Garrigós et al., Daniel et al., and Reina-Quintero et al.).
- The benefit and application of semantic Web and Web 2.0 tools and technologies (Preciado et al, Barla et al.)
- Web User interface migration (Bandelloni et al.).
- The application of software engineering techniques, such as aspect orientation (Bebjak et al., Mondéjar et al.) and object variance/versioning (Grossniklaus et al.), for supporting adaptivity and evolution.

Starting from these contributions and the invited talk by Prof. Barbara Pernici (Politecnico di Milano), our ultimate goal is to facilitate the discussion of key issues, approaches, open problems, innovative applications, and trends in the methodologies and technologies to support adaptive access to and/or evolution in (the design of) Web applications.

June 2007

Sven Casteleyn
Florian Daniel
Peter Dolog
Maristella Matera
Geert-Jan Houben
Olga De Troyer
# Table of Content

Irene Garrigos, Cristian Cruz and Jaime Gomez. A Prototype Tool for the Automatic Generation of Adaptive Websites ......................................................... 13

Florian Daniel, Maristella Matera, Alessandro Morandi, Matteo Mortari and Giuseppe Pozzi. Active Rules for Runtime Adaptivity Management ..................... 28

Michael Grossniklaus and Moira Norrie. Using Object Variants to Support Context-Aware Interactions ........................................................................... 43

Renata Bandelloni, Giulio Mori, Fabio Paternò, Carmen Santoro and Antonio Scorcia. Web User Interface Migration through Different Modalities with Dynamic Device Discovery ......................................................................................... 58

Rubén Mondéjar, Pedro Garcia Lopez, Carles Pairot and Antonio F. Gómez Skarmeta. Adaptive Peer-to-Peer Web Clustering using Distributed Aspect Middleware (Damon) ............................................................................................ 73

Michal Bebjak, Valentino Vranic and Peter Dolog. Evolution of Web Applications with Aspect-Oriented Design Patterns ...................................................... 80

Michal Barla, Peter Bartalos, Mária Bieliková, Roman Filkorn and Michal Tvarozek. Adaptive portal framework for Semantic Web applications ................. 87

Juan Carlos Preciado, Marino Linaje Trigueros, Fernando Sánchez Figueroa. An approach to support the Web User Interfaces evolution ............................. 94

Antonia M. Reina-Quintero, Jesús Torres Valderrama and Miguel Toro Bonilla. Improving the adaptation of web applications to different versions of software with MDA .......................................................... 101
A Prototype Tool for the Automatic Generation of Adaptive Websites

Irene Garrigós, Cristian Cruz and Jaime Gómez

1 Universidad de Alicante, IWAD, Campus de San Vicente del Raspeig, Apartado 99 03080 Alicante, Spain
{igarrigos, ccruz, jgomez}@dlsi.ua.es

Abstract. This paper presents AWAC, a prototype CAWE tool for the automatic generation of adaptive Web applications based on the A-OOH methodology. A-OOH (Adaptive OO-H) is an extension of the OO-H approach to support the modeling of personalized Websites. A-OOH allows modeling the content, structure, presentation and personalization of a Web Application. The AWAC tool takes the A-OOH design models of the adaptive Website to generate as an input. Once generated, the adaptive Website also contains two modules for managing the personalization which, at runtime, analyze the user browsing events and adapt the Website according to the personalization rule(s) triggered. These personalization rules are specified in an independent file so they can be updated without modifying the rest of the application logic.

1 Introduction

The continuous evolution of the WWW is reflected in the growing complexity of the Web applications with rapidly changing information and functionality. This evolution has lead to user disorientation and comprehension problems, as well as development and maintenance problems for designers. The ad-hoc development of Web-based systems lacks a systematic approach and quality control. Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. In this context Web Design Methodologies appeared [3,4,11,12,14,16], giving solutions both for designers and for users. However, new challenges appeared, like the need of continuous evolution, or the different needs and goals of the users. Adapting the structure and the information content and services for concrete users (or for different user groups) tackles the aforementioned (navigation, comprehension and usability) problems. In order to better tailor the site to one particular user, or a group of users, several Web methodologies provide (to some extent) personalization or adaptation support. Yet, few of these approaches provide an underlying CAWE1 tool for Web Engineering and even less provide a tool to support personalization modeling (see next section for an overview). The lack of

1 Computed Aided Web Engineering
such tools causes the personalization to be implemented in an ad-hoc manner. Moreover, the different (adaptive) methodologies are not properly tested.

In this paper, we present the fundamentals of the AWAC (Adaptive Web Applications Creator) prototype tool. This is a CAWE tool which automatically generates adaptive Web applications based on the A-OOH methodology developed at the University of Alicante (Spain). The input of the AWAC tool is the set of A-OOH design models needed to model the adaptive Website to generate. The output is the generated Website with its database. The output includes a Web engine and a personalization module which allow the adaptivity at runtime of the final Web pages. The paper is structured as follows. In the next section, a study of the existing methodologies with a CAWE tool supporting adaptivity is presented. In Section 3 a brief introduction to the A-OOH method is given. The paper continues in Section 4 describing the AWAC architecture and some of the technologies used. This section also describes the personalization support in AWAC. Along the Sections 5, 6, and 7 the different steps for creating and running an adaptive Website using the AWAC tool are explained. A running example (online library) is used to describe the tool support. Finally, Section 8 sketches some conclusions and further work.

2 Related Work

As aforementioned, few (adaptive) Web modeling approaches provide an underlying CAWE tool to give support to their methodologies. We can mention the Hera Presentation Generator (HPG) [9], which is the integrated development environment that supports the Hera methodology developed at the Technical University of Eindhoven (The Netherlands). There are two versions of HPG: HPG-XSLT and HPG-Java. Compared with HPG-XSLT, HPG-Java extends the functionality of a generated Website with user interaction support (form-based). Moreover, instead of generating the full Web presentation like HPG-XSLT does, HPG-Java generates one-page-at-a-time in order to better support the dynamic Web applications. The designer can define adaptation by means of the inclusion of appearance conditions over the elements of the Hera design models. These conditions are expressed in SerQL[15] language and use data from the user/platform profile or conceptual model. A drawback of this approach is the difficult maintenance when the personalization policies change because the conditions are integrated in the models.

Another tool for generating adaptive hypermedia applications on the WWW is AHA! [7] (Adaptive Hypermedia Architecture), based on the AHAM model. It also has been developed at the Eindhoven University of Technology (The Netherlands). It is able to perform adaptation that is based on the user’s browsing actions. AHA! provides adaptive content by conditionally including page fragments, and adaptive navigation support by annotating (actually coloring) links. The updates to attributes of concepts in the user model are done through event-condition-action rules. Every rule is associated with an attribute of a concept, and is "triggered" whenever the
value of that attribute changes. Every page has an *access* attribute which is (virtually) "changed" whenever the end-user visits that page. This triggers the rules associated with this attribute. The AHA! tool claims to be general purpose but has mainly been used to develop on-line courses.

We can also mention WebRatio [17] developed to support the WebML methodology at the Politecnico di Milano (Italy). This tool still does not support dynamic personalization features, but only adaptability (with respect to user profile/preferences and device characteristics). To validate WSDM, at the University of Brussels (Belgium), a prototype tool was created for the support of the methodology [2]. It does not support personalization, but adaptivity for all the users. ArgoUWE [13] is the tool developed to support the UWE approach at the Luwdig Maximilians University of Munich (Germany). UWE supports personalization however it is not yet incorporated on the ArgoUWE tool.

3 A-OOH Fundamentals

The Adaptive OO-H method (A-OOH) is an extension of the OO-H (Object Oriented Hypermedia) approach [11] to support the modeling of adaptive (and personalized) Web applications. It supports most of the OO-H basic features and some updates and extensions for the support of adaptive Web sites modeling.

The same as OO-H, A-OOH is a user-driven methodology based on the object oriented paradigm and partially based on standards (XML, UML, OCL...). The approach provides the designer the semantics and notation needed for the development of adaptive Web-based interfaces and their connection with pre-existing application modules. The main differences with respect to OO-H are next:

- Adaptive hypermedia systems are complex systems which require an appropriate software engineering process for their development. This is why in A-OOH the design process is based on the Unified Process (UP) and not in the spiral model as OO-H design process was based.
- The Navigational Model has been modified separating the presentation features that were mixed in the Navigational Model of OO-H. Moreover a UML profile has been defined for using UML notation for representing the Navigational Model.
- A Presentation Model has been added. This model also uses UML notation.
- A User and Personalization Model have been added for being able of modeling adaptive Web applications.

The set of A-OOH models are defined for the running example in Sections 5 and 6. The personalization model allows the designer to define a collection of rules that can be used to define a personalization strategy for a user or group of users. The rules are Event-Condition-Action [5] rules: they are triggered by a certain event (e.g. a browsing action, the start of a session) and if a certain condition is fulfilled.
(for example “user.age=18”), the associated action is performed. The rules will be defined using a simple and easy to learn language defined in A-OOH. One of the purposes of this language is to help the Web designers\(^2\) to define all the rules required to implement a personalization strategy. This language is called PRML (Personalization Rules Modelling Language) and will be shortly explained in Section 4.2. Next, the fundamentals of AWAC are explained.

### 4 AWAC: Adaptive Web Applications Creator

The AWAC tool main purpose is automatically generating an adaptive Web application from the A-OOH models. The AWAC tool takes as input the A-OOH design models: the **Domain Model**\(^3\) (DM), in which the structure of the domain data is defined, the **Navigation Model** (NM) which defines the structure and behaviour of the navigation view over the domain data, and finally the **Presentation Model** (PM) defines the layout of generated hypermedia presentation. To be able to model adaptation/personalization at design time two new models are added (to the set of models): The **User Model** (UM), in which the structure of information needed for personalization is described. Typically, the information captures beliefs and knowledge the system has about the user and it is a foundation for personalization actions described in the Personalization Model. The **Personalization Model** (PeM), in which personalization policies are specified. Next to the personalization of the content, navigation structure and presentation, the personalization model also defines updates of the user information specified in the User Model. These models are represented by means of XML elements (in XMI [18] format).

The reason for choosing an XMI representation of the models is that this format can be easily generated from UML models (most UML tools allow this transformation). To read and process the A-OOH models for the generation of the final Web pages we have used the .NET technology. This technology provides us with the DOM class (XML Document Object Model), with which we can represent in memory the XML documents. The output of the AWAC tool is:

- **The generated adaptive Website (Web pages):** the actual version of AWAC generates ASP.net Web pages.
- **Modules for managing the personalization:** these modules are explained in the next section.
- **Application database:** The A-OOH models initially represented in XMI models are mapped into an object oriented database. Depending on the personalization actions performed every user has a different set of A-OOH model instances. This database also contains the user information related to the domain. The idea of using a relational database was rejected due to the transitioning complexity from object-oriented thinking to relational persistence. In this way the database can automatically be generated from the set of A-OOH models. The database technology we chose is db4o [6], the most

\(^2\) Web designers are not necessarily experienced Web programmers.

\(^3\) Sometimes called Conceptual Model
popular database for objects of open code, native to Java and .NET. Db4o eliminates the OO-versus-performance tradeoff: it allows you to store even the most complex object structures with ease, while achieving highest level of performance.

4.1 Generated AWAC Application Architecture

The generated Web Application has a three layered architecture as seen in fig. 1a.

- The first layer is the user interface, through which the user can generate http requests and receive http responses.
- The second layer contains the main modules of the Web Application for managing the personalization (i.e. Website Engine and PRML Evaluator, see fig 1b). The Website Engine interacts with the user, gathering the requests and giving back the response. As already mentioned, the A-OOH models are modified for each particular user (i.e. each user will have a different set of models depending on the adaptation actions performed on them). The Website engine loads the models (from the Application Database) of the particular user when s/he starts a new session. These models are modified along the different sessions depending on the adaptation actions performed. This implies each user will see a different adaptive view of the Website every session s/he browses it. The Website engine captures the events the user performs with his browsing actions and sends them to the PRML Evaluator module. This module is responsible of evaluating and performing the personalization rules attached to the events. When a rule is triggered, to evaluate the rule conditions and perform the proper actions we have implemented a .NET component using the ANTLR Parser generator [1]. From the PRML grammar we can generate the syntactic trees which help us to evaluate the rule conditions and perform
them if necessary. Finally to execute the actions of the rules, we have implemented in C# the different actions types that we can find in PRML.

- The third layer contains the Application database and a text file containing the set of rules defining personalization policies on the Website. This set of rules is defined using the PRML rule language. Next an overview of the PRML language is given and the implementation of the actions supported by AWAC is explained.

4.2 AWAC: PRML Support

PRML was born in the context of the OO-H [11] Web design method to extend it with personalization support (A-OOH). PRML is an ECA rule based language. These rules update the information needed for the adaptation in the User Model and perform adaptation actions over the structure of the Website.

The AWAC Tool does not implement all the adaptation events and actions of PRML. AWAC only supports the detection of simple browsing events (i.e. not a sequence of events). The events supported are: SessionStart (triggered with the start of the browsing session by a user) SessionEnd (triggered when the browsing session expires after certain inactivity of the user in the system or when the user explicitly finishes the session), Navigation (i.e. click on a link) and LoadElement (i.e. loading of a navigational node independently of the link that loads the node).

In Table 1 the personalization actions supported by PRML and the ones implemented in AWAC are shown. Next the actions supported by AWAC are detailed.

Table 1: PRML Support in the AWAC tool

<table>
<thead>
<tr>
<th>Action</th>
<th>PRML</th>
<th>Implemented in AWAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updating User Model Content</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Filtering content (concept instances)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Filtering content (concept attributes)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Link hiding</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sorting content (node instances)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Adding / deleting links</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Adding filters to links</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Dynamically grouping users</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

The AWAC tool implements the following actions over the different elements of the A-OOH models:

- Actions over attributes (User and Navigation Models):

---

4 Personalization of the presentation is not yet considered.

5 Note that PRML rules can be attached to nodes or to links of the Navigation Model.
- **Updating an attribute value from the User Model (setContent):** This action allows modifying/setting the value of an attribute (of a concept) of the User Model. To perform a `setContent` action the PRML Evaluator updates the corresponding attribute value in corresponding model of the Application database.

- **Filtering attributes in the Navigation Model nodes (selectAttribute):** By means of this action a node can be restricted by hiding or showing some of the attributes of the Domain Model/User Model related concept. The PRML Evaluator updates the visibility of the corresponding attribute in the proper model of the Application database.

- **Actions over links (Navigation Model):**
  - **Hiding links and their target nodes (hideLink):** Analogous to filtering data content, PRML also supports filtering links. This action affects to the visibility of a link so in the same way as attributes, each link and node in A-OOH contains a visibility property. The PRML Evaluator updates the visibility of the corresponding link (and its target node) in the proper model of the Application database.

- **Actions over nodes (Navigation Model):**
  - **Filtering node instances (selectInstance):** This action shows only the selected instances of a Domain Model/User Model concept for a user depending on the personalization requirements we want to support. The PRML Evaluator module selects the instances to be shown in the current session from the Application database.
  - **Sorting node instances (sort):** In PRML node instances can be sorted by a certain value to satisfy a personalization requirement. The PRML Evaluator module selects and sorts the instances to be shown in the current session from the Application database.

The adaptive actions are only performed once during a session not to overwhelm the user, this means the filtered attributes, links, instances and sorted instances will be the same during the present session. However, the desirable option (future work) is that the designer decides when adaptation should take place to fulfill each personalization requirement. Next, by means of a running example, the steps needed for creating and running a Website using AWAC are described.

## 5 Step 1: Creating the A-OOH Models

To better understand how to generate adaptive Websites with AWAC using A-OOH and PRML, a case study is presented which describes an online library. In this library information about the books is shown, as well as reviews done by readers visiting our Website. Users can consult the books and buy them, adding them first to the shopping basket. In Figure 2 the Domain and User Model are shown. In the User Model we store different information needed to fulfill the personalization requirements initially specified for the Website:
1. Users will see fifteen (maximum) recommendations of books in which authors they are most interested in (sorted by interest).

2. If the user does not have enough interest in any book author to get personalized recommendations, the recommendations link is not shown: In order to fulfil the 1st and 2nd requirements we need to acquire and update the user interest in the different authors.

3. Users that have bought at least 10 books will be offered a discount in the book price: In this case, to fulfil this requirement (and offer the price discount to the user) the number of books bought (in all the sessions) should be stored in the UM.

In the updatable data space defined by UM the information describing the user’s interests on the authors is stored as DomainDependent information, according to the defined personalization requirements. In the UM we also have the Buy class. This class represents a navigation event triggered by the user behaviour and stores the number of clicks done in the link Buy in the attribute clicks (we will use this information to fulfil the 3rd personalization requirement of the running example). This value is stored as long-term data (i.e. independent of the session), because the designer wants to personalize on basis of the number of books bought in total by the user. Note that to store the number of clicks on any other link we would just have to add a new Navigation element to the UM.

Figure 3 shows the Navigational Model of the online library. When the user enters the Website (login) he finds a collection of links (i.e. menu) with the links ConsultBooks, Recommendations, ViewCategories and SecondHandBooks as a
starting point. If the user navigates through the first link (ConsultBooks) he will find an indexed list of all the books (indexed by the book’s name). The user can click in any of the book names to view the details of the chosen book (see in Figure 3 the navigational class BookDetails). Moreover the user can see reviews done by other users of the different books. When the user clicks on ViewCategories, an indexed list of the categories is shown (indexed by the category’s name). When the user clicks in one of the categories he will see the books associated to that category. If the user navigates through the SecondHandBooks link he can see all the books used that are on sale.

Once the NM is specified the Presentation Model has to be defined. It is captured by one or more Design Presentation Diagrams (i.e. DPDs). There should be one DPD for each NM defined in the system. This diagram enriches the Navigation Diagram described in previously. The DPD describes how the navigation elements are presented to the user. The DPD main objectives are to provide the page structure of the Website, grouping the NM Navigational Nodes into Presentation Pages. These Presentation Pages are abstract pages, which in the final implementation can be represented by one or more concrete pages. The designer can add static pages also directly on the DPD. This is represented in the level 0 of the DPD.

The second goal is to describe the layout and style of each page of the interface. The designer should decide which interface components are going to be in the page and where are they going to be positioned. Moreover, he can modify the individual structure of the pages and add static elements directly on the DPD. This is
represented in the level 1 of the DPD, exploding each of the abstract pages previously defined in the level 0.

In Figure 4 the DPD (level 1) for the page that shows the recommendations to the user. This page is defined as a set of layouts and cells. The layouts are also composed of cells. Cells contain interface components that represent the elements of the Web page. The Page Chunk element represents a fragment of an abstract page which has associated a Presentation model where the components shown in this fragment are defined. This fragment can be reused in the different pages that compose the Web application. In this way we avoid to make several diagrams of common parts to all (or many of) the pages. Inside this package the Presentation Model attached to the Page Chunk is shown. In this case, the Menu is defined as a page chunk. The final Web page for recommendations generated on basis of these models is in Figure 8.

Fig. 4. (a) PM for the Recommendations Page (b) PM for the Menu page chunk

Note that it is not the objective of the paper to explain the A-OOH design models in depth, but to give an overview so the reader can better understand the proposal.

6 Step 2: Adding Personalization using a PRML file

What is left now is defining the Personalization Model in which the adaptive actions to perform over the previously defined set of models are specified. For this purpose we use the PRML language. The basic structure of a rule defined with PRML is the following:

```
When event do
    [Foreach expression]
    [If condition then] action
    [endIf]
[endForeach]
endWhen
```

A PRML rule is formed by an event and the body, which contains a condition (optional) and an action to be performed. The event part of the rule states when the...
A rule should be triggered. Once triggered the rule condition is evaluated (if any) and in the case the condition is evaluated positively the action is performed. A Foreach expression can be also present in the rule when the action and the condition act over a set of instances.

Please note that the purpose of this paper is not to explain the PRML language, for a better understanding of the rules the reader can consult [10].

Now we define the PRML configuration file for our running example. For a better comprehension we can divide this file in two sections:

The acquisition rule section defines the rules needed to gather the required information to personalize. In our example we have two acquisition rules:

- The first rule is triggered by the activation of the link ViewDetails. The rule updates the proper instance of the interest class on the author the user consults the details of. This is done using the SetContent action.

- The second rule is triggered by the activation of the link Buy. The number of books bought along all the sessions is stored in the buy class of the User Model by increasing the attribute clicks every time the user buys a book.

The personalization rule section contains the personalization rules which describe the effect of personalization in the Website. In our example, we have three personalization rules:

- The first rule is triggered by the activation of the Recommendations link. To fulfill the first personalization requirement we need to define a Sort action. It operates over a set of instances (i.e. the set of books to sort). The syntax of this action is very similar to SQL. This rule properly sorts the book instances by the user interest degree on the different authors returning the fifteen (maximum) with highest interest (and greater than 100) to be recommended to the user.

- The second rule hides the Recommendations link if there is no recommendation to show to the user. It is triggered by the SessionStart event (i.e. when the user enters the Website).

- The third rule is triggered by the activation of the ViewDetails link. This rule checks the number of books bought by the user and if this number is greater than 10 it shows the attribute discount of the book to be shown.

---

6 Some attributes of the rules have been omitted for simplicity reasons.
### PERSONALIZATION SECTION

#### RULE: "ShowRecommendations"

When Navigation.Recommendations(NM.Book* books) do

Sort books orderBy UM.User.userToInterest.degree ASC LIMIT 15 Where UM.User.userToInterest.interestToAuthor.ID = books.bookToAuthor.ID and UM.User.userToInterest.degree > 100

endWhen

#### RULE: "HideLink"

When SessionStart do

If ForAll (UM.User.userToInterest) (UM.User.userToInterest.degree == null or UM.User.userToInterest.degree < 100) then

hideLink(NM.Recommendations)

endIf

endWhen

#### RULE: "ShowDiscount"

When Navigation.ViewDetails(NM.Book book) do

If UM.User.userToBuy.clicks > 10 then

book.Attributes.selectAttribute(discount)

endIf

endWhen

### 7 Step 3: Generating the Web Application with AWAC

Once modelled, the Web Application has to be generated using the AWAC tool. The AWAC interface is a Web page in ASP.Net. To generate a Web application using AWAC the following steps are to be taken:

1. **Save the UML A-OOH models in XMI format.**
2. **Create a new project in the AWAC environment and load the XMI files containing the A-OOH models.**
   
   This is done in the main view of the AWAC tool, shown in Figure 5(a). The loaded models can be viewed selecting the corresponding option in the Adaptive OO-H models section, as it can be seen in Figure 5(b).
3. **Save the PRML file as a text file and upload the file.**
   
   In the menu, the option PRMLTools → Edit Rules shows a new view of the AWAC tool in which we can load the file containing the PRML rules for our Web application (see Figure 6). It is desirable that the extension of the file is “.p” for clarity purposes, but this is not mandatory.
   
   In further versions it will be possible to edit the rules loaded and check if they are syntactically and semantically correct.
4. **Generate the Web application**
   
   Once the A-OOH models and the PRML file are uploaded, the Web application can be created and downloaded as a compressed rar file (see Figure 7).

   As explained in Section 4, the output of the AWAC tool contains the generated adaptive Website (Web pages in ASP.net), the modules for managing the personalization (the Website Engine and the PRML Evaluator) and the application database.
5. Deploying and running the Website

Once generated, the adaptive Web application can be run in a Web server. The adaptive Web pages shown to the users will differ depending on their browsing actions. In Figure 8 the recommendations page is shown for two different users. Depending on the user interest (stored in the UM) the books to recommend vary. To properly show the recommendations to the user the AWAC modules generated for managing the personalization (explained in Section 4.1) follow the next steps (see Figure 1b): The Website Engine gathers the request of the user for the recommendations page and triggers the user browsing event (i.e. click on recommendations) sending it to the PRML Evaluator module. This module checks if there is any rule triggered, and finds the “ShowRecommendations” rule which executes. This rule selects and sorts the corresponding book instances to be shown from the Navigational Model. These recommendations won’t change until the next time the user starts a session. This decision has been taken not to overwhelm the user with constant updates.
8 Conclusions and Future Work

This paper presents AWAC, a prototype CAWE tool for the automatic generation of personalized Web applications. This tool implements the A-OOH (Adaptive Object Oriented Hypermedia) methodology, which is an extension of the OO-H method for supporting the modelling of personalized Websites. The input of the AWAC tool is the set of A-OOH design models needed to model the adaptive Website to generate. The output is the generated Website with its database. The output includes a Web engine and a personalization module which allow to manage the personalization at runtime of the final Web pages. The personalization rules can be edited in an independent way of the rest of the application, which improves the personalization maintenance.
Some experiments are being done with AWAC. One is the generation and running of the (adaptive) Intranet of the university lab of the authors. This Intranet is now online and the users accesses are being studied. The purpose of this experiment is twofold: to study the satisfaction of the users (in terms of personalization performed, fast response…) and the improvement of the personalization techniques applied. As future work, besides implementing all the PRML functionality, we would like to add a graphical user interface in order to define the A-OOH models using AWAC. Now, to define the A-OOH models and generate the XMI files we use the Enterprise Architect Design tool [8] in which we have defined the UML profiles needed for the modelling of the A-OOH diagrams.

References
1. ANTLR, ANother Tool for Language Recognition, http://www.antlr.org/
6. Db4o Database For Objects www.db4o.com
Active Rules for Runtime Adaptivity Management

Florian Daniel, Maristella Matera, Alessandro Morandi, Matteo Mortari, and Giuseppe Pozzi

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy
{daniel,matera,morandi,mortari,pozzi}@elet.polimi.it

Abstract. The trend over the last years clearly shows that modern Web development is evolving from traditional, HTML-based Web sites to full-fledged, complex Web applications, also equipped with active and/or adaptive application features. While this evolution unavoidably implies higher development costs and times, such implications are contrasted by the dynamics of the modern Web, which demands for even faster application development and evolution cycles. In this paper we address the above problem by focusing on the case of adaptive Web applications. We illustrate an ECA rule-based approach, intended to facilitate the management and evolution of adaptive application features. For this purpose, we stress the importance of decoupling the active logic (i.e. the adaptivity rules) from the execution of the actual application by means of a decoupled rule engine that is able to capture events and to autonomously enact adaptivity actions.

1 Introduction

Adaptability (the design-time adaptation of an application to user preferences and/or device characteristics [1]) and adaptivity (the runtime adaptation of an application to a user profile or a context model [1]) have been studied in the last years by several authors in the field of Web engineering. Adaptability is intended as the capability of the design to fit an application to particular needs prior to the execution of the application. Adaptivity is intended as autonomous capability of the application to react and change in response to specific events occurring during the execution of the application, so as to better suit dynamically changing execution conditions. Recently, adaptivity has been extended to the case of context-aware Web applications [2], where adaptivity is based on a dynamically updated context model, upon which the adaptive application is built.

As is the nature of the Web engineering discipline, the previous approaches to adaptability, context-awareness and adaptivity primarily focus on the definition of design processes to achieve adaptation, thereby providing efficient methods and tools for the design of such a class of applications. For instance, model-driven methods [1, 2], object-oriented approaches [3], aspect-oriented approaches [4],
and rule-based paradigms [5,6] have been proposed for the specification of adaptation features in the development of adaptive Web applications. The resulting specifications facilitate the implementation of the adaptation requirements and may also enhance code coherence and readability. Unfortunately, in most cases during the implementation phase all the formalizations of adaptivity requirements are lost, and the adaptivity features become buried in the application code. This aspect implies that changes and evolutions of adaptive behaviors after deployment of the application are difficult, unless a new version of the application is implemented and released.

Based on our experience in the model-driven design of adaptive/context-aware Web applications [2,7], we are convinced that the next step in this research area is to support the dynamic management of adaptivity features: on one hand this will require proper design time support (e.g. languages or models), on the other hand this will require suitable runtime environments where adaptivity specifications can be easily administered.

In [8] we already outlined a first conceptual framework for this approach. We now focus on the evolution of that work, describing a rule-based language (ECA-Web) for the specification of adaptive behaviors, orthogonally to the application design, and its concrete implementation. The resulting framework provides application designers with the ECA-Web language and application administrators with the possibility to easily manage ECA-Web rules (inserting, dropping, and modifying rules), even after the implementation and the deployment of the application, i.e. at runtime. As envisioned above, by the described approach we enable the decoupled management of adaptivity features at both design- and run-time.

This paper is organized as follows. Section 2 discusses some related works on adaptivity in the Web. Section 3 introduces the ECA-Web rule language for the specification of adaptive behaviors for Web applications and, then, shows how ECA-Web rules can be executed by a proper rule engine and integrated with the execution environment of the adaptive Web application. Section 4 discusses the prototype of an adaptive Web application supported by ECA-Web rules and shows the usage of the active rule language. Section 5 describes the implementation of the overall system and reports on first experiences with the rule-based adaptivity specification and the runtime management of adaptivity rules. Finally, Section 6 concludes the paper and discusses future work.

2 Related Work

Conceptual modeling methods provide systematic approaches to design and deploy Web applications. Several well-established design methods have been so far extended to deal with Web application adaptivity. In [1] the authors extend the Hera methodology with two kinds of adaptation: adaptability with respect to the user device and adaptivity based on user profile data. Adaptation rules (and the Hera schemas) are expressed in RDF(S) (Resource Description Framework/RDF Schema), attached to slices and executed by the AHA engine [9].
The UWA Consortium proposes WUML [10] for conceptual hypertext design. Adaptation requirements are expressed by means of OCL-based customization rules, referring to UML class or package elements. In [11] the authors present an extension of WSDM [12] to cover the specification of adaptive behaviors. In particular, an event-based Adaptive Specification Language (ASL) is defined, which allows designers to express adaptations on the structure and the navigation of the Web site. Such adaptations consist in transformations of the navigation model, which can be applied to nodes (deleting/adding nodes), information chunks (connecting/disconnecting chunks to form a node), and links (adding/deleting links). In [4] the authors explore Aspect-Oriented Programming techniques to model adaptivity in the context of the UML-based Web engineering method UWE. Recently, WebML [13] has been extended to cover adaptivity and context-awareness [2]. New visual primitives cover the specification of adaptivity rules to evaluate conditions and to trigger some actions for adapting page contents, navigation, hypertext structure, and presentation. Also, the data model has been enriched to represent some meta data supporting adaptivity.

The previous works benefit from the adoption of conceptual models, which provide designers with powerful means to reason at a high-level of abstraction, independently of implementation details. However, the resulting specifications of adaptivity rules have the limit of being embedded inside the design models, thus raising problems in the maintenance and evolution of the adaptivity requirements, once the application is released.

Recently, active rules, based on the ECA (Event-Condition-Action) paradigm, have been proposed as a way to solve the previous problem. Initially exploited especially in fields such as content evolution and reactive Web [14–16], ECA rules have been recently adopted to support adaptivity issues in Web applications. In particular, the specification of decoupled adaptivity rules provides a way to design adaptive behaviors along an orthogonal dimension. Among the most recent and notable proposals, the work described in [5] enriches the OO-H model with personalization rules for profile groups: rules are defined in PRML (Personalization Rule Modeling Language) and are attached to links in the OO-H Navigation Access Diagram. The use of a PRML rule engine is envisioned in [6], but its real potential for adaptivity management also at runtime remains unexplored.

In line with the previous work, the approach we describe here proposes a rule-based language adopting the ECA paradigm. We call the language ECA-Web, emphasizing that it is able to express events and actions that may occur in a Web environment. Although the proposed language allows one to reference elements of a conceptual specification of an application\(^1\), it is a self-sufficient language for the specification of adaptivity rules. The novelty of our work is however the development of a decoupled environment for the execution and administration of adaptivity rules, which allows the management of adaptivity features to be kept totally independent of the application execution. This aspect introduces several advantages in terms of maintainability and evolvability.

\(^1\) In this paper we shall briefly show how the language can be bound to WebML [13].
3 Enabling Dynamic Adaptivity Management

In the following we introduce the design component (the ECA-Web language) and the runtime component (the rule execution environment) that enable the dynamic administration of adaptivity features.

3.1 ECA-Web

ECA-Web is an XML-based language for the specification of active rules, conceived to manage adaptivity in Web applications. The syntax of the language is inspired by Chimera-Exception, an active rule language for the specification of expected exceptions in workflow management systems [17]. ECA-Web is an evolution of the Chimera-Web language we already proposed in [8], and it is equipped with a proper rule engine for rule evaluation and execution.

The general structure of an ECA-Web rule is summarized in Figure 1. A typical ECA-Web rule is composed of five parts: scope, events, conditions, action and priority. The **scope** defines the binding of the rule with individual hypertext elements (e.g. pages, links, contents inside pages). By means of **events** we specify how the rule is triggered in response to user navigations or changes in the underlying context model. In the **condition** part it is possible to evaluate the state of application data (e.g. database contents or session variables) to decide whether the action is to be executed or not. The **action** specifies the adaptation of the application in response to a triggered event and a true condition. The **priority** defines an execution order for rules concurrently activated over the same scope; if not specified, a default priority value is assigned. More details on the rule specification by means of ECA-Web are given in the next section, where we discuss the architecture of the runtime environment for rule execution. An example of ECA-Web rule will then be shown in Section 4.
3.2 The Integrated Runtime Architecture

The execution of ECA-Web rules demands for a proper runtime support. Figure 2 summarizes the functional architecture of the system, highlighting the two main actors: the Rule Engine and the Web Server hosting the Web application. The Rule Engine is equipped with a set of Event Managers to capture events, and a set of Action Enactors to enable the execution of actions. The communications among the single modules are achieved through asynchronous message exchanges (Message-Oriented Middleware).

**Event Managers.** Each type of ECA-Web event is supported by a suitable event manager (i.e., Web Event Manager, Data Event Manager, Temporal Event Manager, and External Event Manager). As in [8], event managers and ECA-Web provide support for the following event types:

- *Data events* refer to operations on the application’s data source, such as create, modify, and delete. In adaptive Web applications, such events can be monitored on user, customization, and context data to trigger adaptivity actions with respect to users and their context of use. Data events are
managed by the Data Event Manager, which runs on top of the application’s data source.

- Web events refer to general browsing activities (e.g. the access to a page, the submission of a form, the refresh of a page, the download of a resource), or to events generated by the Web application itself (e.g. the start or end of an operation, a login or logout of the user). Web events are risen in collaboration with the Web application and captured by the Web Event Manager. Since adaptivity actions are typically performed for each user individually, Web events are also provided with a suitable user identifier (if any).

- External events can be configured by a dedicated plug-in mechanism in form of a Web service that can be called by whatever application or resource from the Web. An external event could be for example a notification of news fed into the application via RSS. When an external event occurs, the name of the triggering event and suitable parameters are forwarded to the rule engine. External events are captured by means of the External Event Manager.

- Temporal events are subdivided into instant, periodic, and interval events. Interval events are particularly powerful, since they allow the binding of a time interval to another event (anchor event). For example, the expression “five minutes after the access to page X” represents a temporal event that is raised after the expiration of 5 minutes from the anchor event “access to page X”. Temporal events are managed by the Temporal Event Manager, based on interrupts and the system clock.

The managers for external and temporal events are general in nature and easily reusable. The Data Event Manager is database-dependent. The Web Event Manager requires a tight integration with the Web application.

**Action Enactors.** Actions correspond to modifications to the Web application or to executions of back-end operations. Typical adaptation actions are: adaptation of page contents, automatic navigation actions, adaptation/restructuring of the hypertext structure, adaptation of presentation properties, automatic invocation of operations or services. Adaptations are performed according to the user’s profile or his/her context data.

While some actions can easily be implemented without any explicit support from the Web application (e.g. the adaptation of page contents may just require the setting of suitable page parameters when accessing the page), others may require a tighter integration into the application’s runtime environment (e.g. the restructuring of the hypertext organization). The level of application support required for the implementation of the adaptivity actions thus heavily depends on the actual adaptivity requirements. However, application-specific actions can easily be integrated into the ECA-Web rule logic and do not require the extension of the syntax of the rule language (an example of the use of actions is shown in Figure 7).

---

Footnote: In our current implementation we support PostgreSQL. Modules for other database management systems are planned for future releases.
Rule Engine. In the architecture depicted in Figure 2, the Rule Engine is in charge of identifying the ECA-Web rules that correspond to captured events, of evaluating conditions, and of invoking action enactors – in case of conditions evaluating to true.

In the rule engine, a scalable, multithreaded Rule Evaluator evaluates conditions to determine whether the rule’s action is to be performed or not, depending on the current state of the application. In ECA-Web, conditions consist of predicates over context data, application data, global session variables, and/or page parameters. For example, in the condition part of an ECA-Web rule it is possible to specify parametric queries over the application’s data source, where parameters can be filled with values coming from session variables or page parameters.

The rule engine also includes a Rule Registry for the management of running, deployed ECA-Web rules. Deployed rules are loaded into the Rule Registry, a look-up table for the efficient retrieval of running rules, starting from captured events. The internal execution logic of a triggered rule is graphically summarized in Figure 3.

### 3.3 ECA-Web Rule Management

While the Rule Registry contains only deployed rules for execution, the Rule Repository offers support for the persistent storage of rules. For the management
of both Rule Registry and Rule Repository, we provide a Rule Administration Panel that allows designers to easily view, add, remove, activate, and deactivate rules. Figure 4 shows a screenshot of the Rule Administration Panel.

3.4 Deploying ECA-Web Rules

Activating or deploying an ECA-Web rule is not a trivial task and, depending on the rule specification, may require to set up a different number of modules. During the deployment of an ECA-Web rule, the XML representation of the rule is decomposed into its constituent parts, i.e. scope, events, conditions, action, and priority, which are then individually analyzed to configure the system. The scope is used to configure the Web Event Manager and the Web Action Enactor. The events are interpreted to configure the respective event managers and to set suitable triggers in the application’s data source. The conditions are transformed into executable, parametric queries in the Rule Registry. The action specification and the rule’s priority are as well fed into the Rule Registry. Each active rule in the system is thus represented by an instance in the Rule Registry, (possibly) by a set of data triggers in the database, and by a set of configurations of the event managers and the action enactors.

The registry allows the concurrent access by multiple Rule Evaluators. Priorities are taken into account in the action enactor modules, which select the action to be performed for the page under computation (the scope) from the queue of possible actions, based on rule priorities.
During the deployment of an ECA-Web rule, conflict resolution and termination analyses will be performed in line with the methods conceived and implemented for the Chimera-Exception language [17].

### 3.5 Enacting Adaptivity

*External* and *data* actions can be executed immediately upon reception of the respective instruction from the rule engine. The enaction of *Web* actions, which are characterized by adaptations visible on the user’s browser, is possible only when a “request for adaptation” (a page request) comes from the browser. In fact, only in presence of an explicit page request, the Web application is actually in execution and, thus, capable to apply adaptations. This is due to the lack of suitable push mechanisms in the standard HTTP protocol.

In order to provide the application with active/reactive behaviors, in our previous works we therefore studied two possible solutions: (i) periodically refreshing the adaptive page currently viewed by the user [2], and (ii) periodically monitoring the execution context in the background (e.g. by means of suitable Rich Internet Application – RIA – technologies) and refreshing the adaptive page only in the case adaptivity actions are to be performed [7, 8]. Both mechanisms are compatible with the new rule-based architecture and enable the application to apply possible adaptivity actions that have been forwarded to the *Web Action Enactor* by the *Rule Engine*.

### 4 Case Study

In the context of the Italian research project MAIS³ we have developed a context-aware Web application, called *PoliTour*, supplying information about buildings and roads within our university campus at Politecnico di Milano. The application is accessed through a PDA equipped with a GPS receiver for location sensing. User positioning is based on geographical longitude and latitude. As the user moves around the campus, the application publishes location-aware data, providing details about roads and buildings. The required adaptivity consists of (i) adapting page contents according to the user’s position, and (ii) alerting the user of possible low connectivity conditions, based on the RSSI (Received Signal Strength Indicator) value of the wireless Internet connection. The alert consists in changing the background color of the displayed page.

The application has been designed with the WebML model, a visual notation for specifying the content, composition, and navigation features of hypertext applications [13]. In this paper we use the WebML notation for two distinct purposes: (i) to easily and intuitively describe the reference application, and (ii) to better highlight how the active rules introduced in the next section may take advantage from a formally defined, conceptual application model for the definition of expressive adaptivity rules. The approach we propose in this paper,

³ http://www.mais-project.it
It is worth noting that the approach based on ECA-Web described in this paper is not to be considered an alternative solution to the conceptual design approaches so far proposed in the literature for Web application modeling. Rather, we believe that the best expressiveness and a good level of abstraction for the illustrated adaptivity specification language will be achieved by complementing the current modeling and design methods (such as WebML, Hera, OO-H or OOHDM). In fact, in this paper we hint at the specification of ECA-Web rules on top of WebML (both data and hypertext models), just like SQL triggers are defined on top of relational data models. This consideration is in line with the proposal by Garrigós et. al [6], who show how to apply their PRML rule language to several different conceptual Web application models.

The conceptual model of the application serves as terminological and structural reference model for the specification of adaptivity rules and, thus, allows application developers to keep the same level of abstraction and concepts already used for the design of the main application. In terms of WebML, for example, this could mean to restrict the scope of individual rules to specific hypertext elements like content units, pages, or areas, or to relate events to specific links or units. The same holds for actions, which could for example be applied to single units or even attributes.

4.1 Application Design with WebML

Figure 5 depicts a simplified version of the data schema underlying the PoliTour application, expressed in the Entity-Relationship (ER) notation. Five entities compose the context model, which is required in addition to the user identity to achieve the context-aware features of the application. The entities Connectivity and Position are directly connected to the entity User, as they represent context data which are individual for each user of the system. Position contains the latest GPS coordinates for each user, Connectivity contains a set of discrete connectivity levels that can be associated to users, based on their current RSSI. GPS coordinates and RSSI are sensed at the client side and periodically
communicated to the application server in the background [7]. The entities Area, Building, and Road provide a logical abstraction of raw position data: buildings and roads are mapped onto a set of geographical areas inside the university campus, which enables the association of a user with the building or road he/she is located in, based on the GPS position. The entity Classroom is located outside the context model, as the application is not able to react to that kind of granularity, and the respective data is considered additional application content.

Figure 6 depicts the WebML-based hypertext schema of the PoliTour application defined on top of the ER schema shown in Figure 5. The application hypertext is composed of three pages: Buildings, Roads, and Classroom. Page Buildings shows a list of buildings (BuildingsIndex unit) the user can select from. By choosing one of the buildings, the respective details (BuildingData unit) and the list of classrooms (ClassroomsIndex unit) of the building is shown. If interested in, the user can select one of the building’s classrooms and navigate to the Classroom page. Similarly, page Roads shows a list of roads for selection by the user. The details of selected roads are shown by the RoadData unit positioned in the middle of the page. The identifier of the selected road is further propagated to the NearbyBuildings unit, which shows the buildings adjacent to the road and allows the user to navigate to the Buildings page. The two pages Buildings and Roads are further tagged as landmark pages, meaning that they can be accessed through a global navigation menu. Page Buildings is also tagged as the home page of the application.
4.2 Defining an ECA-Web Rule

The full specification of the application’s adaptivity requires several different ECA-Web rules to manage the adaptation of the contents in the pages Buildings and Roads, and to alert the user of low connectivity conditions. Figure 7 shows the ECA-Web rules that adapts the content of the page Buildings to the position of the user inside the university campus.

The scope of the rule binds the rule to the Buildings page. The triggering part of the rule consists of two data events, one monitoring modifications to the user’s longitude parameter, one monitoring the user’s latitude parameter. In the condition part of the rule we check whether there is a suitable building associated to the user’s current position (notnull condition), in which case we enact the Showpage adaptivity action with new page parameters, suitably computed at runtime; otherwise, no action is performed. The condition evalu-
tion requires the extraction from the data source of two data items (\texttt{<object>}), namely the position of the current user and the area in which the user is located. The selection condition is enclosed within the \texttt{<requirements>} tag. In the action part of the rule we link the \texttt{bellerofonte.actions.Showpage} Java class, which contains the necessary logic for the content adaptation action. The variable \texttt{building_oid} has been computed in the condition part of the rule and is here used to construct the URL query to be attached to the automatic page request that will cause the recomputation of the page and, thus, the adaptation of the shown content.

It is worth noting that the scope of the previous rule is limited to one specific hypertext page. There might be situations requiring a larger scope. For example, the rule for alerting users about low connectivity is characterized by a scope that spans all the application’s pages; in terms of WebML, binding an ECA-Web rule to all pages means to set the scope of the rule to the site view, i.e., a model element (see site view \texttt{PoliTour} in Figure 6). The scope of the rule is specified as follows:

\begin{verbatim}
<scope>
  <siteview>PoliTour</siteview>
</scope>
\end{verbatim}

As for the dynamic management of adaptivity rules, we could for example be interested in testing the two adaptivity features (location-aware contents and the low connectivity alert) independently. We would thus first only deploy the rule(s) necessary to update the contents of the Buildings and Roads pages and test their functionality without also enabling the alert. Then we could disable this set of rules and enable the rule for the alert and test it. If both tests are successful, we finally could enable both adaptivity features in parallel and test their concurrent execution.

5 Implementation

The proposed solution has been developed with scalability and efficiency in mind. The Web application and the rule engine are completely decoupled, and all communications are based on asynchronous message exchanges based on JMS (Java Message Service). The different modules of the proposed system can easily be distributed over several server machines. The overhead introduced into the Web application is reduced to a minimum and only consists of (i) forwarding Web events and (ii) executing adaptivity actions. These two activities in fact require access to the application logic. In fact, depending on the required adaptivity support, event mangers and action enactors may require different levels of customization by the Web application developer. The customization consists in the implementation of the application-specific events and of the actions that are to be supported by the adaptive application.

\footnote{\textit{Bellerofonte} is the current code name of the rule engine project.}
To perform our first experiments with ECA-Web and the rule engine, we have adapted the PoliTour application, which we already extensively tested when developing our model-driven approach to the design of context-aware Web applications [7]. As for now, our experiments with a limited number of rules have yielded promising results. Experimentations with larger numbers of active rules, different adaptive Web applications, and several users in parallel are planned.

Also, to really be able to take full advantage of the flexibility provided by the decoupled adaptivity rule management, a set of suitable adaptivity actions needs to be implemented. Our current implementation provides support for data actions and a limited set of Web actions (namely, ShowPage for adapting page contents, and ChangeStyle for adapting presentation style properties). Data actions are currently applied only to entities and attributes that are directly related to the user for which the action is being executed. Also, condition evaluation is automatically confined to those context entities and attributes that are related to the user for which the rule is being evaluated. We are already working on extending condition evaluation to any application data, coming from the data source as well as from page and session parameters.

In the context of WebML, the provision of a set of predefined adaptivity actions will lead to a library of adaptivity actions, possibly integrated into the WebML runtime environment. In the case of general Web applications, the rule engine can be used in the same fashion and with the same flexibility, provided that implementations of the required adaptivity actions are supplied.

6 Conclusions

We believe that the decoupled runtime management of adaptivity features represents the next step in the area of adaptive Web applications. In this paper we have therefore shown how to empower design methods for adaptivity with the flexibility provided by a decoupled environment for the execution and the administration of adaptivity rules. The development of Web applications in general is more and more based on fast and incremental deployments with multiple development cycles. The same consideration also holds for adaptive Web applications and their adaptivity requirements. Our approach allows us to abstract the adaptive behaviors, to extract them from the main application logic, and to provide a decoupled management support, finally enhancing the maintainability and evolvability of the overall application.

In our future work we shall focus on the extension of the ECA-Web language to fully take advantage of the concepts and notations that can be extracted from conceptual Web application models (e.g. from WebML models). We shall also investigate termination, complexity, and confluence issues, trying to apply Chimera-Exception’s Termination Analysis Machine [17] to ECA-Web. Extensive experimentations are planned to further prove the advantages deriving from the decoupled approach.
References

Using Object Variants to Support Context-Aware Interactions

Michael Grossniklaus and Moira C. Norrie
Institute for Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland
{grossniklaus,norrie}@inf.ethz.ch

Abstract We discuss the need to extend general models and systems for context-awareness to include adaptation of interactions to context. Our approach was motivated by our experiences of developing mobile applications based on novel modes of interaction. We describe how we were able to support context-aware interactions using an object-oriented framework that we had already developed to support context-aware web applications.

1 Introduction

Context-awareness in web engineering involves the adaptation of applications to user situations. At the level of models and frameworks to support web engineering, several generic approaches have been proposed to allow application developers to determine what notions of context and adaptation are relevant to specific applications. General models and mechanisms have therefore been developed that can cater for various forms of adaptation that correspond to personalisation, internationalisation, multi-channel access, location-awareness etc. Furthermore, for full generality, it should be possible to adapt any aspect of a web application, including content, structure and presentation.

However, one aspect that has received relatively little attention is the need to adapt interaction processes to context and how existing models and mechanisms can be generalised to support this. Our experiences have shown that supporting mobile and ubiquitous applications often involves working with new modes of interaction resulting from the characteristics of the different devices used. The nature of these devices is such that the linearity of traditional web-based transactions may be lost and input data may be assembled from various sources and in different orders rather than being specified in a single step. This also means that users need to be carefully guided through the interaction so that they are aware of the current interaction state. An important factor here is that users can be supplied with context-dependent help information according to the interaction state.

In this paper, we describe how we were able to exploit an object-oriented framework that was developed to support context-aware web engineering to support context-aware interactions. We begin in Sect. 2 with a discussion of related
work and a motivation of our approach. Sect. 3 presents the main features of
the object-oriented framework and how it supports context-awareness through a
notion of multi-variant objects. In Sect. 4, we show how the mechanisms used to
supported context-awareness in our framework could be used to support context-
aware interactions. Section 5 provides a general discussion of the approach and
directions of future work. Concluding remarks are given in Sect. 6.

2 Background

The need for context-awareness is well documented in the field of web engineer-
ing. Its impact can be witnessed in several model-based approaches and a few
implementation platforms recently proposed. For example, the Web Modelling
Language (WebML) [1] has been extended with primitives that allow adaptive
and context-aware sites to be modelled [2]. To manage context information,
the data model is extended with a context model that is specific to the de-
veloped application. To gather context information, two additional units—Get URL
Parameter and Get Data—have been introduced. The first unit retrieves con-
text information sent to the server by the client device encoded in the URL.
The second unit extracts context information according to the context model
from the database on the server. Each page that is considered to be context-
dependent is associated in the model with a context cloud that contains the
adaptation operation chains. These operation chains can be built from the stan-
dard WebML operation units as well as from units that have been introduced to
model conditional or switch statements in the specification of workflows. When a
context-aware page is requested, the corresponding operation chain is executed
and the content of the page adapted accordingly. However, in order to adapt
the content itself, the context-dependent entities in the data model have to be
associated with entities representing the relevant context dimensions. Depend-
ing on the complexity of the application, this can lead to a very cumbersome
data model that is no longer true to the orthogonal notion of context. Apart
from such content adaptation, it is also possible to adapt the navigation and
the presentation. The newly introduced Change Site View unit can be used to
forward a client from one site view to another, whereas the Change Style unit
adapts the web site in terms of colours and font properties. Another extensions
to WebML allows reactive web applications [3] to be specified. The proposed
approach uses the Web Behaviour Model (WBM) in combination with WebML
to form a high-level Event-Condition-Action (ECA) paradigm. WBM uses the
notion of timed finite state automatons to specify scripts that track the users’
navigation on the web site. When a WBM script reaches an accepting state, the
condition it represents is fulfilled and the corresponding actions in the form of
a WebML operation chain are executed as soon as the associated event occurs.
Based on this graphical ECA paradigm, applications such as profiling to infer
a user’s interests or updating specific values within the user model as well as
adapting to this information can be specified and implemented automatically
based on an intuitive model.
The Hera methodology [4] is a model-driven approach that integrates concepts from adaptive hypermedia systems with technologies from the semantic web. Faithful to its background of adaptive hypermedia systems, the specification of adaptation has always been an integral part of the Hera methodology [5]. Hera distinguishes between static design-time adaptation called adaptability and dynamic run-time adaptation called adaptivity. The design artefacts of all three models used in the development process can be adapted by annotating them with appearance conditions. Depending on whether the condition specifies an adaptability or adaptivity rule, they are evaluated during the generation step or at run-time. If a condition evaluates to true, the corresponding artefact will be presented to the user, otherwise it is omitted. Thus, alternatives can be specified using a set of mutually exclusive appearance conditions. Similar to the approach taken by WebML, web sites that have been designed with Hera are implemented by using the conceptual models to configure a run-time environment. The Hera Presentation Generator (HPG) [6] is an example of such a platform that combines the data stored as RDF with the models represented in RDFS to generate an adapted presentation according to user preferences as well as device capabilities. The presentation compiled by the Hera presentation generator is rendered as a set of static documents that contain the mark-up and the content for one particular class of clients. Hence, with this approach, it is only possible to implement appearance conditions that express design-time adaptability. More recently, an alternative implementation platform for Hera has been proposed based on the AMACONT [7] project. Based on a layered component-based XML document format [8], reusable elements of a web site can be defined at different levels of granularity. The document components that encapsulate adaptive content, navigation and presentation are then composed through aggregation and interlinkage into adaptive web applications. The proposed document format has three abstraction levels—media components, content unit components and document components—mirroring the iterative development process of most web sites. Adaptation is realised by allowing components of all granularities to have variants. A variant of a component specifies an arbitrarily complex selection condition as part of the metadata in its header. The decision as to whether a component is presented to the user is made by the XSLT stylesheet that generates the presentation according to the current context. AMACONT’s publishing process is based on a pipeline that iteratively applies transformations to a set of input documents to obtain the fully rendered output documents. Through the caching of partial results, intermediate steps can be reused multiple times leading to improved performance.

In UML-based Web Engineering (UWE) [9], adaptation is based on the Munich Reference Model [10] for adaptive hypermedia applications. The architecture and concepts of this reference model are based entirely on the previously discussed Dexter and AHAM reference models. However, while Dexter has been specified in Z without a graphical representation and AHAM has so far only been defined informally, the Munich Reference Model being written in UML offers both a graphical representation and a formal specification using the Ob-
ject Constraint Language (OCL). The model uses the same layering as Dexter—within-component, storage and run-time layers—and partitions the storage layer in the same way as AHAM into a domain, user and adaptation model. In contrast to the existing models, the Munich Reference Model distinguishes three forms of rule-based adaptation instead of two. To match the three layers of the UWE methodology, these forms of adaptation are adaptive content, adaptive links and adaptive presentation. A shortcoming of this rule-based approach is that the rules exist outside the model and thus have no graphical representation. A possible solution to this problem has been proposed through the use of aspects-oriented modelling techniques [11]. As adaptation logic is orthogonal to the basic application logic, the cross-cutting nature of aspects provides a promising solution for separating the two. By introducing the concept of aspects, the UWE metamodel has been extended to support adaptive navigation capabilities such as adaptive link hiding, adaptive link annotation and adaptive link generation.

So far, we have looked at the most influential conceptual models for web engineering and in some cases their proprietary implementation platform. Apart from those, general technologies to support context-awareness and adaptation haven been developed. An example of such a solution is the web authoring language Intensional HTML (IHTML) [12]. Based on version control mechanisms, IHTML supports web pages that have different variants and adapts them to a user-defined context. The concepts proposed by IHTML were later generalised to form the basis for the definition of Multidimensional XML (MXML) [13] which in turn provided the foundation for Multidimensional Semistructured Data (MSSD) [14]. Similar to semi-structured data that is often modelled using the Object Exchange Model (OEM), MSSD is represented in terms of a graph model that extends OEM with multidimensional nodes and context edges. In the resulting Multidimensional Object Exchange Model (OEM), multidimensional nodes capture entities that have multiple variants by grouping the nodes representing the facets. These variants are connected to the multidimensional node using context edges. In contrast to the conventional edges used in OEM, the label of a context edge specifies in which context the variant pointed to is appropriate. Using these specifiers, a MOEM graph can be reduced to a corresponding OEM graph for a given context. Based on this graph representation, a Multidimensional Query Language (MQL) [15] has been defined that allows the specification of context conditions at the level of the language. Thus, it can be used to formulate queries that process data across different contexts.

A general and extensible architecture that supports context-aware data access is proposed in [16]. Their approach is based on the concepts of profiles and configurations. Context is represented as a collection of profiles that each specify one aspect of the context such as the user, the device etc. Each profile contains a set of dimensions that capture certain characteristics and are associated to context values over attributes. Profiles are expressed according to the General Profile Model (GPM) [17] that provides a graphical notation and is general enough to capture a wide variety of formats currently in use to transmit context.
information as well as transforming from one format to another [18]. While such profiles describe the context in which a request has been issued to the web information system, configurations express how the response should be generated. A configuration has three parts that match the general architecture of web information systems in terms of content, structure and presentation. The content part of the configuration is represented by a query formulated in relational algebra. The definition of the structure part is expressed using WebML to define the hypertext. Finally, the presentation part is specified using the notion of a logical stylesheet which unifies languages such as Cascading Stylesheets (CSS).

Configurations are stored in a repository on the server side and matched to the profiles submitted by the client as part of its request. The matching is done based on adaptation rules consisting of a parametrised profile, a condition and a parametrised configuration [19]. The profile allows parameters instead of values to be used that are then assigned the values specified in the client profile. The condition constrains the values that are valid for the rule to be applied by formulating a logical expression over the parameters. Finally, the configuration includes the parameters value to adapt the content delivery. During the matching process, the client profile is compared to the adaptation rules. If the client profile matches the parametrised profile of the rule and the specified values fulfil the condition, the parametrised configuration is instantiated and applied.

3 Multi-Variant Objects

As presented in the previous section, most model-based approaches offer at least some support for context-aware web engineering. Some solutions even offer an integrated implementation platform tailored to the capabilities and requirements of the respective model. Most approaches, however, rely on standard components such as application servers, content management systems or relational databases to implement the modelled specifications. Unfortunately, as we will see, these implementation platforms do not provide native support for context-awareness. Therefore, this functionality has often to be implemented over and over again leading to poor reuse of code and maintainability. In this section, we will present multi-variant objects as an enabling concept for context-aware query processing in information systems.

Multi-variant objects have been specified within the framework of an object-oriented database management system developed at our institute. As this database management system is built on the concepts defined by the OM [20] model, we have decided to define our model as an extension of OM. OM is a rich and flexible object-oriented data model that features multiple instantiation, multiple inheritance and a bidirectional association concept. This model was chosen as, due to its generality, it is possible to use it to implement other conceptual models such as the Entity-Relationship (ER) model or the Resource Description Framework (RDF). Further, the feature of multiple instantiation, i.e. the ability of a single object to have multiple instances that exist on different paths along the inheritance graph, is something that is of frequent use in the domain of web
engineering. Imagine, for instance, a content management system that manages users who have certain roles in the administration of the web site. Based on these user roles, the types of the objects themselves will vary as they include different attributes and methods. In most object-oriented systems, this is usually modelled by defining the abstract concept of a user and then using inheritance to define concrete subtypes of this user. Most of these systems however do not provide a solution for the requirement that a user object needs to have two or more of these subtypes at the same time, whereas in reality users can have any number of roles, as someone can be, for example, both a site administrator and a content editor. In OM, the feature of multiple instantiation can be used to cater for exactly this kind of situation.

Figure 1: Conceptual data model of an object

Therefore, in the original OM model, an object is represented by a number of instances—one for every type of which the object is an instance. All instances of an object share the same object identifier but are distinguishable based on the type of which they are an instance. For the purpose of multi-variant objects, we have broken this relationship between the object and its instances and introduced the additional concept of a variant. As shown in the conceptual data model represented in Fig. 1, in the extended OM model, an object is associated with a number of variants which in turn are each linked to a set of revisions. Finally, each revision is connected to the set of instances containing the actual data. As can be seen from the figure, our model supports two versioning dimensions. Variants are intended to enable context-aware query processing while revisions support the tracking of the development process. However, for the scope of this paper we will focus on variants exclusively and neglect the presence of revisional versions in the model. Note that all versions of an object still share the same object identifier tying them together as a single conceptual unit. As in the traditional OM model, objects can be instantiated with multiple types and therefore both objects and variants can be related to any number of types. A variant of an object is identified by the set of properties associated with it. Any variant can have an
arbitrary number of properties, each consisting of a name and a value. Finally, instances are still identified based on their type. Hence they can only be linked to exactly one of the types to which the object is related. Further, instances are associated with values and thus contain the actual data of an object.

Before presenting how context-dependent queries are evaluated by our system, it is necessary to briefly introduce the notion and representation of context that we are using. In the setting of our context-aware information system, context information is regarded as optional information that is used by the system for augmenting the result of a query rather than specifying it. As a consequence, such a system also needs a well defined default behaviour that can serve as a fallback in the absence of context information. In our approach, context information is gathered outside the information system by the client application. Therefore, it is necessary that client applications can influence the context information that is used during query processing by the information system. To support this, a common context representation that is shared by both components is required. Since several frameworks for context gathering, management and augmentation already exist, our intention was to provide a representation that is as general as possible. Acknowledging the fact that each application has its own understanding of context, this representation is based on the concept of a context space $S$ that defines the names of the context dimensions that occur in an application. Each context dimension name can be associated with a value to form a context value $c = (\text{name}, \text{value})$. Then, a context $C(S)$ is a set of context values for the dimensions specified by $S$. Finally, a context space denoted by $C_\star(S)$ is a special context that contains exactly one value for every context dimension of $S$. While contexts are used to describe in which situation a particular variant of an object is appropriate, the current context state of the system governs how context-dependent queries are evaluated.

Context-aware queries over these multi-variant objects are evaluated using the matching algorithm shown in Fig. 2 to select the appropriate variants whenever objects are accessed by the query processor. The algorithm takes an object $o$ and the current context state of the system $C_\star(S)$ as inputs. First it retrieves all variants of $o$ that are linked to it through the HasVariants association. After building the context state of each variant from the properties that are associated to it through HasProperty, the algorithm applies a scoring function $f_s$ to this variant context state that returns a value measuring how appropriate the variant is in the current context. It then returns the variant of $o$ that has obtained the highest score $s_{\text{max}}$. However, if the highest score is below a certain threshold $s_{\text{min}}$ or if there are multiple variants with that score, the default variant is returned.

Similar to context dimensions, the concrete definition of the scoring function depends on the requirements of a context-aware application. Our system therefore allows the default scoring function to be substituted with an application-specific function. As it is not possible to discuss all issues involved in designing such a scoring function in the scope of this paper, we refrain from going into
MATCH\((o,C_*(S))\)
1 \(V_0 \leftarrow \text{rng}(\text{HasVariants}\ dr(\{o\}))\)
2 \(V_1 \leftarrow V_0 \propto (x \rightarrow (x \times \text{rng}(\text{HasProperty}\ dr(\{x\}))))\)
3 \(V_2 \leftarrow V_1 \propto (x \rightarrow (\text{dom}(x) \times f_s(C_*(S), \text{rng}(x))))\)
4 \(s_{\text{max}} \leftarrow \max(\text{rng}(V_2))\)
5 \(V_3 \leftarrow V_2 \% (x \rightarrow \text{rng}(x) = s_{\text{max}})\)
6 \[\text{if } |V_3| = 1 \land s_{\text{max}} \geq s_{\text{min}} \text{ then } v \leftarrow V_3 \text{ nth } 1\]
7 \[\text{else } v \leftarrow \text{rng}(\text{DefaultVariant}\ dr(\{o\})) \text{ nth } 1\]
8 \[\text{return } v\]

Figure 2: Matching algorithm

further detail. Nevertheless, we will give an intuitive understanding of its effect by means of examples in the next section.

4 Context-Aware Interactions

Based on the concept of multi-variant objects, we implemented a context-aware content management system that was used as the server component of a mobile tourist information system. The tourist information system was designed to assist visitors to the city of Edinburgh during the art festivals held each year during the month of August. A coarse overview of the architecture of the so-called EdFest system [21,22] is shown in Fig. 3. The range of clients that are supported by our system is shown on the left-hand side of the figure. Apart from traditional clients that are based on desktop PCs and PDAs, EdFest introduced a novel interaction channel based on interactive paper [23]. Our context-aware content management system is shown on the right-hand side of the figure. It consists of a web server that handles the communication with the clients, a server that manages publishing metadata [24,25] and an application database that stores the content of the EdFest application database. While the kiosk and PDA clients are implemented using standard HTML, the paper client actually consists of two publishing channels. The paper publisher [26] channel is used at design time to author and print the interactive documents from the content managed by the application database. The paper client channel is then active at run-time when the system is used by the tourists and is responsible for delivering additional information about festival venues and events by using voice feedback when the users interact with the digital pen on the interactive documents.

Of the four publishing channels, the paper and PDA client are mobile and have thus been integrated with the platform shown at the centre of the figure that manages various aspects of context. A range of sensors gather location, weather and network availability information that is then managed in a dedicated context database [27]. Context information is sent from the client to the server by encoding it in the requests sent to the content management server. This is one of the tasks of the client controller component. It acts as a transparent
proxy that intercepts requests and appends the current context state stored in the context database. Another task of this component is to act as a server on the client side, enabling the server to issue call-back requests to the client and thus allowing proactive behaviour to be implemented.

In this paper, we do not go into further details of the functionality offered by the EdFest system. A comprehensive description of the design and implementation of the system can be found in [28]. We instead focus on one particular functionality that demonstrates the need for context-aware interactions. The functionality we have chosen is the reservation process that allows tickets to be booked interactively. To understand what context-aware interactions are, Fig. 4 compares the interaction process of the prototype kiosk interface to the process on the paper client. At the top, Fig. 4(a) and (b) show the two different graphical user interfaces. The kiosk interface offers an HTML form with text fields in which the required information can be entered by the user. If all information has been entered, the data is sent to the server by clicking the submit button. With the paper client, the reservation process is quite different. Instead of entering all information and submitting the form with all data at once, a request is sent to the server for each parameter. The reason for this behaviour is that, in contrast to the web-based user interface, the tourist needs to be guided through the process by constant voice feedback. Also, this feedback serves as a confirmation of the data entered that could not be perceived by the user otherwise. Therefore, to book a ticket with the paper client, the tourist has to first start the reservation process by pointing to the icon labelled “Start reservation”. The system then instructs them to select the event for which they want to book tickets. This is done by selecting the event with the digital pen in a separate brochure listing all events. After an event has been selected, the number of tickets and the date are set in much the same way. The server checks if the booking is valid and, if so, sends a voice prompt to the client instructing the tourist to confirm the reservation by clicking on the icon labelled “reserve”. At the bottom, Fig. 4(c)
and (d) illustrate the communication pattern that results from reserving tickets using the kiosk and paper client, respectively. As can be seen in the figure, accessing the reservation process from the kiosk client results in two request and response pairs where the first retrieves the empty form and the second uploads all values to the server for processing. The picture in the case of the paper client is quite different as each data value required to process the reservation request is sent to the server encoded in an individual request. Additionally, the already selected values have to be managed in a session on the client and retransmitted with every request. 

Implementing the server-side application logic that handles the reservation process across multiple channels is a difficult task if the interaction patterns of the different channels are as heterogeneous as in the given example. In the EdFest system, our solution was inspired by the method dispatching strategies found in object-oriented programming languages. Many object-oriented languages allow
methods to be overloaded, i.e. support the definition of multiple versions of the 
same method with different sets of arguments. At run-time, they select the so-
called most specific method from the set of applicable methods based on the 
number and type of arguments given by the caller of the method. In its basic 
nature, virtual method dispatching is not unlike selecting the best matching vari-
ant of an object. All that has to be done to simulate method dispatching based 
on multi-variant objects is to define an object type that represents operations 
and treat the parameters specified by the client as context values.

Figure 5 gives a graphical representation of the multi-variant method object 
that was created to handle the setReservation process. As shown, for each 
context state that occurs in the process shown in Figure 4(d), a variant of the object 
is defined. As the context values that will be sent by the client cannot be known 
beforehand, the context states describing the variants use the value +* which 
indicates that a value for the corresponding context dimension has to be set but 
the actual value is not important. The default variant is responsible for starting 
the reservation process by generating a reservation number and initiates a ses-
sion on the client. All other variants of the object extract the provided context 
data, update the application database accordingly and send back a response that 
guides the visitor to the next step, except for variant o36995[5] that informs 
the tourists that they have completed the reservation process successfully.

![Figure 5: The setReservation object](image)

The kiosk reservation process only needs to access the default variant and 
the variant shown on the far right in the figure. In the case of the paper client, 
however, the reservation process runs through all variants of the objects before 
completing. An interesting aspect of implementing such processes is the way in 
which errors made by the user are handled. Interacting with the paper client, it 
is impossible to cause an error by entering incorrect values into the reservation 
process as all data is chosen from the pre-authored paper documents. The tourist 
can, however, deviate from the process by prematurely selecting parameters that 
will only be gathered in a later step. In this case, the value will nevertheless be 
stored in the client’s session but the response will be the same as before, asking 
the tourist to select the value corresponding to the current step. When this value 
is finally selected by the user, all steps that have been executed out of order are
skipped automatically as those values have already been stored in the session on the client.

While a tourist cannot deviate from the defined process in the web interface, it is possible to enter arbitrary values as well as to leave out certain parameters altogether. Hence, the system has to be able to additionally cope with these errors. The logic to check whether the form has been completed correctly by the user could be implemented on the client-side using embedded scripts. However, this solution is not generally possible on all required delivery channels as scripting capabilities, if present at all, vary substantially. Our approach to implementing process functionality based on an object with multiple variants is already able to handle cases where the tourist has failed to specify a required value. Even if they are not required in situations where the tourist fills in the form correctly, in the case of an error, the additional variants defined for the interactive paper process can be used for error handling in the kiosk interface. An omitted parameter will lead to the selection of one of these intermediate variants which will be rendered for the client as a form where the missing parameter is highlighted. Although context matching can provide a solution to missing values, it is not capable of addressing the problem of handling errors caused by incorrect data. To also implement this functionality, traditional parsing and error handling technique have to be applied.

5 Discussion

Using the ticket reservation process available in the EdFest system as an example, we have argued that interactive paper not only affects the way in which content is accessed and delivered but also the nature of information interaction. In EdFest, this problem was solved by creating context-aware operations that were realised based on multi-variant objects. Apart from the aspects already discussed, the interaction processes implemented for the interactive paper client have additional interesting characteristics. Looking back at the communication pattern between client and server given in Figure 4(d), a similarity to modern web applications can be observed. In order to prevent page reloads and provide immediate feedback to the user, many web sites nowadays use a technique called Asynchronous JavaScript and XML (AJAX). As indicated by its name, AJAX is a combination of existing technologies that are used together to provide more interactive web pages. In AJAX, a web page uses client-side scripting to connect to a server and to transmit values without reloading the whole page. At the time of opening the connection, a response handler is registered that will be invoked as soon as the request has been processed. Using JavaScript, the response handler can then update the web page asynchronously by accessing the Document Object Model (DOM) of the mark-up used to render the current page. Web applications based on AJAX communicate with the server at a finer level of granularity that is not unlike the interaction processes encountered on the paper client. The solution presented here to handle such processes could therefore form
The basis for integrating delivery channels that support AJAX with those that do not.

The use of context in this implementation raises an interesting question. We must ask ourselves whether it is sensible to apply the same mechanisms not only to data but also to programs. We have conducted preliminary research into this direction with the implementation of a prototype language that supports multi-variant programming [29]. The language is an extension of Prolog that allows predicate implementations to be defined for a given context state. The current context state of the system is managed by library predicates that allow context values to be set and removed. Before a context-aware Prolog program can be executed, it needs to be loaded by a special parser that replaces all predicate calls in the program with a call to a dispatching predicate that takes context into consideration. Experiences gained from a set of example programs have shown that the approach has its merits even though writing context-aware programs can be quite challenging, especially if context-dependent predicates are allowed to modify the context state. Naturally, our prototype implementation suffers from a few limitations and problems such as poor performance. Also, it is still unclear how to combine context-dependent predicate invocation with the backtracking mechanism of Prolog. Nevertheless, we believe that the potential benefits of this approach outweigh these challenges and will therefore continue to investigate the application of our version model to programming languages.

6 Conclusions

In this paper we have motivated the need for implementation platforms that allow context-aware applications to be implemented in a flexible and elegant way. Our approach proposes to extend information systems with the concept of multi-variant objects that form the basis for context-aware query processing. Based on this concept, we have implemented a context-aware content management system that has since been used to implement several web-based systems. The most ambitious system implemented so far is a mobile tourist information system targeted at visitors to the Edinburgh art festivals. Apart from traditional client devices, this EdFest system also supports a mobile paper-based client. In contrast to supporting conventional delivery channels where it is sufficient to adapt the content, structure and presentation, a paper-based interface also requires that the interaction process is adapted. As an example, we have discussed the implementation of the reservation process based on the EdFest interactive paper documents. In order to address the situation that the paper client requires a different communication pattern than traditional browser-based clients, we have created context-dependent interaction processes. Technically, these interaction processes were realised through different implementation variants of the database macro implementing the corresponding application logic. In this setting, context has been used to dispatch the request made by the client to the desired implementation similar to object-oriented programming languages.
dispatch a call to an overloaded method dispatching based on the parameters provided by the caller.

References


Web User Interface Migration through Different Modalities with Dynamic Device Discovery

Renata Bandelloni, Giulio Mori, Fabio Paternò, Carmen Santoro, Antonio Scorcia

ISTI-CNR, Via G. Moruzzi, 1
56124 Pisa, Italy
{renata.bandelloni, giulio.mori, fabio.paterno, carmen.santoro, antonio.scorcia}@isti.cnr.it

Abstract. In this paper we present a new environment for supporting Web user interface migration through different modalities. The goal is to furnish user interfaces that are able to migrate across different devices offering different interaction modalities, in such a way as to support task continuity for the mobile user. This is obtained through a number of transformations that exploit logical descriptions of the user interfaces to be handled. The new migration environment makes use of service discovery both for the automatic discovery of client devices and for the dynamic composition of the software services required to perform a migration request.

Keywords: User Interface Migration, Adaptation to the Interaction Platform, Ubiquitous Environments.

1 Introduction

One important aspect of pervasive environments is the possibility for users to freely move about and continue interacting with the services available through a variety of interactive devices (i.e. cell phones, PDAs, desktop computers, digital television sets, intelligent watches, and so on). However, continuous task performance implies that applications be able to follow users and adapt to the changing context of users and the environment itself. In practise, it is sufficient that only the part of an application that is interacting with the user, migrates to different devices.

In this paper, we present a new solution for supporting migration of Web application interfaces among different types of devices that overcomes the limitations of previous work [2] in many respects. Our solution is able to detect any user interaction performed at the client level. Then, we can get the state resulting from the different user interactions and associate it to a new user interface version that is activated in the migration target device. Solutions based on maintaining the application state on the server side have been discarded because they are not able to detect several user interactions that can modify the interface state. In particular, we present how the solution proposed has been encapsulated in a service-oriented architecture and supports Web interfaces with different platforms (fixed and mobile) and modalities (graphical, vocal, and their combination). The new solution also includes a discovery module, which is able to detect the devices that are present in the environment and...
collect information on their features. Users can therefore conduct their regular access to the Web application and then ask for a migration to any device that has already been discovered by the migration server. The discovery module also monitors the state of the discovered devices, automatically collecting their state-change information in order to understand if there is any need for a server-initiated migration. Moreover, we show how the approach is able to support migration across devices that support various interaction modalities. This has been made possible thanks to the use of a logical language for user interface descriptions that is independent of the modalities involved, and a number of associated transformations that incorporate design rules and take into account the specific aspects of the target platforms.

In the following section we discuss related work. Next, we provide an overall introduction of the environment, followed by a discussion on the logical descriptions used by the migration environment and how they are created by a reverse engineering process starting with the source desktop Web pages. Then, we provide the description of the semantic redesign module, explain how the migration environment functionalities have been incorporated, and describe the device discovery module. Lastly, we present an example application describing a migration through desktop, mobile and vocal, and draw some conclusions.

2 Related Work

The increasing availability of various types of electronic interactive devices has raised interest in model-based approaches, mainly because they provide logical descriptions that can be used as a starting point for generating interfaces that adapt to the various devices at hand. In recent years, such interest has been accompanied by the use of XML-based languages, such as UsiXML [5] and TERESA XML [7], in order to represent the aforementioned logical descriptions. The research in this area has mainly focused on how to help designers efficiently obtain different versions that adapt to the various interaction features, but contributions for runtime support have started to be proposed. For example, the Personal Universal Controller [8] automatically generates user interfaces for remote control of domestic appliances. The remote controller device is a mobile device, which is able to download specifications of the functions of appliances and then generate the appropriate user interface to access them. The architecture is based on a bidirectional asynchronous communication between the appliance and the remote controller. However, the process of discovering the device is far from automatic as the user needs to manually enter the device’s network address in the remote control application before any other action can be performed. ICrafter [11] is a more general solution for user interaction in ubiquitous environments, which generates adaptive interfaces for accessing services in such environments. In ICrafter, services beacon their presence by periodically sending broadcast messages. A control appliance then requests a user interface for accessing a service or an aggregation of services by sending its own description, consisting of the user interface languages supported (i.e. HTML, VoiceXML) to an entity known as the Interface Manager, which then generates the user interface and sends it back to the appliance. However, ICrafter does not support
the possibility of transferring the user interface from one platform to another, while the user is interacting with it, maintaining the client-side state of the interface. SUPPLE [4] generates adaptive user interfaces taking functional specifications of the interfaces, a device model and a user model as input. The remote solver server that acts as the user interface generator is discovered at bootstrap by the client devices, and they can thus request rendering of interfaces to it once it is discovered. However, discovery is limited to the setup stage of the system, and it does not monitor the runtime status of the system, thus loosing some of the benefits that could arise from a continuous monitoring activity. SUPPLE does not support the migration of a user interface from one device to another, but only adapts it to different types of platforms. Luyten and Coninx [6] present a system for supporting distribution of the user interface over a federation or group of devices. Migratability, in their words, is an essential property of an interface and marks it as being continuously redistributable. These authors consider migration and distribution of only graphical user interfaces, while we provide a new solution supporting graphic, vocal and even multimodal user interfaces migration. A general reference model for user interfaces aiming to support migration, distribution and adaptation to the platform is proposed in [1]. Our system, in particular, proposes a more concrete software architecture that is able to support migration of user interfaces, associated with Web applications hosted by different application servers, among automatically discovered devices.

3 Overall Description of the Environment

Our migration environment is based on a service-oriented architecture involving multiple clients and servers. We assume that the desktop version of the considered applications already exists in the application servers. In addition, we have a proxy service and the services of the migration platform, which can be hosted by either the same or different systems.

![Migration scenario](image)

**Fig. 1.** Migration scenario.
Figure 1 provides an overview of the system through an example from the user viewpoint. First, there is a dynamic discovery of the new user device as it enters the environment. Then, the environment may suggest migration to a nearby device (automatic migration), or the user can explicitly request a specific migration, e.g. by pointing with a RFID reader the projector (user-initiated migration), and the user interface migrates to the target device with the mediation of the migration server. In the example in Figure 1, the projector is associated with a PC, which will be considered the target device.

Fig. 2. Main communication among the migration services.

When a migration has to be processed, no matter which agent (the user or the system) starts the process, the Migration Manager, which acts as the main server module, retrieves the original version of the Web page that has to be migrated by invoking the HTTP Proxy service and retrieving the Web page(s) the user has accessed. Once the Web page(s) are retrieved, the Migration Manager builds the corresponding logical descriptions, at a different abstraction level, by invoking the Reverse Engineering service of our system. The result of the reverse engineering process is then used as input for the Semantic Redesigner service, in order to perform a redesign of the user interface for the target platform, taking into account the logical descriptions it received from the Migration Manager. Once the target Web pages have been generated, the Presentation Mapper service is used to identify the page which should be uploaded first into the target device.

In order to support task continuity throughout the migration process, the state resulting from the user interactions with the source device (filled data fields, selected items, ...) is gathered through a dynamic access to the DOM of the pages in the source device. Then, such information is associated to the corresponding elements of the newly generated pages and adapted to the interaction features of the target device.
by the State Mapper module, in a totally user-transparent way. Figure 2 represents with UML interaction diagrams the main communication among the migration services.

In order to allow for a good choice of the target device, the migration server retrieves and stores information about the devices that are automatically discovered in the environment. The collected information mainly concerns device identification and interaction capabilities. Such information allows, on the one hand, users to choose a target migration device with more accurate and coherent information on the available targets and, on the other hand, the system to suggest or automatically trigger migrations when the conditions for one arise. Thus, both the system and the user have the possibility to trigger the migration process, depending on the surrounding context conditions. A review of the different migration situations is described later on.

Users have two different ways of issuing migration requests. The first one is to graphically select the desired target device in their migration client. Users only have the possibility of choosing those devices that they are allowed to use and are currently available for migration. The second possibility for issuing migration requests occurs when the user is interacting with the system through a device equipped with an RFID reader. In this case, users could move their device near a tagged migration target and keep it close from it for a number of seconds in order to trigger a migration to that device. In this case a time threshold has been defined in order to avoid accidental migration, for example when the user is just passing by a tagged device. This second choice offers users a chance to naturally interact with the system, requesting a migration just by moving their personal device close to the desired migration target, in a straightforward manner.

Migration can also be initiated by the system skipping explicit user intervention in critical situations when the user session could accidentally be interrupted by external factors. For example, we can foresee the likelihood of having a user interacting with a mobile device that is shutting down because its battery power is getting too low. Such situations can be recognised by the system and a migration is automatically started to allow the user to continue the task from a different device, avoiding potential data loss.

Alternatively, the server can provide users with migration suggestions in order to improve the overall user experience. This happens when the system detects that in the current environment there are other devices that can better support the task being performed by the user. For example, if the user is watching a video on a PDA and a wide wall-mounted screen, obtained through connecting a projector to a desktop PC, is detected and available in the same room, the system will prompt the user to migrate to that device, as it could improve his performance. However, the user can continue to work with the current device and refuse the migration. Receiving undesired migration suggestions can be annoying for the user, thus users who want to receive such suggestions when better devices are available must explicitly subscribe to allow for this mixed-initiative migration activation service. In any case, once a migration has taken place, nothing prevents the user or the system from performing a new migration to another available device.
4 The User Interface Logical Descriptions Supported

Our migration service considers different logical views of the user interface, each one giving a different level of abstraction of the users’ interactions with the system:

- The task level, where the logical activities are considered.
- The abstract interface level, consisting of a platform-independent description of the user interface, for example at this level we can find elements such as a selection object.
- The concrete interface level, consisting of a platform-dependent but implementation language independent description of the user interface, for example in the case of a graphical user interface, the abstract selection object can become a radio button, a list or a pull-down menu.
- The final user interface, the actual implemented user interface.

The abstract description level represents platform-independent semantics of the user interface and is responsible for how interactors are arranged and composed together (this will also influence the structure of the final presentations).

Fig. 3. The specification of the abstract user interface language used in our migration approach
The concrete description represents platform-dependent descriptions of the user interface and is responsible for how interactors and composition operators are refined in the chosen platform with their related content information (text, labels, etc.). The concrete description adds information regarding concrete attributes to the structure provided by the abstract description. The abstract description is used in the interface redesign phase in order to drive the changes in the choice of some interaction object implementations and their features and rearrange their distribution into the redesigned pages. Both task and logical interface descriptions are used in order to find associations between how the task has been supported in the original interface and how the same task should be supported in the redesigned interface, and associate the runtime state of the migrating application. We have used TERESA XML for the logical description of the structure and interface elements [7]. The logical description of the user interface is organised in presentation(s) interconnected by connection elements (see Figure 3). Connections are defined by their source and target presentations, and the particular interactor in the source presentation in charge of triggering the activation of the target presentation. Presentations are made up of logical descriptions of interaction objects called interactor elements. Interactors are composed by means of composition operators. The goal of such composition operators is to identify the designers’ communication goals, which determine how the interactor should be arranged in the presentation. Thus, we have a grouping operator indicating that there is a set of elements logically connected to each other, a relation operator indicating that there is one element controlling a set of elements, a hierarchy operator indicating that different elements have different importance for users, and an ordering operator indicating some ordinal relation (such as a temporal relation) among some elements.

5 From Web Pages to Logical Descriptions

As we have introduced before, our environment exploits transformations that are able to move back and forth between various user interface description levels. Thus, reverse engineering support has been introduced. The main purpose of the reverse engineering part is to capture the logical design of the interface (in terms of tasks and ways to structure the user interface) and then use it as a key element to drive the generation of the interface for the target device. Some work in this area has been carried out previously. For example, WebRevEnge [9] automatically builds the task model associated with a Web application, whereas Vaquita [3] and its evolutions build the concrete description associated with a Web page. In order to support the automatic redesign for migration purposes, we need to access all the relevant abstract descriptions (the concrete, abstract and task level). Thus, the reverse engineering module of our migration environment is able to take Web pages and then provide any abstract logical descriptions, as needed.

Each page is reversed into a presentation, its elements associated with interactors or their composition operators. The reversing algorithm recursively analyses the DOM tree of the X/HTML page starting with the body element and going in depth. For each tag that can be directly mapped onto an interactor a specific function analyses the
corresponding node and extracts information to generate the proper interactor or composition operator. After the first generation step, the logical description is optimised by eliminating some unnecessary grouping operators (mainly groupings composed of one single element) that may result from the first phase. Then, according to the X/HTML DOM node analysed by the recursive function, we have three basic cases:

- The X/HTML element is mapped into a concrete interactor. This is a recursion endpoint. The appropriate interactor element is built and inserted into the XML-based logical description.
- The X/HTML node corresponds to a composition operator. The proper composition element is built and the function is called recursively on the X/HTML node subtrees. The subtree analysis can return both interactor and interactor composition elements. In any case, the resulting concrete nodes are appended to the composition element from which the recursive analysis started.
- The X/HTML node has no direct mapping to any concrete element. If the element has no child nodes, no action is taken and we have a recursion endpoint, otherwise recursion is applied to the element subtrees and each child subtree is reversed and the resulting nodes are collected into a grouping composition.

Each logical presentation can contain both elements that are the description of single interactor objects and composition operator elements. The composition operators can contain both simple interactors and multiple composition operators. Our reverse engineering transformation identifies the corresponding logical tasks [10]. This is useful for two main reasons: the interface on the target device should be activated at a point supporting the last basic task performed on the source device in order to allow continuity, and in the redesign phase it is important to consider whether the type of tasks to support is suitable for the target device.

6. Semantic Redesign for Different Types of Platforms

The redesign transformation aims at changing the design of a user interface. In particular, we propose a redesign for identifying solutions suitable for a different platform, which is performed automatically by exploiting semantic information contained in the logical description of the user interface (created by the reverse process). Given the limited resources in screen size of mobile devices (such as cell phones or PDAs), desktop presentations generally must be split into a number of different presentations for the mobile devices. The logical description provides us with some semantic information that can be useful for identifying meaningful ways to split the desktop presentations along with the user interface state information (the actual implemented elements, such as labels, images, etc.). The redesign module analyses the input from the desktop logical descriptions and generates an abstract and concrete description for the mobile device from which it is possible to automatically obtain the corresponding user interfaces. The redesign module also decides how
abstract interactors and composition operators should be implemented in the target mobile platform. In order to automatically redesign a desktop presentation for a mobile presentation we need to consider semantic information and the limits of the available resources. If we only consider the physical limitations we may end up dividing large pages into small pages which are not meaningful. To avoid this, we also consider the composition operators indicated in the logical descriptions. To this end, our algorithm tries to maintain interactors that are composed through some operator at the conceptual level in the same page, thus preserving the communication goals of the designer. However, this is not always possible because of the limitations of the platform, such as limited screen resolution. In this case, the algorithm aims at equally distributing the interactors into presentations of the mobile device. In addition, the splitting of the pages requires a change in the navigation structure with the need for additional navigator interactors that allow access to the newly created pages. More specifically, the transformation follows the subsequent main criteria:

- The presentation split from desktop to mobile takes into account the composition operators because they indicate semantic relations among the elements that should be preserved in the resulting mobile interface.
- Another aspect considered is the number and cost of interactors. The cost is related to the interaction resources consumed, so it depends on pixels required, size of the fonts and other similar aspects.
- The implementation of the logical interactors may change according to the interaction resources available in the target platform (for example an input desktop text area, could be transformed into an input mobile text edit or also removed, because writing of long text is not a proper activity for a mobile device).
- The connections of the resulting interface should include the original ones and add those derived from the presentation split.
- The images should be resized according to the screen size of the target devices, keeping the same aspect ratio. In some cases they may not be rendered at all because the resulting resized image is too small or the mobile device does not support them.
- Text and labels can be transformed as well because they may be too long for the mobile devices. In converting labels we use tables able to identify shorter synonyms.

In particular, the following rules have been applied for creating the new connections:

- Original connections of desktop presentations are associated to the mobile presentations that contain the interactor triggering the associated transition. The destination for each of these connections is the first mobile presentation obtained by splitting the original desktop destination presentation.
- Composition operators that are allocated to a new mobile presentation are substituted in the original presentation by a link to the new presentation containing the first interactor associated with the composition operators.
- When a set of interactors composed through a specific operator has been split into multiple presentations because they do not fit into a single mobile presentation, then we need to introduce new connections to navigate through the new mobile presentations.
In the transformation process we take into account semantic aspects and the cost in terms of interaction resources of the elements considered. We have defined for each mobile device class identified (large, medium or small) a maximum acceptable overall cost in terms of the interaction resources utilizable in a single presentation. Thus, each interactor and (even each composition operator) has a different cost in terms of interaction resources. The algorithm inserts interactors into a mobile presentation until the sum of individual interactor and composition operator costs reaches the maximum global cost supported. Examples of elements that determine the cost of interactors are the font size (in pixels) and number of characters in a text, and image size (in pixels), if present. One example of the costs associated with composition operators is the minimum additional space (in pixels) needed to contain all its interactors in a readable layout. This additional value depends on the way the composition operator is implemented (for example, if a grouping is implemented with a fieldset or with bullets). Another example is the minimum and maximum interspace (in pixels) between the composed interactors. After such considerations, it is easy to understand that each mobile presentation could contain a varying number of interactors depending on their consumption of interaction resources.

There are various differences to consider between graphical and vocal interfaces. In vocal interfaces there are several specific features that are important in order to support effective interaction with the user. For example, it is important that the system always provides feedback when it correctly interprets a vocal input and it is also useful to provide meaningful error feedback in the event of poor recognition of the user’s vocal input. At any time, users should be able to interrupt the system with vocal keywords (for example “menu”) to access other vocal sections/presentations or to activate particular features (such as the system reading a long text). An important aspect to consider is that sometimes users do not have an overall control of the system state, such as in graphic interfaces. In fact, short term memory can be easily disturbed by any kind of distraction. Thus, a useful technique is to provide some indication about the interface state in the application after a period of silence (timeout). Another useful technique for dealing with this problem can be the use of speech titles and welcome or location sentences in each vocal presentation to allow users to understand their position and the subject of the current presentation and what input the system needs at that point. Another important difference between speech and graphic interfaces is that the vocal platform supports only sequential presentations and interactions while the graphical ones allow concurrent interactions. Thus, in vocal interfaces we have to find the right balance between the logical information structure and the length of presentations. The analysis of the result of the reverse engineering provides useful information to understand how to organise the vocal version (for example what elements should be grouped) and then the arrangement is implemented using vocal constructs. When problems with long labels require looking up shorter synonyms then we use specific tables for them.

The main criteria of the redesign algorithm for the vocal platform are:

- Before redesign for vocal interaction, elements regarding tasks unsuitable for the vocal platform (for example, long text inputs) are removed and labels which are too long are modified (with the help of a database of terms suitable for vocal activities).
Semantic relations among interactors in the original platform are maintained in the vocal platform, keeping interactors composed through the same composition operators in the same vocal presentation, and implementing them with techniques more suitable for the vocal device.

Before redesign, images are removed and substituted by alternative descriptions (ALT tag).

Implementation of interactors and composition operators will change according to the resources of the new vocal devices.

The algorithm aims at providing a logical structure to vocal presentations avoiding too deep navigation levels because they may disorient users. To this end, only the highest level composition operators (in the case of nested operators) are used to split desktop presentations into vocal presentations.

Composition operators that are allocated to new vocal presentations are substituted in the main vocal presentation that cannot contain them by a vocal link to the new presentation, which contains the first interactor associated with the composition operator.

7. Device Discovery

Device discovery is another important aspect in migratory user interface environments. It allows the system to notice potential migration-source and migration-target devices. The technology that enables this discovery in our migration architecture is a custom discovery protocol explicitly created to handle service discovery, service description, and service state monitoring tasks at Internet Protocol level. The protocol is implemented as a module in the server, and as a client application on each of the devices. The design of this protocol provides multicast mechanisms for peer-to-peer device and service discovery, using well-known UDP/IP mechanisms. Once the module and the user devices have discovered each other, they make use of reliable unicast TCP/IP connections to provide service description and service monitoring primitives to the system. The implementation and use of the description capabilities of our discovery protocol provides means for the system to gather a rich set of information from the devices that are present in the environment, regarding both their interaction and communication capabilities as well as their general computational ones.

The device discovery of our migration infrastructure is based on multicast datagrams using UDP/IP. When one device enters the network, it issues a discovery (DIS) message to a well-known multicast address to which all existing devices must subscribe. Subscription to multicast groups is handled by the network equipment and usually limited to the current subnet. In any event, when this discovery (DIS) message is received by the other network peers, they respond by sending a unicast message (RES) to the issuer of the discovery (DIS) request. This way, all active migration clients and servers found in the network discover each other via a minimal exchange of network packets. At this point, the discovery algorithm changes depending on the nature of the migration application running on that particular device. If the device is to act as a server, then a unicast description request (DES) will be sent to all the
discovered devices, requesting them to discover themselves by sending an XML description file to the server device. This description will be saved in the server for future reference. If, on the other hand, the device is to act as a client to the migration infrastructure, then it will wait until a server is found and a description file is requested by it. Once this state is reached, the system is configured and fully functional. In order to guarantee consistency, keep-alive (DIS) multicast messages are sent by all parties with a periodicity of 1 second. When no keep-alive is received from a given device for a configurable amount of time, the device is deemed as having departed the network and no further communications are allowed with it until proof of its re-activation is gathered, in the manner of new multicast keep-alive messages. The periodicity of the keep-alive datagrams is low enough to ensure no considerable network traffic will be generated.

In order to supply the migration server with information about the devices that are present in the environment, XML-based device description files have been used. These files include all the relevant information the migration server needs in order to identify the device and find out its capabilities and features. The description files also provide an efficient way to monitor the state of the devices available in the environment by allowing the migration server and other interested parties to subscribe to certain events in order to receive a notification message each time a device state-change occurs. This has improved the support for automatic migration through richer and more accurate monitoring of the environment and the user interactions with it. The use of our custom discovery protocol in combination with these XML description files has proven to be successful and addresses our objectives and goals. In our new discovery-enabled prototype, users do not need to manually specify the IP address of the migration server, the middleware automatically discovers it for them. Neither do they need to login their personal interaction device into the migration environment, as their devices are automatically detected by the system both when connecting to it and when disconnecting from it. Thus, the new migration architecture offers an increased robustness and better consistency over the previous versions of our migration prototype, without increasing the prototype’s complexity from the development point of view, and keeping things transparent and simple for the end user.

8. Example Application

This section presents an example application of our migration environment. In the example, John is planning to go on vacation and would like to buy a new camera. He decides to search for a bargain on an online auction website and accesses the “e-Bid” website through his desktop PC. He checks the information about the available cameras by looking at item descriptions and prices. He finds an interesting offer and accesses the page containing information about the selected camera. He then decides to bid on this item, but discovers that he has to register first, and thus starts filling out the long registration form required by the website. Suddenly, the alarm on the desktop reminds him about a meeting which is going to take place this afternoon at his office, so he has to leave. The form is too long to be completed in time, thus he quickly
migrates the application to his PDA and goes out walking towards his car, while he continues filling in the form.

Figure 4 shows the desktop form partially filled in and how it is transformed for the mobile device if migration is triggered at that time. After the reverse engineering phase, the original desktop interface is transformed into a composition of concrete/abstract objects. Then, the composition operators (indicating a semantic relationship among the objects involved) and the number and cost of the interactors of the various presentations are considered in order to redesign the original desktop page for the new device. As a result of this process, the long desktop form is split into two pages or presentations for the PDA. Additional connections are inserted for handling the page splitting and allowing the user to navigate from/to the two pages. After completing the registration, John, with his PDA, places a bid on the camera before the auction ends in a matter of a few minutes, and then he is redirected to the page containing the camera description, where he can monitor the status of his bid.

**Fig. 4.** Example of migration through different devices.
While he is keeping an eye on the bidding, he enters his car and the application automatically migrates from the PDA to his mobile phone and can now be accessed through the vocal interface thanks to the wireless connection to the car voice system. Indeed, the environment carries out a redesign of the application for the new platform (vocal) and therefore identifies how the user interface design should be adapted for the new platform. Moreover, by identifying the point where the user was before the migration was activated, the environment is also able to update the new user interface with the data gathered from user so far, allowing the user not to start from scratch but continuing the interaction from the point where it was left off. Indeed, the speaker says “you have bid on item number 12345678 say item to hear the description of the item, price to know the current price, time to hear the ending time and bid to bid again on the item”. John says “price”. The car voice system replies “the price is 135 $, you have been outbid, if you want to make a new offer say bid followed by the new amount”, “bid 140 $”, “you are the highest bidder for item 12345678, time is ending in ten minutes, say continue if you want continue checking the item or exit if you want to exit the application”. John has reached the maximum amount of money he is willing to spend for the camera and thus says “exit” and continues driving towards his office. He will access the e-Bid website later on to check if he won the auction.

9. Conclusions

This paper has presented an environment supporting migration of user interfaces, even with different modalities. The implementation of the system, from the architectural point of view, follows a service-oriented architecture, with its corresponding benefits, both for the end user and for the developers of the system. The user interfaces that can be generated by the system are implemented using XHTML, XHTML Mobile Profile, VoiceXML and Java for the Digital TV. There are many applications that can benefit from migratory interfaces. In general, services that require time to be completed (such as games, booking reservations) or services that have some rigid deadline and thus need to be completed wherever the user is. Our environment is able to reverse engineer, redesign, and migrate Web sites implemented with XHTML and CSS. All the tags of these standards can be considered and manipulated.

An algorithm has been identified for handling code in Web pages that is implemented in different languages, for instance applets and Flash applications, which are generally identified by object tags with further attributes in their specification (e.g. title, etc.). The algorithm tries to map applets/flash elements to concrete (simpler) elements, taking into account the provided specification of such elements and also the capability of the target platform considered. For instance, if an applet/flash element has no siblings and there is no further data in its specification, the algorithm simply removes the corresponding node, otherwise it might map it into a e.g. textual string whose label is derived from the title attribute within the specification of the flash/applet code.
We are now working on a new version of our environment which is able to generate Microsoft C#-based user interface implementations, even supporting different modalities, such as gestural interaction.

We have conducted a number of preliminary user studies with the objective of analyzing the user perception of interface migration and we have found the first results to be encouraging. However, we plan to carry out a further user study in the nearby future to better measure the effectiveness of the resulting interfaces.

Acknowledgments
We thank Zigor Salvador for his help in the implementation of the dynamic device discovery part.

10 References

Adaptive Peer-to-Peer Web Clustering using Distributed Aspect Middleware (Damon) *

Rubén Mondéjar¹, Pedro García¹, Carles Pairot¹, and Antonio F. Gómez Skarmeta²

¹ Department of Computer Science and Mathematics, Universitat Rovira i Virgili
Avinguda dels Països Catalans 26, 43007 Tarragona, Spain
{ruben.mondejar, pedro.garcia, carles.pairot}@urv.cat
² Department of Computer Engineering, Universidad de Murcia
Apartado 4021, 30001 Murcia, Spain
skarmeta@fcu.um.es

Abstract. In this paper, we introduce the concept of adaptive peer-to-peer cluster and present our contributions on SNAP, a decentralized web deployment platform. In addition, we focus on the design and implementation of a load balancing facility by using the functionalities provided by our distributed AOP middleware (Damon). Using this approach, we are able to implement new mechanisms like decentralized session tracking and dynamic policies in a decoupled architecture. We believe that our model offers a novel approximation for modularizing decentralized crosscutting concerns in web environments.

1 Introduction

Nowadays, scalability and availability are two of WWW’s main challenges. Therefore, servers may stop serving requests if their network bandwidth is exhausted or their computing capacity is overwhelmed. One way to deal with the scalability problem is to have several identical servers and give the user the option to select among them. This approach is simple, but it is not transparent to the client. An alternative is to rely on an architecture that distributes the incoming requests among these servers in an unobtrusive way. A successful solution to this problem comes in the form of clustering or federation of servers. Following a distributed pattern, servers are made redundant so as when one becomes unavailable, another one can take its place.

Many important websites operate in this way, but these replicated server alternatives are normally expensive to achieve and maintain. As a matter of fact, the actual trend is to head towards decentralized models. These models take advantage of the computing at the edge paradigm, where resources available from any computer in the network can be used and are normally made available to their members. However, such architecture also introduces new issues which have to be taken care of. Some of these issues include how to deal with constant node joins and leaves, network heterogeneity, and many others. Moreover, another important issue is the development complexity of new applications on top of this kind of networks.

* This work has been partially funded by the European Union under the 6th Framework Program, POPEYE IST-2006-034241.
For these reasons, we need a **middleware platform** that provides the necessary abstractions and mechanisms to construct distributed applications. The Java Enterprise Edition (JavaEE) (formerly known as J2EE) architecture is a worldwide successful middleware alternative for development of distributed applications. Nevertheless, it feels tied to the client-server model. In this setting, Aspect Oriented Programming (AOP) presents an interesting solution to modulate crosscutting concerns on JavaEE environments [1]. Thus, our aim is to modify any JavaEE server behaviour so as it is able to work on a peer-to-peer (p2p) web cluster.

Thereby, we present a solution based on distributed AOP [2, 3, 4]. Specifically, we have developed a new approach to support decentralized JavaEE crosscutting concerns that includes: **session failover** for stateful applications, a complete **HTTP load-balancing** technique which permits dynamic client redirection to other servers, and a **runtime policy** system that defines node selection and workload distribution patterns. The advantages of our solution are as follows: a complete abstraction and decoupled design, and a transparent and generic interception server side scheme which is valid for any JavaEE servlet container implementation.

2 Adaptive p2pWeb Clustering

WWW is the most used technology in the Internet. Wide-area application development usually targets web environments. However, clients suffer non-desirable errors like “page is currently not available” or “resource is not accessible” due to server problems. On the other hand, p2p computing provides and shares resources efficiently among all network peers. As a consequence, it seems natural to merge standardized WWW wide-area applications with the benefits p2p has to offer. Until now, we have worked in this synergy of p2p and web technology with our p2pWeb model. In order to support web applications and services deployment and management, we have developed a p2pWeb platform called SNAP [5]. In this context, one of SNAP’s current limitations is in providing stateful applications. Specifically, the problems this paper tries to resolve are the use of front-side load balancing and the lack of distributed session tracking. Moreover, due to the complexity of JavaEE architecture it is difficult to add new functionalities or behaviours in a transparent way.

AOP permits to elegantly intercept and modularize crosscutting concerns in runtime. In addition, **distributed AOP** offers many interesting features for our p2p web cluster including monitoring and adaptability capabilities. In this way and making good use of structured p2p substrates and dynamic aspect frameworks, we have designed Damon [4], a fully decentralized aspect middleware built on top of a structured p2p overlay. Therefore, this work represents our first approach merging both concepts: p2p clustering and distributed AOP. As seen in Figure 1, we present an adaptive p2pWeb architecture where Damon enables transparent and distributed interception over the SNAP platform.

In this section, we explain the mechanisms to supply clustering issues, including distributed session failover, load balancing techniques, and runtime policies. In this line, we are focused on the inclusion of these features based on the separation of De-
centralized Crosscutting Concerns [4]. Thus, we also guarantee the necessary interdependence between the new aspectized mechanisms and the SNAP web server code. The real cohesion between SNAP and Damon is achieved with the corresponding pointcuts of the distributed aspects. Finally, these aspects are deployed on nodes that are running the SNAP applications that they aim to intercept.

![p2pWeb architecture diagram](image)

**Fig. 1.** p2pWeb architecture diagram.

### 2.1 Achieving Session Failover

In clustered environments, HTTP sessions from a web server are frequently replicated to other group servers. Session replication is expensive for an application server, because session request synchronization usually is a complex problem. Therefore, the initial issue we intend to solve is session tracking for stateful applications. Certainly, we need to use session migration when a node is shutdown (i.e. it crashes) or the load balancer decides to redirect the client to a different node.

Our solution is based on our Damon aspect framework, its persistence service and URL rewriting. For decentralized session persistence we use its ID to identify the session. This solution has a structural problem though, because session ID is only considered to be unique in the original host, but this is not applicable to whole network. Therefore, we need to change the session ID generator by means of intercepting the session creation code.

Once our session data is accessible throughout the network, we need a way to restore it whenever a new server becomes responsible for that client. The idea is to have meta-information that identifies the session directly embedded into the URL. This technique is known as **URL rewriting**. We mainly use URL rewriting to report the
client session ID to new server. URLs are modified before fetching the requested item, attaching the session ID like a usual request parameter. For instance: 
"http://server1:8080/dcalendar/calendar.jsp?JSESSIONID=08445a31a78661b5c746eff39a9db6e4e2cc5cf".

Fig. 2. Retrieve Session Damon Pointcut.

Figure 2 shows the retrieveSession pointcut. Before requests are made, the retrieveSession pointcut is executed. This pointcut checks whether the session ID is among the request parameters in order to restore previous session information from the network. If it is found, such remote session data is loaded into the new local server session.

Fig. 3. Load-Balancer Damon Pointcut.

2.2 Load Balancing Technique

A common approach chosen in these cases is known as front-side load balancing. This technique is to perform load balancing only at the beginning of a session and thereafter the connection between client and server is fixed, without any interaction with a load balancing instance anymore. Regarding SNAP’s initial load-balancer, it clearly follows a front-side based strategy using p2p locators [5]. For some applications, it can be adequate to bind clients to specific servers. However, such approxima-
tion has apparent limitations, such as what happens when the specific server the client is bound to stops working.

Trying to better improve SNAP’s load balancing algorithm, we intend to dynamically map load-balancers to web applications. These are implemented using a Damon aspect (see Figure 3). By means of the session tracking aspect session data can be effectively restored. Again, the attached session ID in the URL identifies the client among the servers. The loadBalancer pointcut intercepts any client requests and determines whether they are to be served or redirected to any other running application server. It also provides two new extension methods: isTimeToRedirect(String from) and getNewHost().

2.2.1 Load-Balancing Runtime Policies

In order to complete our system we provide policies designed to demonstrate the viability of our approach. Workload distribution in traditional web clusters is different from our decentralized system, because we do not have any centralized spot. Therefore existing centralized load balancing policies are to be modified to be efficient in our p2p system. In such case, we need to implement policies that perform well under heavy-loaded systems with highly variable workload.

There are many different algorithms which define the load distribution policy, among them are: random, Round-Robin, weight-based, least recently used (LRU), minimum load, threshold-based, local-space, last access time or other parameter-based policy. Following, we describe two examples of our implemented policies.

First, the Round-Robin is the most basic policy, although it is not the simplest, as it is the random one. We have implemented a decentralized version of the traditional Round-Robin policy where requests to each host are scattered throughout all hosts holding an application instance. Basically, it uses Damon reflection layer [4] to obtain the other instances, and chooses the next host after the previous one.

Secondly, we have also implemented the Least Recently Used (LRU) policy as well. In this policy, the host’s stress index is calculated as the average of server requests per second. Since we need to perform communication among other instances, messaging methods are used to distribute current server stress information.

2.3 Validation

We observe that the cost of our new concerns locally is directly produced by the aspect engine of our Damon framework’s implementation and it is similar to other evaluation results. We have as well conducted several experiments to measure the cost of our solution in a distributed scenario. In summary, we have mainly measured the system reaction to new policies activation and requests management.

The experiments were conducted on the PlanetLab [http://www.planet-lab.org] network, located in a wide variety of geographical locations, to measure the overhead of our system. Before each test, we estimated the average latency between hosts so as
to get a measure of how much overhead is incurred by the aspect activation and the following requests after the activation phase.

The values shown in Table 1 are the median of all the tests run. Each test was done using 500 random activations and advice calls for each pair of hosts. In conclusion, using the PlanetLab testbed, we verified the correct behaviour of the system and that Damon does not impose an excessive latency (the normalized incurred activation overhead is 3.27 and the one imposed by advice calls is 1.78).

Table 1. Overhead observed of runtime aspect activation tests in milliseconds

<table>
<thead>
<tr>
<th>Originator Host</th>
<th>Destination Host</th>
<th>Latency</th>
<th>Activations</th>
<th>Advice calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>planetlab2.urv.net</td>
<td>planetlab4.upc.es</td>
<td>10</td>
<td>97</td>
<td>35</td>
</tr>
<tr>
<td>planetlab-5.cs.princeton.edu</td>
<td>planetlab02.erin.utoronto.ca</td>
<td>73</td>
<td>214</td>
<td>103</td>
</tr>
<tr>
<td>planetlab1.scs.stanford.edu</td>
<td>bonnie.ibds.uka.de</td>
<td>180</td>
<td>449</td>
<td>260</td>
</tr>
<tr>
<td>planetlab02.dis.unima.it</td>
<td>planet1.manchester.ac.uk</td>
<td>45</td>
<td>192</td>
<td>122</td>
</tr>
<tr>
<td>planet1.cs.rochester.edu</td>
<td>planetlab-2.it.uu.se</td>
<td>108</td>
<td>409</td>
<td>220</td>
</tr>
</tbody>
</table>

3 Related Work

To the best of our knowledge, this work is the first approach in use distributed AOP in order to provide session tracking and load-balancing policies on a p2p web cluster. For this reason, we will basically focus this section on describing related work in more traditional solutions and in AOP approaches.

There exist different non-AOP solutions to introduce clustering in web environments. Regarding servlet filters and server mods, the Java Servlet specification version 2.3 introduces a new component type, called filter. A filter dynamically intercepts requests, before a servlet is reached. Responses are additionally captured after the servlet is left. There also exist a variety of server mods (like load-balancers) that directly depend on the server’s implementation. Usually, these mods are difficult to bind to a specific server. Finally, WADI [http://wadi.codehaus.org] aims to solve problems about the state propagation in clustered web servers. Thus, WADI provides several services useful for clustering on JavaEE platforms. Nevertheless, its main drawback is that it needs wrapping extensions for each different server’s implementation and forthcoming versions.

On the other hand, there are a number of non-distributed AOP solutions existent in the literature about concerns in JavaEE architectures [1, 6]. However, there are only a few distributed AOP solutions in complex systems and previous to Damon [4]. In [2], authors present a distributed AOP language called DjCutter. This language is the precursor of the remote pointcut concept. Remote pointcuts are similar to traditional remote method calls, which invoke the execution on a remote host. Nevertheless, these advices are executed in a unique and centralized host, thus making this solution inappropriate for dynamic systems. In this way, we can find more recent solutions like AWED [3] that solves many of these problems. AWED presents a more complete language for explicit aspect distribution. In our case, Damon is more like an abstracted and decoupled middleware that presents easier integration in this kind of scenarios.
4 Conclusions and Future Work

In this paper we have presented an adaptive p2p cluster using a distributed AOP approach. By recalling Figure 1, we observe that the novelty of this paper results in the two being merged into one cohesive solution. It is important to stress out that we could transparently swap SNAP p2p cluster by another web system thanks to uncohesion nature provided by Damon. Otherwise, we have designed the necessary mechanisms to allow stateful wide-area web applications: a session failover, an HTTP load-balancing mechanism, and a runtime policy system.

As a consequence, our solution aims to be as generic as possible thus supporting more dynamic environments. Our p2pWeb cluster, instead of being a traditional cluster with replicated servers; it is an effectively wide-area platform where each server holds different applications running on top of it. Moreover, we have designed our architecture using AOP which transparently intercepts the most significant servlet methods. By using such a solution we achieve more elegant, modular, and suitable mechanisms than traditional alternatives.

Finally, by means of our contributions, the client’s experience and usability when browsing p2p web applications is improved, since all fault tolerance and load balancing algorithms run in the background transparently. As a consequence, the client remains unaware of server changes due to overwhelming or failures, as its session state propagates from one to another.

To conclude, we pretend to further develop our adaptive middleware in mobile scenarios (Mobile Ad-hoc Networks, MANETS) within the POPEYE IST project [http://www.ist-popeye.eu].

References

Evolution of Web Applications with Aspect-Oriented Design Patterns

Michal Bebjak¹, Valentino Vranic¹, and Peter Dolog²

¹ Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technology
Slovak University of Technology,
Ilkovicova 3, 84216 Bratislava 4, Slovakia
bebjak02@student.fiit.stuba.sk, vranic@fiit.stuba.sk

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

Abstract. It is more convenient to talk about changes in a domain-specific way than to formulate them at the programming construct level or—even worse—purely lexical level. Using aspect-oriented programming, changes can be modularized and made reapplicable. In this paper, selected change types in web applications are analyzed. They are expressed in terms of general change types which, in turn, are implemented using aspect-oriented programming. Some of general change types match aspect-oriented design patterns or their combinations.

1 Introduction

Changes are inseparable part of software evolution. Changes take place in the process of development as well as during software maintenance. Huge costs and low speed of implementation are characteristic to change implementation. Often, change implementation implies a redesign of the whole application. The necessity of improving the software adaptability is fairly evident.

Changes are usually specified as alterations of the base application behavior. Sometimes, we need to revert a change, which would be best done if it was expressed in a pluggable way. Another benefit of change pluggability is apparent if it has to be reapplied. However, it is impossible to have a change implemented to fit any context, but it would be sufficiently helpful if a change could be extracted and applied to another version of the same base application. Such a pluggability can be achieved by representing changes as aspects [5]. Some changes appear as real crosscutting concerns in the sense of affecting many places in the code, which is yet another reason for expressing them as aspects.

This would be especially useful in the customization of web applications. Typically, a general web application is adapted to a certain context by a series of changes. With arrival of a new version of the base application all these changes
have to be applied to it. In many occasions, the difference between the new and
the old application does not affect the structure of changes.

A successful application of aspect-oriented programming requires a struc-
tured base application. Well structured web applications are usually based on the
Model-View-Controller (MVC) pattern with three distinguishable layers: model
layer, presentation layer, and persistence layer.

The rest of the paper is organized as follows. Section 2 establishes a scenario
of changes in the process of adapting affiliate tracking software used throughout
the paper. Section 3 proposes aspect-oriented program schemes and patterns
that can be used to realize these changes. Section 4 identifies several interesting
change types in this scenario applicable to the whole range of web applications.
Section 5 envisions an aspect-oriented change realization framework and puts
the identified change types into the context of it. Section 6 discusses related
work. Section 7 presents conclusions and directions of further work.

2 Adapting Affiliate Tracking Software: A Change
Scenario

To illustrate our approach, we will employ a scenario of a web application
throughout the rest of the paper which undergoes a lively evolution: affiliate
tracking software. Affiliate tracking software is used to support the so-called
affiliate marketing [6], a method of advertising web businesses (merchants) at
third party web sites. The owners of the advertising web sites are called af-
iliates. They are being rewarded for each visitor, subscriber, sale, and so on.
Therefore, the main functions of such affiliate tracking software is to maintain
affiliates, compensation schemes for affiliates, and integration of the advertising
campaigns and associated scripts with the affiliates web sites.

In a simplified schema of affiliate marketing a customer visits an affiliate’s
page which refers him to the merchant page. When he buys something from the
merchant, the provision is given to the affiliate who referred the sale. A general
affiliate tracking software enables to manage affiliates, track sales referred by
these affiliates, and compute provisions for referred sales. It is also able to send
notifications about new sales, signed up affiliates, etc.

Suppose such a general affiliate tracking software is bought by a merchant
who runs an online music shop. The general affiliate software has to be adapted
through a series of changes. We assume the affiliate tracking software is prepared
for the integration with the shopping cart. One of the changes of the affiliate
tracking software is adding a backup SMTP server to ensure delivery of the
news, new marketing methods, etc., to the users.

The merchant wants to integrate the affiliate tracking software with the third
party newsletter which he uses. Every affiliate should be a member of the newslet-
ter. When selling music, it is important for him to know a genre of the music
which is promoted by his affiliates. We need to add the genre field to the generic
affiliate signup form and his profile screen to acquire the information about the
genre to be promoted at different affiliate web sites. To display it, we need to
modify the affiliate table of the merchant panel so it displays genre in a new column. The marketing is managed by several co-workers with different roles. Therefore, the database of the tracking software has to be updated with an administrator account with limited permissions. A limited administrator should not be able to decline or delete affiliates, nor modify campaigns and banners.

3 Aspect-Oriented Change Representation

In the AspectJ style of aspect-oriented programming, the crosscutting concerns are captured in units called aspects. Aspects may contain fields and methods much the same way the usual Java classes do, but what makes possible for them to affect other code are genuine aspect-oriented constructs, namely: pointcuts, which specify the places in the code to be affected, advices, which implement the additional behavior before, after, or instead of the captured join point, and inter-type declarations, which enable introduction of new members into existing types, as well as introduction of compile warnings and errors.

These constructs enable to affect a method with a code to be executed before, after, or instead of it, which may be successfully used to implement any kind of Method Substitution change (not presented here due to space limitations). Here we will present two other aspect-oriented program schemes that can be used to realize some common changes in web application. Such schemes may actually be recognized as aspect-oriented design patterns, but it is not the intent of this paper to explore this issue in detail.

3.1 Class Exchange

Sometimes, a class has to be exchanged with another one either in the whole application, or in a part of it. This may be achieved by employing the Cuckoo’s Egg design pattern [8]. A general code scheme is as follows:

```java
public aspect ExchangeClass {
    public pointcut exchangedClassConstructor(): call(ExchangedClass.new(..);
    Object around(): exchangedClassConstructor() { return getExchangingObject();} ExchangeObject getExchangingObject() {
        if (...) {
            new ExchangingClass();
        } else {
            proceed();
        }
    }
}
```

The exchangedClassConstructor() is a pointcut that captures the ExchangedClass constructor calls using the call() primitive pointcut. The around advice captures these calls and prevents the ExchangedClass instance from being created. Instead, it calls the getExchangingObject() method which implements the exchange logic. ExchangingClass has to be a subtype of ExchangedClass.

\(^3\) Join points represent well-defined places in the program execution.
The example above sketches the case in which we need to allow the construction of the original class instance under some circumstances. A more complicated case would involve several exchanging classes each of which would be appropriate under different conditions. This conditional logic could be implemented in the getExchangingObject() method or—if location based—by appropriate pointcuts.

3.2 Perform an Action After an Event

We often need to perform some action after an event, such as sending a notification, unlocking product download for user after sale, displaying some user interface control, performing some business logic, etc. Since events are actually represented by method calls, the desired action can be implemented in an after advice:

```java
public aspect AdditionalReturnValueProcessing {
    pointcut methodCallsPointcut(TargetClass t, int a): ... ;
    after(∗ captured arguments ∗): methodCallsPointcut(∗ captured arguments ∗) {
        performAction(∗ captured arguments ∗);
    }
    private void performAction(∗ arguments ∗) { ∗ action logic ∗ }
}
```

4 Changes in Web Applications

The changes which are required by our scenario include integration changes, grid display changes, input form changes, user rights management changes, user interface adaptation, and resource backup. These changes are applicable to the whole range of web applications. Here we will discuss three selected changes and their realization.

4.1 Integration Changes

Web applications often have to be integrated with other systems (usually other web applications). Integration with a newsletter in our scenario is a typical example of one way integration. When an affiliate signs up to the affiliate tracking software, we want to sign him up to a newsletter, too. When the affiliate account is deleted, he should be removed from the newsletter, too.

The essence of this integration type is one way notification: only the integrating application notifies the integrated application of relevant events. In our case, such events are the affiliate signup and affiliate account deletion. A user can be signed up or signed out from the newsletter by posting his e-mail and name to the one of the newsletter scripts. Such an integration corresponds to the Perform an Action After an Event change (see Sect. 3.2). In the after advice we will make a post to the newsletter sign up/sign out script and pass it the e-mail address and name of the newly signed up or deleted affiliate. We can seamlessly combine multiple one way integrations to integrate a system with several systems.
Introducing a *two way integration* can be seen as two one way integration changes: one applied to each system. A typical example of such a change is data synchronization (e.g., synchronization of user accounts) across multiple systems. When the user changes his profile in one of the systems, these changes should be visible in all of them. For example, we may want to have a forum for affiliates. To make it convenient to affiliates, user accounts of the forum and affiliate tracking system should be synchronized.

### 4.2 Introducing User Rights Management

Many web applications don’t implement user rights management. If the web application is structured appropriately, it should be possible to specify user rights upon the individual objects and their methods, which is a precondition for applying aspect-oriented programming.

User rights management can be implemented as a Border Control design pattern [8]. According to our scenario, we have to create a restricted administrator account that will prevent the administrator from modifying campaigns and banners and decline/delete affiliates. All the methods for campaigns and banners are located in the campaigns and banners packages. The appropriate region specification will be as follows:

```java
pointcut prohibitedRegion(): (within(application.Proxy) && call(void *,*(..)))
    || (within(application.campaigns.+) && call(void *,*(..)))
    || within(application.banners.+) && call(void Affiliate.decline(..)) || call(void Affiliate.delete(..));
```

Subsequently, we have to create an around advice which will check whether the user has rights to access the specified region. This can be implemented using the Method Substitution change applied to the pointcut specified above.

### 4.3 Introducing a Resource Backup

As specified in our scenario, we would like to have a backup SMTP server for sending notifications. Each time the affiliate tracking software needs to send a notification, it creates an instance of the SMTPServer class which handles the connection to the SMTP server and sends an e-mail. The change to be implemented will ensure employing the backup server if the connection to the primary server fails. This change can be implemented straightforwardly as a Class Exchange (see Sect. 3.1)

### 5 Aspect-Oriented Change Realization Framework

The previous two sections have demonstrated how aspect-oriented programming can be used in the evolution of web applications. Change realizations we have proposed actually cover a broad range of changes independent of the application
domain. Each change realization is accompanied by its own specification. On the other hand, the initial description of the changes to be applied in our scenario is application specific. With respect to its specification, each application specific change can be seen as a specialization of some generally applicable change. This is depicted in Fig. 1 in which a general change with two specializations is presented. However, the realization of such a change is application specific. Thus, we determine the generally applicable change whose specialization our application specific change is and adapt its realization scheme.

![Fig. 1. General and specific changes with realization.](image)

When planning changes, it is more convenient to think in a domain specific manner than to cope with programming language specific issues directly. In other words, it is much easier to select a change specified in an application specific manner than to decide for one of the generally applicable changes. For example, in our scenario, an introduction of a backup SMTP server was needed. This is easily identified as a resource backup, which subsequently brings us to the realization in the form of the Class Exchange.

## 6 Related Work

Various researchers have concentrated on the notion of evolution from automatic adaptation point of view. Evolutionary actions which are applied when particular events occur have been introduced [9]. The actions usually affect content presentation and navigation. Similarly, active rules have been proposed for adaptive web applications with the focus on evolution [4]. However, we see evolution as changes of the base application introduced in a specific context. We use aspect orientation to modularize the changes and reapply them when needed.

Our work is based on early work on aspect-oriented change management [5]. We argue that this approach is applicable in wider context if supported by a version model for aspect dependency management [10] and with appropriate aspect model that enables to control aspect recursion and stratification [1]. Aspect-oriented programming community explored several specific issues in software evolution such as database schema evolution with aspects [7] or aspect-oriented extensions of business processes and web services with crosscutting concerns of reliability, security, and transactions [3]. However, we are not aware of any work aiming specifically at capturing changes by aspects in web applications.
7 Conclusions and Further Work

We have proposed an approach to web application evolution in which changes are represented by aspect-oriented design patterns and program schemes. We identified several change types that occur in web applications as evolution or customization steps and discussed selected ones along with their realization. We also envisioned an aspect-oriented change realization framework.

To support the process of change selection, the catalogue of changes is needed in which the generalization-specialization relationships between change types would be explicitly established. We plan to search for further change types and their realizations. It is also necessary to explore change interactions and evaluate the approach practically.

Acknowledgements The work was supported by the Scientific Grant Agency of Slovak Republic (VEGA) grant No. VG 1/3102/06 and Science and Technology Assistance Agency of Slovak Republic contract No. APVT-20-007104.

References

Adaptive portal framework
for Semantic Web applications

Michal Barla, Peter Bartalos, Mária Bieliková,
Roman Filkorn, Michal Tvarožek

Institute of Informatics and Software Engineering, Faculty of Informatics
and Information Technologies, Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
{Name.Surname}@fiit.stuba.sk

Abstract. In this paper we propose a framework for the creation of
adaptive portal solutions for the Semantic Web. It supports different
target domains in a single portal instance. We propose a platform envi-
ronment where the ontology models and adaptivity are among first-class
features. Adaptivity is supported by the personalized presentation layer
that integrates software tools for automatic user characteristic acquisi-
tion. A significant contribution of the design lies in our method for au-
tomatic form building from the domain ontology and automated CRUD
pattern support by object-ontology mapping. We evaluate the framework
in two domains – online labor market and scientific publications.

1 Introduction

Many current information systems need a suitable way of communicating with
users by means of a user-friendly (graphical) user interface. Consequently, many
systems adopted a web-based user interface, which can be accessed via a thin
client, such as a generic web browser. This introduces new challenges since the
architecture, design and overall approach to engineering of web-based applica-
tions differs from traditional desktop thick client applications.

While typical web applications offer specific services to users, web portal so-
lutions aim to provide a single point of access for personalized services, informa-
tion sharing and collaboration support. Furthermore, portals serve as gateways
to other content and services provided either locally or more often as distributed
applications. Thus, system integration plays a very important role where interop-
erability is becoming paramount. Unlike traditional desktop applications, web
portals often employ a diverse range of middleware, specialized methods and
tools to integrate and process information from various sources. Consequently,
portal solutions strive not only for maximal flexibility and variability but also for
shared semantics to which the Semantic Web principles may be applicable [10].
These however are not yet supported by the state of the art portal frameworks.

Web-based information systems in general, and portal systems in particular
can also be viewed from the client users’ perspective where the overall design,
functionality and user-friendliness of the user interface is important. Personalized
approaches such as adaptive hypermedia have been proposed to solve common problems like the “lost in hyperspace” syndrome and information overload, while social approaches were proposed for collaboration and information sharing.

In this paper we propose a framework for the creation of adaptive web-based portal solutions for integrating different web applications. We strongly focus on reusability, component-based design, personalization and interoperability taking advantage of ontologies, adaptive navigation and presentation.

2 Related work

There are many commercial quality portal solution products from a variety of top rank software vendors, e.g. Microsoft SharePoint, Sun Java System Portal Server, IBM WebSphere or BEA WebLogic. Although definitely varying in specific technology particulars, the list of supported features and off-the-shelf components is overwhelming. They offer consistent solutions and share many characteristics such as security, enterprise information and services integration, documentation, steep learning curve, ease and comfort use and administration. Similarly, open-source Apache Foundation projects such as Cocoon, Struts, Tapestry or Jetspeed are examples of commonly used, technically mature, reusable portal, albeit less sophisticated, frameworks for fast web application development.

Some state of the art methods in web application development, based on model drive approaches, include HERA [14], WebML [8], SHDM/OOHDM [9], UWE [5]. Ideally, these are aimed at designing web applications, which are well understood and where the respective models can be (easily) defined. However, they do not directly address the integration and common aspects of different distributed web applications and/or data sources into a single portal instance.

The idea of using ontologies in portal solutions for the Semantic Web has already been examined in several works. OntoPortal uses ontologies and adaptive hypermedia principles to enrich the linking between resources [4]. The AKT project aims to develop technologies for knowledge processing, management and publishing, e.g. OntoWeaver-S [6], which is a comprehensive ontology-based infrastructure for building knowledge portals with support for web services.

The SEAL [11] framework for semantic portals takes advantage of semantics for the presentation of information in a portal with focus on semantic querying and browsing. The semantic web portal tool OntoViews [7] is designed for publishing of RDF content on the web and provides the user with a content-based search engine and link generation/recommendation based on relationships between ontological concepts. SOIP-F [13] describes a framework for the development of semantic organization information portals based on “conventional” web frameworks, web conceptual models, ontologies as well as additional metadata.

A lot of work has already been done in the field of semantic web portals. Existing approaches take extensive advantage of ontologies, web services and different navigation and presentation models. However, while support for personalization (via presentation adaptation to user context) was already addressed in some approaches, they do not offer fully automatic semantic user action logging. Our
approach takes advantage of semantic server-side logging which supports and augments the successive user characteristic estimation.

Furthermore, issues concerning the evolution of open information spaces should be addressed with respect to effective portal development and maintenance with the aim of reducing workload when developing new portal solutions or maintaining existing ones in changing environments. Our automated support of CRUD patterns contributes to this issue.

3 Adaptive portal solution architecture

The proposed framework for the creation of adaptive web-based portal solutions has two major goals: to be able to support different target domains in a single portal instance, and to set up a platform environment where the ontology models and adaptivity will be among first-class features. A portal created using the framework stands as an integration platform for different existing or newly developed web applications, which are available via a single access point, and which can be either independent or interconnected.

A target domain is represented by a domain specific model. It captures and specifies its characteristic concepts, structures, relations, behavior and constraints. In order to easily change models one has to focus on a meta-model. Ontologies constitute a way to manage and process the model and its meta-model in a consistent and uniform way. While being able to manipulate the entities at the instance level of the ontology, the inference mechanisms may take both levels in consideration and the result may improve and alter either the model or the meta-model of the particular target domain.

From this point of view, even two consecutive versions of the same ontology may be considered as two different models and a suite of tools and inference rules may be able to process the data between these two instances. In such a way our framework is able to adapt to changes to its own (meta-)model.

Our design reflects the following requirements:

- Adaptivity and adaptability of the system’s presentation and functionality.
- Built-in automatic user modeling based on user action logging with semantics and automatic user characteristic estimation.
- Reusability and generic design suitable for multiple application domains.
- Extensibility with additional tools for specific tasks and overall flexibility with respect to tool orchestration.
- Tolerance towards changes in the domain and user ontologies.

In our design we take advantage of MVC-based frameworks, component-based web development and XML processing, which are based on the pipes and filters architectural pattern, what makes them specifically suitable for RDF/RDFS and OWL processing. One such framework is the open-source web development framework Apache Cocoon (http://cocoon.apache.org/), which we used as the underlying portal framework for our solution. Figure 1 depicts an overview
of the portal architecture that extends the basic functionality of Cocoon with additional software components in order to fulfill the aforementioned requirements.

**Corporate Memory.** We store data in the *Corporate Memory* repository which stores the domain, user and event ontologies (Figure 1, bottom). We use a domain ontology to capture and formally specify domain specific data – concepts, structures, relations, behavior and constraints characteristic for a particular application domain. A user ontology is derived from the domain ontology to define users, their characteristics and preferences towards specific domain concepts. We employ an event ontology to capture the semantics of user actions during system operation for their successive processing in the user modeling process.

**Cocoon extensions.** The core Cocoon extensions include (Figure 1, center):

- *User Management* used for creating and altering of user accounts.

---

![Diagram of the portal architecture](image-url)
– *Common Configuration* of individual tools, which is used to access data in the *Corporate Memory* repository.
– *Security*, which ensures that only authorized users access protected resources.
– *Coplet Management* used to customize the overall portal interface, i.e. to add, remove or edit the layout and use of individual coplets (i.e., Cocoon servlets corresponding to specific GUI parts) and skins.

**CRUD support.** A significant contribution of our design is the *CRUD* component. It supports form generation from the domain ontology and the automated *CRUD* pattern (Figure 1, top left) as means of improving reusability for different application domains. *CRUD* organizes the persistence operations of an application into Create, Retrieve, Update and Delete operations that are implemented by a persistence layer, and includes the generation of form descriptions for Cocoon (*Form Generator*), the generation of the underlying JavaBeans (*Bean Generator*) and the associated mapping and persistence of JavaBeans (*Ontology-Object Mapper*) in the ontological repository [2].

**User characteristics acquisition.** We employ the personalized presentation layer architecture proposed in [12] that facilitates *User Characteristic Acquisition* – a two stage process consisting of sever-side and client-side *User Action Logging* and *User Modeling*. The process takes advantage of a set of software tools, integrated into the portal framework, that form a configurable user modeling chain which transforms user actions into a user model that can be used by all tools integrated in the portal.

The *User Action Logging* stage produces logs with semantics which are processed using a rule-based approach [1] resulting in user characteristics stored in an ontology-based user model. Every presentation tool in the portal is responsible for the logging of its respective events and their semantics by means of a common logging service and thus contributes to user characteristics acquisition process.

**Presentation.** The *Portal* tool is used to aggregate output from individual adaptive *Presentation tools*, which support adaptation based on user context, and assist in the creation of comprehensive user action logs. The user context itself contains different types of data, e.g., a user model describing user characteristics and an environment model describing the client device or client connection.

In particular, we utilize *Form Presentation* tools that take advantage of *CRUD* pattern support to provide users with personalized form filling functionality for specific domain concepts. For navigation in the domain ontology we use an adaptive faceted browser and a cluster navigation tool that supports visual navigation in clusters of domain concepts. We also employ several search tools that allow the user to specify different search criteria and ranking algorithms.
4 Evaluation

We successfully employed the proposed portal solution in two projects dealing with different domains. Using the framework we created a portal Job Offer Portal (JOP) used in research project NAZOU (http://nazou.fiit.stuba.sk) in the domain of online job offers. JOP offers its users several ways of navigation through the information space using different presentation tools, which work with the ontological database produced by a chain of data harvesting tools that acquire and process data from the Internet [3].

The whole system utilizes multiple data processing chains. Starting with the data sources, users can submit new job offers using a set of forms generated by the framework. On the other hand, a set of automatic wrapping and web crawling tools collects (structured) documents (tools WrapperGenerator, WebCrawler, RIDAR – Relevant Internet Data Resource Identification). To support information retrieval, approaches like clustering (tools Clusterer, ASPECT – Probabilistic document clustering) and criteria and top-k ordering (tools CriteriaSearch, SQEx – Semantic Query Expansion Tool, TopK aggregator, IGAP – Induction of Generalized Annotated Programs) are employed. The data and search results presentation is performed by JOP – the primary adaptable user interface, which integrates individual presentation and user modeling tools (tools Factic - Faceted browser and ClusterNavigator) and user modeling (tools Click, LogAnalyser, SemanticLog).

Another portal, called Publication Presentation Portal (P3) was created in research project MAPEKUS (http://mapekus.fiit.stuba.sk). It uses metadata about scientific publications downloaded from digital libraries and aids users in finding relevant ones by adapting the presented information.

Both created portals use ontology-based back-end and user modeling features provided by our portal framework. Both stand for an integration platform for various domain-specific tools and data processing workflows. The features of common ontologies and adaptivity significantly improve their overall quality.

5 Conclusion

We described the design of a framework for the creation of adaptive web-based portal solutions with support for both adaptability and adaptivity.

We take advantage of component-based design and built a working portal from a set of interconnected software tools that perform specific tasks. Furthermore, we employ ontologies in order to incorporate semantics shared across individual tools, data and metadata into the respective domain and user models in a consistent and uniform way. In this way our solution supports different target domains in single portal instance.

The automated form generation from the domain ontology and object-ontology mapping contributes to the flexibility and the easy reuse of the solution. Using these components we can flexibly react to domain ontology changes by changing the corresponding parts of the application automatically.
Acknowledgment. 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.

References

An approach to support the Web User Interfaces evolution

Preciado, J.C.; Linaje, M.; Sánchez-Figueroa, F.

Quercus Software Engineering Group
Escuela Politécnica. Universidad de Extremadura (10071 – Cáceres, Spain)
{jcpreciado; mlinaje; fernando}@unex.es

Abstract. Currently, there is a growing group of Web 1.0 applications that is migrating towards Web 2.0 where their data and business logic could be maintained but their User Interface (UI) must be adapted. Our proposal facilitates the adaptation of existing Web 1.0 applications to Web 2.0, focusing on UIs and taking advantage of functionality already provided by legacy Web Models. In this paper we present, as an example, how to adapt applications already modelled with WebML to RUX-Model, a method that allows designing rich UIs for multi-device Web 2.0 UIs. One of our main goals in attending the workshop is discussing other potential adaptations for applications modelled with OOHDM, UWE or OO-H among others.

Keywords: Web Engineering, Adaptation, User Interfaces, Web 1.0, Web 2.0, Rich Internet Applications.

1 Introduction

Over the past few years, the traditional HTML-based Web Applications (Web 1.0) development has been supported by different models and methodologies coming from the Web Engineering community.

Nowadays, the complexity of activities performed via Web User Interfaces (UIs) keeps increasing and ubiquity becomes fundamental in a growing number of Web applications. In this context, many Web 1.0 applications are showing their limits to reach high levels of interaction and multimedia support, so many of them are migrating towards Web 2.0 UIs.

The majority of the features of Web 2.0 UIs may be developed using Rich Internet Applications (RIAs) technologies [8] which combine the benefits of the Web distribution model with the interface interactivity and multimedia support available in desktop applications. UI development is one of the most resource-consuming stages of application development [2]. A systematic development approach would decrease resource usage. However, there is still a lack of complete models and methodologies related with RIA [1]. An interesting partial proposal can be found in [11].

Our statement is that it is not only important to develop applications for Web 2.0 from scratch, but it is also important to adapt existing Web 1.0 applications based on Web Models to the new requirements and necessities following a methodology.
In this paper we use RUX-Model (Rich User eXperience Model) [9], a Model Driven Method for engineering the adaptation of legacy Web 1.0 applications to Web 2.0 UI expectations [8]. A case study is presented using the RUX-Model CASE Tool (RUX-Tool, available at http://www.ruxproject.org), which supports RIA UI code generation and it is validated by implementation.

RUX-Model proposes three UI transformation phases that we describe in Section 2. However, the main contribution of this paper focuses on the definition of the connection with the Web Model to be adapted (Section 3).

2 RUX-Model Overview

RUX-Model is an intuitive visual method that allows designing rich UIs for RIAs and its concepts are associated with a perceptive graphical representation. In the context of adapting existing Web 1.0 applications to Web 2.0, RUX-Model can be seen as an adaptor as it is depicted in Figure 1 (left). Due to it being a multidisciplinary proposal and in order to decrease cross-cutting concepts, the UI specification is divided into levels. According to [3] an interface can be broken down into four levels, Concepts and Tasks, Abstract Interface, Concrete Interface and Final Interface. The RUX-Model process starts from Abstract Interface and each Interface level is composed by Interface Components. Concepts and Tasks are taken by RUX-Model from the underlying legacy Web Model.

Abstract Interface provides a UI representation common to all RIA devices and development platforms without any kind of spatial arrangement, look&feel or behaviour, so all the devices that can run RIAs have the same Abstract Interface. Abstract Interface elements are:

- **Connectors**, we have included them to establish the relation to the data model once the hypertext model specifies how they are going to be recovered;
- **Media**, they represent an atomic information element that is independent of the client rendering technology. We have categorized media into discrete media (texts and images) and continuous media (videos, audios and animations). Each media gives support to Input/Output processes;
- **Views**, 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.

Then in Concrete Interface we are able to optimize the UI for a specific device or group of devices. Concrete Interface is divided into three Presentation levels: Spatial, Temporal and Interaction Presentation. Spatial Presentation allows the spatial arrangement of the UI to be specified, as well as the look&feel of the Interface Components. Temporal Presentation allows the specification of those behaviours which require a temporal synchronization (e.g. animations). Interaction Presentation allows modelling the user’s behaviour with the UI.

The RUX-Model process ends with Final Interface which provides the code generation of the modelled application. This generated code is specific for a device or a group of devices and for a RIA development platform and it is ready to be deployed. RUX-Model adaptation process from Web 1.0 applications to Web 2.0 has three different transformation phases. Figure 1 (left) shows the different interface levels and
transformation phases. The first transformation phase (Connection Rules), marked as 1 in the figure, is automatically performed and extracts all the relevant information from the previous Web Model to build a first version of Abstract Interface. Then, the second phase is performed, marked as 2 in Figure 1 where Concrete Interface is automatically obtained from Abstract Interface. Finally, in the third phase, marked as 3 in the figure, Final Interface is automatically generated depending on the chosen RIA rendering technology (e.g., Lazslo, Flex, Ajax, XAML). Phases 1 and 2 can be improved by modellers to achieve their goals according to their needs.

Fig 1. Left: RUX-Model architecture overview; Right: Example of RUX-Model method

In Figure 1 (right) we show the Interface levels and the transformation phases, but here from a practical point of view. In this figure, RUX-Model obtains Abstract Interface automatically by means of a connection to an existing Web application developed using a Web Model (e.g., WebML [4]). This Abstract Interface is transformed into two Concrete Interfaces, one with a special arrangement adapted for PCs and the other for a group of mobile devices. One Concrete Interface is transformed into two Final Interfaces for a PC using different technologies (one uses Lazslo and the other uses AJAX) and the other Concrete Interface into two Final Interfaces for similar devices (e.g., PDA and Smartphone) using Lazslo rendering technology.
3 Web Model to RUX-Model Adaptation

From now on, we focus on the adaptation process to the Web Model being adapted. RUX-Model connection process takes from the connected Web Model two kinds of information regarding its structure and navigation. Information regarding the presentation model is not considered in RUX-Model because Web 1.0 presentation models are not oriented to Web 2.0 UIs (e.g. no single page application, no partial UI refreshment, etc). The structure and navigation are for allowing Final Interface triggering the Operation Chains defined in the underlying Web application being adapted and for building the RUX-Model Abstract Interface.

The connection process starts selecting the set of Connection Rules, phase 1 in the Figure 1, according to the Web Model that we have chosen. A set of Connection Rules exists for each potential Web Model being considered (e.g. WebML [4], OOHDM [5], UWE [6], OO-H [7] or Hera [12] among others).

3.1 WebML Specific Case

The selection of WebML [4] as the Web Model for this case study is based on previous studies [1]. WebML allows modellers to express conceptual navigation and business logic of the Website. WebML is supported by a CASE tool called WebRatio (http://www.webratio.org) that generates application code. This code is based on JSP templates and XSL style sheets for building the application’s presentation.

Regarding the triggering of Operation Chains, this is solved as in [10], using the “pointing” links, given that WebML links use the typical HTTP GET format: \texttt{pageid.do?PL} where \texttt{pageid} denotes a Web page and \texttt{PL} a list of tag-value pairs.

![Fig 2. WebML Connection process schema](image)

Regarding the building of Abstract Interface, it is important to note that all the concepts of WebML are associated with a graphical notation and a textual XML syntax. WebML XML is composed of several tags (and content). Next we show those ones most relevant for our connection process.

- \texttt{<Structure>}: related to Entity, Attribute and Relationship,
- \texttt{<Navigation>}: related to containers (\texttt{<Siteview>, \texttt{<Area> and <Page>}) and units (\texttt{<ContentUnits>, <DataUnit>, <IndexUnit>, <HierarchicalIndexUnit>, <Multi-dataUnit> and <EntryUnit>} to express and to organize the Web Model.
This information is used all along the RUX-Model designing process as depicted in Figure 2. The Connection Rules filter the information offered by WebRatio, obtaining only the information needed to build Abstract Interface, that is the `<Structure>` and `<Navigation>` elements.

Due to the fact that the WebML navigation model is composed by several siteviews, the first step that takes place in the connection process is to create a basic empty presentation abstract model, in order to insert in it an alternative root view that will contain a simple view for each defined site view. Later on, for each one of these site views, we will process the content placed in each page using the algorithm whose pseudocode is shown in Table 1.

The algorithm works following a basic rule: if the page contains only one WebML Unit it is transformed directly to Abstract Interface Component(s) according to the Connection Rules. If the page contains more than one Unit, a RUX-Model simple view (Figure 2) will be created. This simple view will contain the results of WebML Unit processing. `<Page>` and `<Area>` are treated in the same way.

All the nodes contained in `<Structure>` are used as in the previous Web Model, using their original connectors. All the nodes of the hierarchy defined in `<Navigation>` will be transformed according to the Connection Rules, calling to their identifiers of connectors described in `<Structure>`.

### Table 1. Connection Rules pseudocode.

<table>
<thead>
<tr>
<th>ConnectionRules( AI : AbstractInterface, WML : WebML_Element )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vars</td>
</tr>
<tr>
<td>AIE : AbstractInterfaceElement</td>
</tr>
<tr>
<td>AIC : Connector</td>
</tr>
<tr>
<td>AIV : SimpleView</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>If WML is SITEVIEW or ALTERNATIVE</td>
</tr>
<tr>
<td>AIE Å AI.new_alternative_view( WML.name )</td>
</tr>
<tr>
<td>EndIf</td>
</tr>
<tr>
<td>If WML is PAGE or AREA</td>
</tr>
<tr>
<td>AIE Å AI.new_simple_view( WML.name )</td>
</tr>
<tr>
<td>EndIf</td>
</tr>
<tr>
<td>If WML is CONTENT_UNIT</td>
</tr>
<tr>
<td>AIE Å AI.new_simple_view( WML.name )</td>
</tr>
<tr>
<td>AIE.insert_connector( AIC )</td>
</tr>
<tr>
<td>If WML is DATA_UNIT ...</td>
</tr>
<tr>
<td>AIV Å AI.new_simple_view( WML.name )</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>AIV Å AI.new_replicate_view( WML.name)</td>
</tr>
<tr>
<td>EndIf</td>
</tr>
<tr>
<td>AIE.insert_view( AIV )</td>
</tr>
<tr>
<td>End</td>
</tr>
</tbody>
</table>

### 3.3 Case Study

With the aim of validating our proposal, we show a simple real-life case study. This case study is inspired by the “Pedro-Verhue” Website (http://www.pedro-verhue.be), an on-line catalogue for home interior decoration, based on RIA technologies to provide high interaction and presentation capacities.

Due to the case study extension, we only focus on the Connection Rules that is the main objective of this paper. Notwithstanding, the full engineering process is available on-line through a video tutorial and the Web 2.0 application is deployed at http://www.ruxproject.org.
At the top of Figure 3 the underlying Web Model is depicted (i.e. WebML hypertext model) and at the bottom the RUX-Model Abstract Interface automatically obtained using the connection process (i.e. Connection Rules).

Fig 3. From WebML hypertext Model to RUX-Model Abstract Interface.

Mainly, *Pedro-Verhue* has a first level category (called “Menu” in Figure 3) that *onmouseover* displays a second level category (called “Menu Photos” in Figure 3) in order to show the photograph index. When one of the photographs is selected, the detailed information (called “Show Photo” in Figure 3) is shown to the user. All this process is carried out in a single page, following RIA concepts. Figure 3 focuses on “Menu Photos” to explain how the transformation is carried out. WebML “Menu Photos” page becomes (Figure 3 arrow a) the “Menu Photos” Simple View in the RUX-Model Abstract Interface. “Index Unit 2”, that uses *photo* entity from the WebML structure, becomes (Figure 3 arrow b) the “[Photos]” Simple View with a Replicate View inside. Finally, for each attribute available in the WebML “Index Unit 2” the process creates (Figure 3 arrows c and d) one Media element with a common Connector inside a Simple View.

### 4 Conclusions and Future Work

In this paper we use RUX-Model (Rich User eXperience Model) [12], a Model Driven Method for the systematic adaptation of RIAs UIs over existing HTML-based Web Applications based on Models in order to give them multimedia support, offering more effective, interactive and intuitive user experiences.
Among the transformation phases proposed in RUX-Model, we have focused on the definition of the connection process with the Web Model being adapted. This phase is crucial in the process, due to it being the only part of RUX-Model that depends on the Web Model selected.

We have proposed a case study to demonstrate our approach in practical terms using RUX-Tool. Currently, RUX-Tool is able to take advantage of applications generated using WebML and it is able to auto-generate RIA UIs code for several extended rich client rendering technologies such as AJAX (using DHTML) and OpenLaszlo[8].

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 automatically. Moreover, conceptually, RUX-Model may be used on several existing Web Models such as OOHDM, UWE, OO-H or HERA among others. Discussing this issue is one of our main objectives at the workshop.

Acknowledgments. Partially funded by PDT06A042 and TIN2005-09405-C02-02.

References

Improving the adaptation of web applications to different versions of software with MDA

A. M. Reina Quintero¹, J. Torres Valderrama¹, and M. Toro Bonilla¹

Department of Languages and Computer Systems
E.T.S. Ingeniería Informática.
Avda. Reina Mercedes, s/n.
41007 Seville, Spain
{reinaqu, jtorres, mtoro}@lsi.us.es
http://www.lsi.us.es/~reinaqu

Abstract. The Model-Driven Architecture (MDA) has been proposed as a way of separating the details of an implementation platform from the problem domain. This paper shows that this approach is also good for the adaptation of software to the different versions of the same platform. As an example, Spring Web Flow (SWF), a framework that allows the definition and representation of user interface flows in web applications, has been chosen. After six months of evolution, the web flows defined with SWF 1.0 RC1 were not compatible with SWF 1.0. The paper analyzes the changes introduced by the new release, and it proposes an MDA-based approach to soften the impact of these changes.

1 Introduction

The fast evolution of technology has caused the period of time that companies take for providing new versions of their products be shortened, if they want to be up-to-date. Many times, new releases offer new and improved features, but also cause backward incompatibility. This problem is stressed in open source projects, because new versions are often released out due to their continuous interaction with end users. Therefore, it is crucial to adapt software products that are being developed with these frameworks at a minimum cost.

On the other hand, web applications are becoming more and more complex, and nowadays, they are more than just simple interconnected web pages. Thus, an important piece in its development are Web Objects [4], that is, pieces of compiled code, that provide a service to the rest of the software system. These pieces of code are often supported by open source frameworks, and as a consequence, the evolution of these frameworks has also become an important challenge in web application evolution.

The process of releasing out new versions of a framework or a software product can be considered as a software evolution process. There are several techniques for software evolution that range from formal methods to ad-hoc solutions. But the most promising ones are: reengineering, impact analysis techniques, category-theoretic techniques and automated evolution support. MDA
also seems to be a promising philosophy for dealing with software evolution, not only because it allows the separation of the domain concerns from the technological concerns, but also because design information and dependencies are explicit in models and transformations, respectively.

This paper shows how the MDA philosophy can help us to adapt easily to a new release of a framework the software artifacts produced with an earlier version. Furthermore, it analyzes the evolution process in the MDA. To demonstrate the benefits of this philosophy, the paper will be based on a case study. The case study describes how a web flow defined with Spring Web Flow 1.0-RC1 can be adapted to Spring Web Flow 1.0, a more stable release. Spring Web Flow (SWF)\(^1\) is a component of the Spring Framework’s web stack focused on the definition and execution of user interface (UI) flows within a web application. A user interface flow can be considered as part of the navigation process that a user has to deal with while interacting with a web application. For the sake of having a clear idea of the period of time between the two releases, it should be highlighted that SWF 1.0 RC1 was out in May 2006, while SWF 1.0 was publicly accessible in October 2006. And, although both versions share most of the concepts, there are some technical details that differ and that cause backward incompatibility.

The rest of the paper is structured as follows: In section 2, the problem is introduced by example, that is, the working example is explained and problems are briefly highlighted. Secondly, the approach is explained following three stages: metamodel and tranformation definitions, evolution analysis and change propagation. After that, some related works are enumerated and, at last, the paper is concluded and some future lines of work are pointed out.

2 Problem Statement

Due to the constant evolution of technology, new versions of software products are released out more and more frequently. This cycle of new versions is specially speeded up when dealing with open source products. This is due to user participation: users are constantly sending reports about mistakes. In this context, a frequent operation is software migration, thus the study case is going to be focused on a migration from Spring Webflow 1.0 RC1 to Spring Webflow 1.0, a new, more stable release, which appeared just 6 months after the public appearance of the 1.0 RC1 release.

A flow defines a user dialog that responds to user events to drive the execution of application code to complete a business goal. And, although the elements or constructs needed to define a web flow are the same in both versions, the technical details differ from one release to the other. As a consequence, there is no backward compatibility.

In order to be clear enough, the flow used as study case is simple, but it covers the main issues needed. The initial example has been obtain from [2],

\(^1\) The Spring Web Flow Home Page: http://opensource.atlassian.com/confluence/spring/display/WEBFLOW/Home
and can be seen as part of the navigation path from a simple e-shop web site. The flow simulates part of the dialog that takes place when a user wants to buy certain product. The navigation steps that a user has to pass through to buy some thing are: Firstly, select the Start link. Secondly, enter his personal data. Thirdly, select the properties of the product. And, finally, after pushing the Buy button, he will obtain a message reporting about the shopping.

![State chart corresponding to the web flow specified in SWF 1.0 RC1](image)

**Fig. 1.** State chart corresponding to the web flow specified in SWF 1.0 RC1

The implementation of this dialog using SWF requires the definition of a web flow. This web flow can be specified in two different ways, by means of an XML-file or using a Java-code. It is a good technique to draw a state chart in order to understand the web flow better. The flow consists of six states (Fig. 1(a)): three ActionState’s, two ViewState’s and one DecisionState. Initially, the flow starts its execution by an ActionState which is in charge of setting up the form. Then, the form is displayed through the personalDetailsView state. After that, the flow enters into another ActionState, which will bind and validate the data introduced by the user. If there are any problems with data, the flow will go back again to the personalDetailsView state; otherwise, it will enter into the orderDetailsView state, and the process will be repeated. Finally, if all data are right, the flow will enter into the testQuantity state, a DecisionState, that can route the flow depending on the value of the attribute cancelled. However, if the flow evolves in order to be SWF 1.0 compliant, we have the state chart
shown in Figure 1(b). The number of states has been reduced from six to three. That is, all the ActionState’s have disappeared.

3 The approach

3.1 Metamodel and transformation definition

The first step in the approach is to obtain a metamodel which expresses the concerns that are implicit in the framework and their relationships. In our case study, a metamodel of Spring Web Flow is needed. In this case, two different metamodels should be defined, one for the SWF 1.0 RC1, and the other one for SWF 1.0. The great advantage here is that, at this point, both metamodels should not differ too much. The models conforming to these metamodels are also needed. It is likely that we have models conforming to SWF 1.0 RC1, but if the model-driven process has been not followed, some reengineering techniques could be applied to obtain them. Finally, a set of transformations for obtaining the code should be given, whether model-to-text or model-to-model transformations. Furthermore, with all these artifacts, an analysis should be done in order to determine the kind of evolution that has been entered into the new release of the framework.

3.2 Analysing the evolution

In order to face up to the adaptation process, the artifacts that are subject to change should be identified, and also, which of these changes should be classified as evolution. In MDA there are three ways of evolution: model evolution, transformation evolution and metamodel evolution. In model evolution, changes to source models must be mapped into changes to target models through the transformation rules. In transformation evolution, changes to the transformation definition must be mapped into changes to target models. Finally, in metamodel evolution, changes to a metamodel must be mapped to changes to described models, plus to transformation definitions.

This section will analyze the different changes introduced in the Spring Web Flow framework, and it will classify them according to the ways of evolution in MDA. In our study case, the main changes introduced to Spring Web Flow are:

1. FROM DTD’S TO XMLSCHEMAS. Although structurally, this change is important, conceptually is very simple. The only thing to do is to define a new set of model-to-text transformations. And, if the model-to-text transformer is based on templates, we only have to modify the template. This is a kind of model evolution.
2. CHANGING THE ROOT AND THE INITIAL STATE. This change is also simple. While in SWF 1.0 RC 1, the root of the XML webflow was \texttt{<webflow>} in SWF 1.0, the root is \texttt{<flow>}. Moreover, the way of specifying the start state has also been modified: in SWF 1.0 RC, the initial state was specified as an
attribute of the root element <webflow id="orderFlow" start-state="-setupForm">; however, in SWF 1.0, it is defined as an XML element <start-state idref="personalDetailsView"/>. These modifications only imply the template redefinition. This is a kind of model evolution.

3. New renderAction property in the ViewState This change is due to the introduction of a new property renderAction belonging to ViewState. But this new property implies a bit of conceptual change, because we should change the design of the flow in order to take advantage of the advanced features of the framework. Thus in SWF 1.0 RC1 an initial ActionState was needed in order to setup initially a form. If we look at the Figure 1, we will see that the start state is an ActionState named setupForm. However, in SWF 1.0 this can be represented by a property <render-actions> linked to the ViewState that is in charge of rendering the form. As a result of the new design, in the Figure 1(b), the ActionState has completely disappeared. This is a kind of metamodel evolution.

4. Actions in Transitions This change is not really due to the new version of SWF, but due to the inexperience of the authors with SWF when working with SWF RC1. Thus, two states (one ViewState and one ActionState) were defined in order to specify, on the one hand, a web form, and on the other hand, the binding and validation of data introduced by the user in that form. There, the ViewState named personalDetailsView is followed by the ActionState named bindAndValidatePersonalDetails, which is in charge of the binding and validation of user data. In that flow there is also another pair of states that follow the same pattern orderDetailsView and bindAndValidateOrderDetailsView. However, these two states can be replaced for just one ViewState, and the validation and binding can be triggered by the transition, which implies the disappearance of the ActionState. Thus, if we compare the Figures 1(a) and 1(a), we will see that the number of states has been reduced, and now, the bindAndValidatePersonalDetails and bindAndValidateOrderDetails have been missed. This is a kind of transformation evolution.

3.3 Change propagation

In order to migrate our application to be compliant to the new version of the framework, different actions should be undertaken. And the concrete action will depend on the kind of evolution. The easiest evolution to face up to is the model evolution. In this case, the only thing to do is reformulate the set of model to text transformations defined for generating the XML-Webflow files.

The metamodel evolution implies not only the modification of the Spring Webflow metamodel, but also the definition of new model to text transformations. In order to migrate the old Spring Webflow models into the new ones, two different strategies can be followed: one based on horizontal transformations, and the other one based on vertical transformations. Horizontal transformations [1] change the modular structure of an application at the same level of abstraction. If we think in model transformations, source models and target models should
be expressed at the same abstraction level. On the other hand, vertical transformations [1] involve the transformation of a high abstraction level model into a lower level one.

In the strategy based on horizontal transformations, besides the original model-to-text transformations, a set of horizontal model-to-model transformations has to be defined. These transformations along with the original and evolved metamodels, and the original model will be the input of a model transformation engine, which will produce the evolved model as output. Applying the model to text transformations, the evolved Spring Webflow will be obtained.

The second strategy is more aligned to MDA and it consists on the definition of a new metamodel which captures only the relevant concerns, that is, it should ignore those elements that are platform dependent. This second approach is a bit more expensive than the first one, in the sense that new artifacts are needed, and they are conceptually more complex. But it also has some advantages. Firstly, as the different versions deal with the same concepts, it is likely that the PIM metamodel will not change very often, and the important modifications should be at the PSM and transformation levels. Secondly, with the second approach we have to define a new metamodel for the new version of the product, but many times, this new metamodel is very similar to the one defined for the previous version, so this task does not suppose too much work. And, finally, if we define a PIM metamodel, we can deal with the same concepts but in other platforms. Finally the model evolution can be solved defining a set of horizontal transformations to reformulate the old models. In this case, the metamodel remains the same, and also the model to text transformations.

4 Related Work

The model-driven software evolution is a new area of interest, thus in the 11th European Conference on Software Maintenance and Reengineering a workshop on model-driven software evolution has been held. In [7] a survey of the problems raised by the evolution of model-based software systems is made and it is stated that Model-Driven Engineering requires multiple dimensions of evolution. Our approach deals with three of these dimensions. On the other hand, in [6] the drawbacks of model driven software evolution are analyzed, and as a conclusion the authors state that a dual approach is needed, that is, to use requirements evolution to generate the model specification and the test specification to validate the system. Our approach follows a top-down approach, but, at this point we are not interested in validation or verification.

In [3] incompatibilities between models and metamodels caused by metamodel revisions are faced up. The proposed approach is based on the synchronization of models with evolving metamodels, but this approach only deals with one dimension of evolution, the metamodel evolution. [5] proposes a framework where software artifacts that can be represented as MOF-compliant models can be synchronized using model transformations, but they are focused on traceability of changes of software models. Finally, in [8], a systematic, MDA-based
process for engineering and maintaining middleware solutions is outline. In our approach, SWF can be considered as part of a corporate middleware.

5 Conclusions and Further Work

This paper has pointed out how we can take benefit of the MDA philosophy in order to have a better adaptation to the different versions of the same software product. To do so, a case study based on the Spring Web Flow framework has been introduced. An analysis of the changes introduced in the new version of the framework has been made. Furthermore, two different approaches based on model transformations have been considered to face to metamodel evolution. In this case, the following artifacts are needed: one SWF 1.0 RC1 metamodel, one SWF 1.0 metamodel, one web flow metamodel (this one, at the PIM level), two sets of model to text transformations (one for obtaining the XML file conforming to SWF 1.0 RC1, and another one for obtaining the XML conforming to SWF 1.0); and, finally, a set of model to model transformations, which will transform the web flow model (at the PIM level) into a model for SWF 1.0 RC1 and SWF 1.0, respectively.

One of the future works is the implementation, via web, of a metamodel repository, in such a way that we can count with the metamodels of the different versions of the frameworks. Thus, a set of metamodels will be publicly available in order to improve the adaptation to different releases of a framework.

References

International Conference on Web Engineering 2007

1st Workshop on Aligning Web Systems and Organization Requirements

16th July 2007, Como, Italy

Organisers

David Lowe, Didar Zowghi
University of Technology, Sydney

Workshop Program Committee

Dr. Sotiris Christodoulou, University Of Patras, Greece
A/Prof. Jacob Cybulski, Deakin University, Australia
Prof. Daniel Berry, University of Waterloo, Canada
Prof. Jim Whitehead, University of California, Santa Cruz, USA
Dr. Lorna Uden, Staffordshire University, UK
Prof. Al Davis, University of Colorado, USA
Prof. Armin Eberlein, University of Calgary, Canada
Dr. Emilia Mendes, University of Auckland, NZ
Dr. Scott Overmyer, Baker College, USA
Prof. Roel Wieringa, University of Twente, The Netherlands
Dr. Davide Bolchini, University of Lugano, Switzerland
Prof. Ray Welland, University of Glasgow, Scotland
Table of Contents:

Web Based Requirements Management Approach for Organizational Situation Awareness ................................ 112

Vikram Sorathia, Anutosh Maitra

This paper introduces an approach towards building situation awareness systems for organizations that are targeted to function in dynamic environment. Considering process engineering related issues, a unified process for situation awareness is proposed. With introduction to novel artifacts that are found essential for achieving proper situation awareness, unique requirements are identified for a metaCASE tool that will provide access to these artifacts in collaborative environment. The proposed approach suggests capturing organizational requirements in formal Ontology and provides mechanism for deriving traceability among various artifacts in web environment.

Conceptual Modelling of Service-Oriented Systems .................. 122

Mario A. Bochicchio, Vincenzo D'Andrea, Natallia Kokash, and Federica Longo

The design of service-oriented systems currently is one of the most important issues in the software engineering. In this paper, a conceptual framework for designing Web service-based systems is presented. This approach is characterized by client-centered analysis and presence of business-process modelling to identify functionalities and collaboration patterns of involved Web services. Service discovery and selection are parts of the design process. A case study is provided to explain the principle steps of the proposed framework.

Aligning Web System and Organisational Models ..................... 132

Andrew Bucknell, David Lowe and Didar Zowghi

In this paper we describe an approach to facilitating the alignment of web system and organisational requirements by using graphical models of the processes that are being supported by a web-based system. This approach is supported by the AW eSOM e modelling architecture. This architecture allows us to investigate the effectiveness of different notations for modelling systems. The architecture is being implemented as the AW eSOM e modelling tool, which will be used to investigate our approach to alignment in industry-based case studies.
Foreword

Whilst there has been considerable attention applied to the design and implementation of Web systems – and this has justifiably been a focus of most research in the Web Engineering field – there is a growing recognition of, and interest in, how we most effectively determine the scope of these systems and integrate them efficiently within their organisational context. Indeed it has been argued that lack of consideration of this area is a major cause of creating Web applications which are technically successfully, but operationally a failure. This workshop aims to bring together researchers and practitioners who are working in the early stages of the Web development process – i.e. those activities where Web systems are scoped, specified, and where impacts on, and effective integration with, existing organisational processes are considered – and to explore how improvements in techniques, models, and tools can lead to better integration of Web systems and organisational processes.

The workshop is designed to be relevant to researchers in the Web Engineering field, but to also draw in researchers from related areas (requirements engineering, system specification, business analysis, business/IT alignment, etc.), who would not normally have considered attending the ICWE conference, but who are undertaking research which is highly relevant. In order to make the workshop as relevant and practical as possible we have included significant opportunities during the workshop for critical discussion and analysis of the issues raised in the papers. We have also included a thought-provoking keynote by Professor Roel Wieringa which will set the scene for the workshop. Professor Wieringa is internationally recognized in the area of requirements engineering. It is our hope that this workshop will increase awareness of and interests in these important issues within the Web Engineering community and serve as a starting point for future collaboration among researchers from different disciplines.

Professor David Lowe
A/Professor Didar Zowghi

30th May 2007
This paper introduces an approach towards building situation awareness systems for organizations that are targeted to function in dynamic environment. Considering process engineering related issues, a unified process for situation awareness is proposed. With introduction to novel artifacts that are found essential for achieving proper situation awareness, unique requirements are identified for a meta CASE tool that will provide access to these artifacts in collaborative environment. The proposed approach suggests capturing organizational requirements in formal Ontology and provides mechanism for deriving traceability among various artifacts in web environment.

1 Introduction


With expanding business enterprises, consortiums, trade groups, governmental alliances, and international organizations, building of information system has become quite challenging even with the modeling techniques mentioned above. For example global policies for reducing disaster risks require information collection, processing, sharing and dissemination at various geographical scales[5]. Wide geographical coverage, multiple disciplines and dynamic scenario make it difficult to capture all the aspects of the Universe of Discourse (UoD) a priori.

Considering a kind of a scenario given in Figure 1 for an information system, the universe of discourse can be represented from multiple domain point-of-
views for the given organization. The interpretation of situation is therefore based on rules defined in respective domains[6]. To realize the information flow as indicated, various specialized tasks are to be carried out to obtain appropriate domain representations. Once representation is available, the information processing and determination of required set of actions can be done, result of this activity should be sent to real-world actors who can alter or maintain the situation in their desired status. Hence actors in real-world, as well as the actors playing different roles in: creating representations, domain world view, inferring required actions, and communicating to appropriate actors in the given UoD; all should be equipped with appropriate Situation Awareness.

The concept of Situation Awareness was initially introduced[7] in reference to flight automation domain yet with required modifications, it can be adopted in building reactive systems for any domain in general. Information systems in dynamic enterprises provide a challenging case for designing a reactive system. All team members responsible for information system must be able to react to the changing needs of the organization. The changing need can be detected and communicated with proper situation awareness framework. Hence in present context, Situation Awareness is a state achieved in which a role is provided with information at specific space, time and conceptual granularity; determined in the prevailing context and the underlying information communication configuration. It not only provides world-view of domains relevant to the role, but also provides actions required.

1.1 Orthogonal Concerns

The following representation depicts how a specific instance in enterprise information system can be traced back to real world processes in a UoD depicted in
This kind of mapping reveals that at each of the defined stage, various roles carryout certain activity within or outside the system to produce specific artifacts. An important observation is that though each of these activities are having specific sequence and input output dependency among them, the actors performing the activity may have completely orthogonal interests. For example service developers, service providers, data providers, configurators and users: all are associated to a specific service, but a service developer may not show any concern about how data will be provided, or how service will be deployed.

2 SA Unified Process

During the requirement gathering stage of an enterprise information system many non-functional requirements that are essential for collaboration in dynamic environment can be missed. Efforts toward team integration later becomes difficult task. The basic principle of the unified process is that, organizations will define their commitment toward methods instead, and then the developers will develop and test according to the given reference model. Thus rapid adoption of missing components in the system could me made possible.

2.1 Life cycle

Different phases in a life cycle of an SA Process are depicted in Figure 2. The phases are defined as follow:

– **Policy**: The life-cycle starts with the policy of the organization to collaborate for situational awareness needs.
– **Analysis**: The requirements phase identifies how situational needs can be fulfilled with the available infrastructure.
– **Design**: Designing allows the development of required extensions of core SA services and components to suite the local needs. This phase includes development of DoDAF architectural products that guides the realization of identified needs.
– **Mapping**: Mapping is done at the semantics level. Once the application is defined, the organizational rules and other Ontologies can be mapped appropriately.
– **Configuration**: Configuration is the process in which actual instances are configured.
– **Management**: Once the configuration is up, the intermediate task for data management, fine tuning and resource management for load balancing are required.
Review: The behavior of the configuration is evaluated according to the needs. A number of traceability matrices are studied to determine the capability of the system. Coverage analysis from Traceability matrix provides basic information required for comprehensive Gap Analysis.

Archive: The archival phase purges existing instances, stores the traces, and other logs for future references.

2.2 Architectural Products

The foundation of the proposed SA unified process is based on DoDAF, TOGAF, RUP and some other existing proposals. Many of the architectural products in the SA Unified process are same as identified in these basic products. For example, it imbibes all the architectural products of DoDAF[8]. SA Process is planned to be service centric and service related products are same as defined in SOMA[9]. These Architectural Products not sufficiently equipped to support the required situation awareness to various actors during different phases in SA Process life-cycle. Hence following web based architectural products categorized as Situation Awareness Views (SAV) targeted at actor situation awareness are introduced, details of which is discussed in [10].

- Organizational Knowledge base URL View (SAV-1)
- SA Role Product Matrix (SAV-2)
- SA Information Need-Component Matrix (SAV-3)
- SA Information Need-Service Matrix (SAV-4)
- SA Information Need-Data Matrix (SAV-5)
- SA Information Need-ETL Matrix (SAV-6)
3 Requirements for a Meta CASE tool

While many popular CASE tools allow provisions for composing the methods, certain additional features must be incorporated to support the required situation awareness to the users.

3.1 Supporting Separation of Concern

The actors associated in the process may have completely orthogonal concerns and hence, tooling must provide support for customized experience of each actor playing one or more roles in a given instance. This need becomes more relevant when actors just provide their skill-set and the CASE tool thereafter should be able to infer what possible roles they can play.

3.2 Event Driven

Targeted systems are reactive in nature; and the CASE tool must support event based triggers. The RUP may specify if a particular activity is event triggered or not, but the mechanism of detecting the event is not known. The events are also considered to be delivered not to the explicit subscriptions but the general roles. As the RUP further allows specification of skills, this information can be used for identification of the proper recipient of the event notification. Hence the CASE tool must be able to collect profiles of the potential members, and the event detection mechanism must be able to identify appropriate recipient and be able to deliver message to them over collaborative environment.

3.3 Dynamism in Organization

In situations where organizations are responding to emergency situations or crisis, the decision for setting up an Emergency Operations Center (EOC), and decisions related to allocation of resources keep on changing as the event unfolds. As exact boundary of emergency is discovered, organizations may have
to come together at national or international level resulting in change often in drastic manners. The resources in terms of skilled manpower and ICT infrastructure may change with volunteers and donated resources, that needs to be incorporated instantaneously. The CASE tool must support the required level of dynamism experienced by the organizations during the events.

3.4 Estimation of Efforts

For any organization, the volume of work required to achieve it’s goal should be determined. There can be some systems already in place, which can be integrated with the planned information system. The knowledge representation needs to be incorporated also in the estimation, that allows the organization to identify the items they are supposed to make provision for.

3.5 Knowledge Representation

Consistent flow of information in situation awareness system is only possible if consistent representation of knowledge is adhered to throughout the process life-cycle. It also allows reusability of the domain knowledge. The spatial, temporal and semantic reasoning is very critical for the success of the CASE tool, so the knowledge representation that can allow such reasoning is a basic necessity. While knowledge representation and reasoning allows support for spatial and temporal relationship, the architectural products should be able to use this aspect. For example various grid nodes which are part of a Virtual Organization (VO) can be scattered across a larger area. Hence the decisions regarding data/artifact repository regionalization and other such provision requires spatial nature to be considered. The meta data is also having spatial elements in the schema, hence architectural product should be able to render instances and allow query based on spatial/temporal attributes.

3.6 From Monolithic to VO

The collaborating teams should be considered as members of a VO. Like successful applications as bug tracking is handled in collaborative mode in web environment, the SA Unified Process must also be handled in similar manner.

3.7 MoM Patterns

Members in the VO have different connectivity scenario. So appropriate Message Oriented Middleware (MOM) patterns must be supported by the CASE tool to achieve enterprise integration through appropriate messaging.
3.8 Semantics based Traceability

Traceability among architectural products provide basis for tracking the coverage of the effort. The software teams involved in SA is provided the tasks during the entire life-cycle, and hence each and every part is interpreted as required due to some precursor. Thus traceability can actually be based on semantics and should be able to cover the entire set of the tasks.

3.9 Task Allotment

Task allotment to volunteers or the team members can be done based on evaluating the skills. Once the volunteers define that they will be continuing with the task, system should be able to take notice of the same. The status of ongoing task should be identifiable at any intermediate interval.

3.10 Visualization

Complex level architectural approaches demand high degree of technical expertise for the user. How users will take up the task depends on how effectively it is provided with proper visualization. Monitoring of allotted work, overview of the process status, search of architectural components, the hierarchical view coverage and errors should be rendered to improve the situation awareness.

3.11 Standardization

The standardization related concepts in the ontology suggest their applicability. Each artifact or work product should be traced to appropriate standards. Standardization traceability matrix not only supports the developers to consider standards, it also provides the standard specification that must be considered while developing given application.

3.12 Artifact Impact and Reusability

The SA life-cycle demands the knowledge of when and how reusable artifacts can be published, discovered and utilized. Figure 3 describes all SA Process artifacts based on the frequency of update and spatial relevance. There can be some artifacts that are relevant for some time only for very specific region and can not be reused. The data collected, the execution environment, system level tasks, organizational decisions and reviews fall in to this category. Some artifacts are local but valid for longer period of time and do not require frequent updates. For example, SA Configuration, organizational knowledge, mapping, transformation and organizational policies. SA review and SA management can be considered to be globally relevant but short term and needs to be verified by multiple implementations. The fourth quadrant contains the set of artifacts that can be reused. For example SA Process, Services, Components Standardizations and domain knowledge, can be created or incorporated by any SA Configuration and can be reused globally for long period of time.
3.13 Web Based Access

In order to meet all the requirements discussed above, the proposed CASE tool must be web-based for the following reasons. As a dynamic set of actors form a VO to identify, create, monitor, utilize and share various process artifacts, web based access to artifact repository fulfills this requirement. The requirement of creating assertions in reference to Ontology, inferencing the events, determining the traceability etc. can be achieved by accessing ontology server. Visualization of process artifacts, task assignment, allocation, monitoring and other required features can only be achieved in web environment. Hence various core services of the CASE tool are accessed over http in collaborative manner.

4 SACore Configuration

In any given situation, the CASE tool must be able to infer the information need for a given instance.

– The collection of required information, the provision of messaging and managing the reports by the user must also be supported.
– From the collected data, event detection tasks should be identified and for the identified events the set of action needs to be defined.
– For the identified actions, status must be reported back.
– All these should immediately reflect updates as organizational decision changes. Hence determination of task to be done at particular instance is a critical requirement.
– Artifacts and work-products can be considered as jobs submitted to a processing engine. The status thereof becomes important in tracking the progress.
– System level and middleware level job monitoring provides reliable mechanism of monitoring job status, but the CASE tool itself must also provide support to monitor and report the status of the job.
Various roles are responsible for artifacts or work products that can be reused in other configurations. This identifies a requirement for a mechanism by which a role can publish and discover newly created work products to improve reusability.

Figure 4 depicts the Situation Awareness Process artifacts that are introduced in Section 2.2. The artifacts are part of a message bus that connects all the users of the SA Core infrastructure. SA Core is Service-Component infrastructure realized in the form of various Eclipse plug-ins and is configured and deployed in grid environment. The SA Core consists of four major parts for MoM, KM, VO and Data Management components—details of which is beyond the scope of this paper. The proposed architecture of SA Core has a programming model and a series of artifacts configured in a way to support the required functionality of a meta CASE tool discussed above.
5 Conclusion

This paper presented some ideas for building information systems that are targeted at Situation Awareness Systems in dynamically changing environments. With review of the present status in Unified Processes, Enterprise Architecture Frameworks and other aspects in building information systems, a Unified SA process is proposed that extends the current capability of available approaches. Many novel architectural products that are required for the purpose have been presented. The requirements of meta CASE tool more suitable of providing appropriate access to users have been identified. Finally a configuration with SACore that can be considered as meta CASE tool is discussed. The approach also demonstrated separation of concern amongst various types of users in collaborative virtual environments.

References

Conceptual Modelling of Service-Oriented Systems

Mario A. Bochicchio\textsuperscript{1}, Vincenzo D’Andrea\textsuperscript{2}, Natallia Kokash\textsuperscript{2}, and Federica Longo\textsuperscript{1}

\textsuperscript{1} SetLab, University of Salento, Italy \{mario.bochicchio, federica.longo\}@unile.it
\textsuperscript{2} DIT - University of Trento, Italy \{natallia.kokash, vincenzo.dandrea\}@dit.unitn.it

Abstract. The design of service-oriented systems currently is one of the most important issues in the software engineering. In this paper, a conceptual framework for designing Web service-based systems is presented. This approach is characterized by client-centered analysis and presence of business-process modelling to identify functionalities and collaboration patterns of involved Web services. Service discovery and selection are parts of the design process. A case study is provided to explain the principle steps of the proposed framework.

1 Introduction

Web services are self-contained software units that can be published, located and invoked via the Internet. They are seen as the future of the Web simplifying business integration and achieving a separation of concerns. Many organizations are interested in wide acceptance of Service-Oriented Architectures (SOAs) and Web service standards, leveraging both from a perspective to use the Web as a market for their own services and ability to consume already existing ones. In practice, this technology brings multiple technical and conceptual problems. One of them concerns the development of Web services and their further reuse in more complex systems. It is assumed that designers of such systems either are aware about existing Web services and can model systems able to collaborate with them or describe abstract services they would like to use relying then on automated (even runtime) service discovery to substitute these abstract descriptions with real Web services. None of these scenarios is really acceptable. In the former case, all information about a required Web service should be known in advance (before starting actual system design) to allow for its location in a UDDI registry or a like. In the latter one, it is unlikely that exactly such a service will be found and, therefore, a bulk of problems with service or system adaptation to achieve interoperability appear. To resolve this issue, a design methodology that follows the principle “meet-in-the-middle” is required.

Another question is what functionalities should be implemented as Web services to be both self-contained and allow for their reuse in different contexts. Who are their intended clients? Such functionalities can be singled out through analysis of existing business processes to accomplish logically independent tasks.
that are not highly specific for a given process. It means that while modelling a system designers can stand out potentially reusable parts as Web services with additional efforts to make them customizable and acceptable by all members of an intended audience.

The objective of this paper is to find a conceptual design methodology that addresses the above problems. Conceptual modelling of business-level collaboration protocols is essential for attaining the system composed from a set of communicating services, easily replaceable with other services and potentially reusable in different application contexts. The design process in this case requires analysis of all stakeholders of the system and its goals at the abstract level. We adopt the xBPEM (Business Process Enhanced Model Extended with Systemic Variables) [1] framework and extend it for service-capable modelling.

The rest of the paper is organized as follows. In section 2, a necessary background on Web services and service-oriented design is given. Section 3 presents the proposed methodology for modelling service-based business processes. In Section 4, a case study is given. Section 5 summarizes the contributions of this paper and outlines future work.

2 Background

In this section we analyze existing service-oriented modelling techniques. Web Service Description Language (WSDL) describes a Web service interface by means of operation signatures. Different languages have been proposed to supplement static service interface descriptions with dynamic information about service functionalities. Business Process Execution Language for Web Services (BPEL4WS) includes means for specifying abstract and executable processes. Global behavior of all parties involved in a collaboration can be specified with Web Service Choreography Definition Language (WS-CDL). Use of WSDL, BPEL4WS and WS-CDL for modelling service-based business processes brings two problems: (1) the level of abstraction is too low for convenient modelling, and (2) the languages are lacking a standardized graphical representation which would ease the design process.

Basic design principles of service-based applications have been described by Papazoglou and Yang [2]. A weak point of this work is that it does not distinguish logical business processes from their implementation. Its first step is to “identify, group and describe activities that together implement a business process,” which is a sort of bottom-up approach. The presented framework is fully service-oriented, i.e., all activities are modelled as Web service invocations, while in real-world systems other functionalities are needed. Model Driven Architecture (MDA) aims at simplifying business and technology change by separating business and application logics from underlying platform technology. It clearly separates business processes, designed based on pure business concepts, from technical processes, or software realizations of abstract business processes. Quartel et al. [3] propose Interaction Systems Design Language (ISDL) for graphical
modelling of service-oriented systems, considering specific concepts such as internal and external activities.

We suppose that a step from proprietary notations towards Unified Modelling Language (UML), standard, well-known and close to software system design language, should be done. Such approaches take advantage of model-based analysis of semantical interoperability between different components of a system. UML-based service discovery tools (e.g., [4]) can simplify discovery and adaptation of external Web services minimizing risk of logical flaws through analysis of their structure and internal behavior. Another argument towards UML is that Web services represent a new dimension in development of Web Information Systems (WIS). Such systems now can be constructed by means of transparent integration of services available on the Web and graphical Web interfaces. A language for modelling Web services should be compatible with traditional software design techniques that normally employ UML.


There is a need for describing the procedure on how to derive “good” service abstractions from high-level business requirements and business process models (e.g., identify candidate services within a given UML analysis model) [14]. Too coarse grained services tend to have a low reuse potential, while too fine-grained services may not be loosely coupled and require a big amount of message exchange and coordination efforts. An idea of using a meet-in-the-middle modelling approach as opposed to a top-down or bottom-up solutions seems to be effective in this context [15]. A service-based systems must be designed using a combination of business-driven service identification coupled with service identification through leveraging legacy assets and systems.

Design of service-oriented systems that implement abstract cross-organizational business processes is not an easy task. Such systems should lead to creation of reusable context-independent services and be oriented on adaptation of existing
legacy components and provision of appropriate interfaces and QoS for all kinds of end users. The goal of the current work is to come with a set of design steps that can help to make good SOA-related decisions. We examine the possibility to adopt xBPEM methodology [1] for designing service-oriented systems.

3 Methodology

In this section we propose our framework for conceptual modelling of service-based business processes.

UML is a standard general purpose modelling language, but it is mainly system oriented. BPEM [16] is an extension of UML for business process modelling. BPEM qualitatively represents business goals in natural or semiformal language but does not consider impact of Web technology on them. xBPEM approach further elaborates BPEM by introducing a consistent business strategy design methodology. This methodology is used to control business goals through business process Key Performance Indicators (KPIs), which enable measurement of both internal process performance and QoS from customer viewpoints.

Figure 1 draws the main conceptual steps of the xBPEM and its extension to enable seamless involvement of Web services into the process. In a nutshell, the proposed framework consists from the following steps:

1. Business Process Design. The overall process begins with the business process modelling phase according to the xBPEM methodology. Thus, stakeholders are defined with their high-level goals and basic KPIs, which measure the global performance from the viewpoints of each stakeholder.
2. **Service Identification.** This step is focused on separating loosely coupled functional parts of the collaborative business process into standalone service components. Considering that business processes evolve in time in order to adapt to business strategy which changes following market rules (i.e. new products, new services, new business models), we may observe that candidate services can be identified as:
   - repeated blocks of activities inside a business process;
   - similar blocks of activities among various business processes or different applications;
   - time-invariant blocks of activities in a time-variant business process.
An accurate analysis of the activity diagrams produced at the first step is essential to identify these blocks of activities and to make them eligible to become Web services. The analysis process can be partially automated by searching for similar graphical patterns in the activity diagrams representing abstract business processes.

3. **Choreography Design.** Desired communication patterns among identified abstract services are defined using UML sequence or collaboration diagrams based on the interaction patterns among stakeholders, Web services and the system which are shown into the xBPEM’s swimlanes diagrams.

4. **Requirements Elicitation.** Requirements for discovery of existing software components and Web services (inputs/outputs, pre-conditions/effects, behavior patters and desired quality levels) are defined based on the identified abstract services.

5. **Components Discovery and Selection.** Software components that can be used to compose the system are of two essential types:
   - **Modifiable**, i.e. available as documented source-code (e.g., open source projects) or coming from organizations available to customize it (e.g., software houses);
   - **Unmodifiable**, i.e. coming from providers not particularly interested in customization or not customizable at all (e.g., Web services).
In particular, this step covers discovery of Web services (interfaces, behavioral specifications and coordination patterns specified at the previous stages). Then, the QoS information about Web services and their providers must be collected (using data published by service providers, authorized agencies or other service clients), in order to select the best candidate Web services.

6. **Risk Analysis.** This step is needed to assess risk related to use of external Web services (loss of service, loss of data, security/privacy concerns, etc.). On the basis of this assessment, the existing risks can be mitigated through selection of more appropriate services, use of alternative services for critical tasks, system re-design, data replication, and so on.

7. **Design Refinement and Component Adaptation.** At this step the prepared models can be refined to allow for the seamless integration of the external Web services. Found Web services and existing legacy systems are tested and analyzed in order to decide how to introduce them to the system. It may happen that adaptors or wrappers are needed. Providers of chosen services...
become stakeholders of the system. If no services with required functionalities are found, the organization should think about implementing them, thus, increasing code reusability both in its own future projects and in the projects of third parties.

8. **QoS Negotiation and Service Contraction.** If quality parameters of discovered Web services do not correspond to identified KPIs, they can be negotiated with service providers. At this step identified KPIs should be mapped into direct requirements for QoS further resulting in SLAs.

### 4 Case Study

In this section we consider a case study extracted from a news publishing domain.

The case-study scenario is as follows: A journalist writes an article via mobile phone or via PC and sends it to the editorial office. The system must identify the author and notify the editor about a news event. In case of successful authorization, the news item is stored in a system archive. The editorial staff can see the registered articles, send comments to the author, ask for news revision, select one or more publication channels, and publish the news. The author is notified that his/her article has been published. To access the news, a reader selects a preferred news channel.

According to the methodology presented in Section 4 this scenario is modelled with the help of goal diagrams and business process diagrams (Activity 1). For each stakeholder main goals are shown in Fig. 2. Related goals are connected with dotted lines. Then, KPIs are defined from perspective of each user type. Table 1 describes examples of KPIs for news editor and news writer.

Figure 4 shows the news distribution process as the flow of three sub-processes: news writing, news publishing and news reading. Activity diagrams and swimlanes, which describe what each of the stakeholders does within the process,
Table 1. KPIs from the news writer’s and news editor’s perspectives

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Indicator</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>News Writer</td>
<td>Level of automatization</td>
<td>Number of steps supported by a computer system divided by a total number of steps</td>
</tr>
<tr>
<td></td>
<td>Simplicity of the interface</td>
<td>Number of operations to complete the task</td>
</tr>
<tr>
<td></td>
<td>Time to accomplish a task</td>
<td>Measured in minutes</td>
</tr>
<tr>
<td></td>
<td>Information sharing cost</td>
<td>Time from a data entry to a delivery output</td>
</tr>
<tr>
<td>News Editor</td>
<td>Task error prevention</td>
<td>Number of incorrectly published news (i.e., number of news in a wrong section)</td>
</tr>
<tr>
<td></td>
<td>Misprint prevention</td>
<td>Number of syntactic errors in published material</td>
</tr>
<tr>
<td></td>
<td>Integration</td>
<td>Number of data transmission errors</td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td>Number of security problems per annum</td>
</tr>
</tbody>
</table>

are shown. Blocks of activities emphasized with color can be provided by Web services. Based on this diagram the following Web services are identified (Activity 2):

– A service (services) responsible for user authentication via SMS, MMS or the Web. Authentication is a functionality with a constant graphical pattern used in three sub-processes, namely news writing, news publishing and news reading, and potentially can be required in other places of the system or other business processes of this organization.

– A service (services) responsible for interaction between a pair of stakeholders or a pair system-stakeholder via SMS, MMS or the Web. Interaction activity has a constant pattern used in different points of the business process;

– A service (services) responsible for syntax checking. Syntax checking is a time-invariant functionality.

– A service (services) responsible for a payment procedure. Payment is a time-invariant functionality that can be used in other business processes (e.g., for author rewarding in the context of the current application).

At the choreography design step (Activity 3), interaction patterns among stakeholders, Web services and the system are modelled with collaboration diagrams. In Fig. 4 news writer and news editor identification processes are shown. This step helps to elicit requirements (e.g., operation signatures, capabilities, pre-conditions and effects) for Web services to be discovered or created (Activity 4). So, in our case study, service operations for user identification via SMS/MMS and the Web must deal with a login and a password or sender’s phone numbers as input. The output message should contain acknowledgement or rejection, and a reason in the latter case. Moreover, a service must support assignment of roles to users. A message must be passed through system controller to enable the permitted functions for this user (e.g., submit news or edit saved news).

The next step (Activity 5) is to find Web services with approximate behavior defined at the previous steps. For example, a set of Web services capable to perform on-line payments and send messages via SMS and email have been discovered in the xMethods service registry\(^3\)(see Table 2).

\(^3\) XMethods web service registry - http://xMethods.com/
Risk analysis both at technical and business levels is required to establish potential threats affecting the system and damaging stakeholders (through monetary loss, breach of reputation, etc.) at the early stages of the design process (Activity 6). It can be noticed that in our scenario some mechanism is required to collect feedback from a news reader to reveal inappropriate behavior of the payment Web service, unavailability of the syntax checker can be tolerated while a failure of the interaction services may cause severe consequences for the system and, therefore, must be prevented through selection of strategic business partners or service replication.

Thus, after detailed analysis, testing and opportunity evaluation from a business strategy point of view these services can substitute the corresponding activities in our scenario while a Web service supporting interaction via MMS, syntax checker and authorization service should be designed from scratch (or discovered among software components or open source projects). The collaborative diagrams can be refined (Activity 7) to allow for the system’s interoperability with the found components. At this step, structure and behavior of existing Web services can be represented by means of UML [17] and included into our model.

Once appropriate Web services have been found, their QoS levels must be evaluated by considering the defined KPIs (Activity 8). Thus, the security indicator of news publisher (see Table 1) means that the interaction Web services have to assure secure connection. The information sharing cost indicator means, in particular, that the total response time of authorization and interaction Web services and system components involved in news submission process must not
Table 2. Examples of Web services discovered for the news distribution process

<table>
<thead>
<tr>
<th>Provider</th>
<th>Web Service</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>richsolutions.com</td>
<td>SmartPayments</td>
<td>Payment Web service that supports credit cards, debit cards and check services</td>
</tr>
<tr>
<td>rpamplona</td>
<td>ACHWORKS SOAP (T$$ - Rico Pamplona)</td>
<td>Web services for ACH Processing and Payments (can accept and process check and credit card payments electronically)</td>
</tr>
<tr>
<td>adambird</td>
<td>Esendex Send SMS</td>
<td>Sends an SMS message to a mobile phone</td>
</tr>
<tr>
<td>StrikeIron</td>
<td>StrikeIron Mobile Email Messaging</td>
<td>Allows SMS to be sent to mobile telephones programmatically for many different service providers</td>
</tr>
<tr>
<td>jmf</td>
<td>SendEmail</td>
<td>Send mail messages to any email address</td>
</tr>
</tbody>
</table>

exceed some reasonable value (e.g., 2 minutes) to deal with urgent news like intermediate scores in football matches or preliminary voting results. If a Web service provider does not guarantee high QoS levels by default, they can be negotiated.

5 Conclusion and Future Work

In this paper we have proposed a methodology for conceptual modelling of Web service-based systems following a business modelling approach. In detail, activity diagrams are useful at Requirements Elicitation stage because abstract services are identified through repeated, similar for different processes and time-invariant blocks of the activities which compose a business process. The choreography design among identified abstract services is based on the interaction patterns among stakeholders, Web services and the system described into the swimlanes diagrams. Moreover, the KPIs based on stakeholders’ goals are useful for defining desired quality levels for each abstract service at Service Identification stage and for evaluating and trading quality parameters of discovered Web services at QoS Negotiation and Service Contraction stage.

Our future work includes further elaboration and verification of the proposed methodology. In particular, tighter connection with Web service specification formats must be established to enable automated support of the design process for WIS.
References

Aligning Web System and Organisational Models

Andrew Bucknell\textsuperscript{1}, David Lowe\textsuperscript{1}, Didar Zowghi\textsuperscript{1}

\textsuperscript{1} University of Technology, Sydney
P.O. Box 123, Broadway NSW 2007, Australia
{andrew.j.bucknell, david.lowe, didar.zowghi}@uts.edu.au

Abstract. A significant area of research is the alignment of web system and organisational requirements. In this paper we describe an approach to facilitating this alignment using graphical models of the processes that are being supported by a web-based system. This approach is supported by the AWeSOMe modelling architecture. This architecture allows us to investigate the effectiveness of different notations for modelling systems. The architecture is being implemented as the AWeSOMe modelling tool, which will be used to investigate our approach to alignment in industry-based case studies.

Keywords: Alignment, Modelling, Web-systems

1 Introduction

Current software development is often characterised by significant early uncertainty in system scope, particularly where the Web is leveraged in supporting changes to business processes. This uncertainty can then lead to poor alignment between business processes and IT systems, substantial ongoing system redevelopment, and client and customer dissatisfaction with the resultant systems. The lack of appropriate approaches to address these problems leads to significant costs in web development projects [11], which is a significant business activity in its own right [6], with cost overruns being a significant waste of resources.

The AWeSOMe tool described in this paper is being developed to support research which investigates an innovative approach to reducing this uncertainty and the resulting volatility, and through this support a more rapid resolution of the development scope for software systems. Specifically, the research will utilise recent progress in the development of high-level modelling approaches [14, 16] (which more effectively link system information management and functional behaviours with business processes) to enable the automated identification of potential discordances between the software system being developed and the organisational context in which the system exists. These discordances arise when aspects of the (proposed) system or business processes are changed without appropriate consideration of the impacts on
the complex inter-relationships which exist within the composite software/business environment.

Specific objectives of the research include:

**Customisation and extension of existing modelling approaches to support identification of key discordances:** Existing modelling approaches can represent the relationship between software systems and business domains [15], but have not traditionally been used to identify or reason about discordances in these relationships when changes are made to either of these (see the section below for an explanation of this issue). We aim to adapt these modelling approaches to allow this reasoning about potential discordances to occur. The AWeSOMe tool will allow us to prototype extensions to modelling notations and use these notations to model systems.

**Development of algorithms for automated identification of discordances:** Once the modelling approaches have been developed, we will develop algorithms to allow reasoning about the models and subsequent automated identification and appropriate reporting of discordances. The AWeSOMe tool will support the integration of software components implementing algorithms that analyse the underlying data model for discordances.

**Evaluation of the effectiveness of the techniques:** The AWeSOMe tool can be used in case studies that demonstrate the modelling and reasoning algorithms and allows evaluation of the resultant impact on the reduction of scope volatility during the development process. Particular attention will be payed to applicability of the approaches across a range of problem domains and technologies, and the applicability to managing real-world projects.

This paper describes the AWeSOMe tool which is being developed to support research into aligning web system and organisation models. We begin by discussing the existing research that is informing this work. We then discuss the problem that this research seeks to address. Next we discuss the research methodology that is being applied to investigate this problem, and discuss the role of the AWeSOMe tool in this research. Next we discuss the conceptual and software frameworks that are supporting the development of AWeSOMe. We conclude by outlining the next stages of our work building on the AWeSOMe tool and a brief discussion of open research questions around this work.

### 2 Background

This project will build upon our earlier research in several key areas. The first area, related to the earlier stages of the Web development process, includes research into Web characterisation models [11], development processes [10], requirements volatility[22], Web Impact analysis [12, 20, 21] and - most recently - our work on the role of issue resolution in supporting the determination of system scope in Web projects. This last body of research is crucial in the context of this project. For the commercial Web projects which we studied, once an initial project brief had been established the key trigger for almost all subsequent adjustments to the project scope was the resolution of issues. These issues related to the way in which the system environment impacts on, and is impacted (or changed) by the introduction of the
This leads us to an approach to scope refinement which will be based on the dualistic modelling of the system and environment as they currently exist prior to the development (an as-is view), and as they are desired to be (a to-be view).

The second area of our previous research which feeds into this project is our development of high-level information flow models [16]. Whilst the detailed behaviour and low-level information design of a Web system tends to be extremely volatile during (and usually after) the system development the flow of information between the system and its environment tends to be much more stable. We subsequently showed [17] that a representation based on high-level information flows provided a clearer basis for system design, and – most significantly – facilitated identification of ways in which a proposed design change may result in impacts upon the systems environment. Further, the information flow models, when combined with conventional work flows (a simplified form of which was shown in Figures 1 and 2 above), appeared to capture a key set of interfaces between the system and the organisation in which it is embedded – those interfaces which, when changed, lead to changes in the scope, and vice versa. The interfaces do not uniquely and completely define the system, but they do appear to provide a source of crucial information for identifying and resolving the discordances that were discussed in the background section above, and which are the focus of this research.

Merging the above two threads of previous research provides a clear approach to supporting the automated identification of those discordances between an IT system and its environment that have the potential to affect decisions on the system scope and its corresponding boundary. These identified discordances can then be used to raise issues within a linked issue-tracking system. The issues, when resolved, will allow a progressive clarification and elaboration of the agreed system scope (as well as the concomitant changes to the associated organisational workflows and business processes).

The approach will be based on the development of a modelling notation which links information flows, functional boundaries and work flows - and which supports the development of models that initially represent the current (as-is) domain. The models are then progressively adjusted to show the incorporation of variations on the proposed IT system, with discordances being automatically identified and issues being raised with the developer as the models are modified. The value of a visual notation that can be evolved can be seen in the success of approaches such as Threat Model Analysis (TMA). Visualising the system helps to make the boundaries more readily apparent, and thus less likely to be overlooked. The existence of the model also allows the impact of changes in the relationships between elements on the security threats to the system to be assessed and addressed. A prototype tool which forms part of this research will allow evaluation of the effectiveness of the models and algorithms in managing the construction and evolution of a system, and identifying boundary conditions that require negotiation between the client and the developer. We believe this approach will lead to a much more rapid convergence of the agreed system scope and a substantial reduction in overlooked adverse misalignment between the system and its environment.
3 Approach

As was mentioned above, a key characteristic of much software systems development is early uncertainty in the system scope. There exists a significant and growing body of research into the early stages of the development cycle when the development scope is nominally resolved - mostly focussing on domain modelling and requirements capture and analysis. Most of this research however assumes an initially unknown but largely invariant system scope which simply must be elicited and analysed. Both research and practice often overlooks the complex interdependencies through which the emerging definition of a system can directly or indirectly affect the environment in which that system exists, and thereby create a feedback loop which leads to subsequent changes in the scope of that system [21]. This type of interdependency between system and environment is most common where part of a business process is being fundamentally changed or replaced by a software system. Whilst this characteristic of system development is not uncommon in most (if not all) software systems, recent technologies, and especially web-based systems, are exemplars of where the scale of the feedback mechanism makes addressing it early an imperative.

Before we consider existing research which is relevant, let us illustrate the above problem with a simplified illustrative example. Consider an existing business process (variants of which exist in many small businesses) that involves casual employees completing a paper-based timesheet at the end of each week which is subsequently checked by a supervisor and either returned for amendment or approved and submitted to a payroll administrator for payment. This is shown in Figure 1 in a simplified process modelling notation, in which the shaded region represents an existing payroll IT system. This figures adopts a simplistic notation, but is used to illustrate the representation of the relationship between the business workflows and the existing (or anticipated) software systems. The modelling notation to be developed will be much more sophisticated than shown in this figure.

Fig 1: Existing casual staff payment process
Assume that a decision was then taken to develop a simple system to support online submission (and modification, when necessary) of timesheets by casual employees. The initial system scope was focused on just support for the casual employees, and could be defined as shown in Figure 2. Such a change in the relevant subsection of the workflow would however have impacts on - and potentially require changes to - other sections of the overall business process. For example, the processing of timesheets by the staff supervisor was previously carried out on the paper-based timesheets, but these no longer exist in that form (now being electronic). The result is a discordance between the new software system and part of the existing process (the checking of the timesheets which assumes paper-based input). Resolving this discordance will involve changes to the scope of the proposed system and/or the business processes, and would be the basis for negotiation with the client.

In other words, when we move from a system as it currently exists (the as-is view of the business/system) to the definition of a potential new or changed system (the to-be view of the business/system) we introduce potential discordances between the system and the business processes to be supported by the new system. Resolving these discordances leads either to changes in the scope of the system, or changes in the associated business processes - either of which can lead to further changes to either or both. Much of the complexity of the early stages of project scoping resolves around understanding and negotiating these changes and defining the boundary of what functionality should “the system to be” support and which it should not. Whilst this form of scope resolution is well accepted and understood, it is our contention that it is invariably overlooked, and there has been little research specifically addressing how it can be most effectively managed. We contend that by undertaking richer forms of the modelling shown in the above examples, based on a merger of existing process modelling techniques and our own information flow modelling formalism...
[16], it is possible to automatically identify potential discordances early in the project scoping / specification, and to then raise these as development issues that need to be resolved - thereby leading to more rapid convergence onto an agreed system scope which is integrated with the modified organisational workflows and business processes.

As mentioned above, there has been substantial research over the last 30+ years, focusing on the early stages of software systems development and the relationships between software systems and business processes. Requirements Engineering (RE) is an active research area which focuses on approaches to capturing, analysing and modelling requirements for software systems. The majority of research in this field assumes a fixed, though initially unknown, scope which must be discovered, analysed and documented. Where requirements are recognised as varying (or volatile [22]) this is usually attributed to uncertainty on the part of the client [13] or changes occurring in the domain independently of the introduction of the system-to-be. Very little research has considered the way in which the introduced system itself can lead to changes in the domain - and hence create a positive feedback loop leading to changes in the system. Our earlier research, which has had a specific focus on Web systems development, has shown that often Web systems and the organisation domain in which the systems exists co-evolve, with the nature and extent of this evolution often not being clearly understood until early design prototypes are available [8,9].

In many respects, this research is also closely related to work on IT-business alignment, which focuses on ways in which a software system can be most appropriately designed to seamlessly integrate with, and support, existing or proposed business activities. Of particular interest is research on strategic alignment [15]. Research has shown that strategic alignment can have substantial positive impact on business performance [4]. Whilst the desired end result is similar (the absence of discordances between the system and the business processes - or the business objectives which are supported by those processes), the focus of work on IT-business alignment is typically on how to ensure that software systems appropriately support a given set of business objectives, rather than the identification of specific aspects where an existing business process requires modification as a consequence of the introduction or modification of a software system to which it interfaces. Furthermore, the research in this area has not paid due attention to the nature and extent of impact that the "system-to-be" may have on the corresponding business rules that govern business processes and workflows within organisations.

Similarly, work in areas as diverse as soft-systems methodologies (SSM) [3], problem frames [7] and COTS (Commercial Off-The-Shelf) development [18] also provides insights into the interdependence of software systems and the organisational processes within which they are embedded. For example, rich pictures - a tool used within SSM and elsewhere - can be used to understand the relationships between software systems and the contexts within which they exist. Nevertheless, again, these techniques are useful in supporting effective system design, but not in identifying specific points of discordance. This identification is typically assumed to occur as a natural consequence of the system design process, and hence has lacked any focus as a research topic. Indeed, this assumption is partially true - the discordances do indeed become obvious - but often not until later in the development cycle, well after design
or indeed implementation has commenced and scoping contracts have been agreed upon and signed off.

Given that there is much work on what defines a software system effectively aligned with organisational processes or business goals, but not on specific techniques for identifying particular points where they are potentially misaligned, the obvious question is how such discordances can be discovered as early in the development as possible and what strategies could be employed in early resolution of issues arising from the identification of these discordances. Answering these questions is the core of this research project.

Our particular approach to answering this question emerges from the convergence of two of our earlier research contributions: investigation of the ways in which issues that emerge during software system design (and Web development in particular) have influenced the developers understanding of the system scope [10,21]; and the potential role in web system design of a high-level information flow model [16]. Taken together, these two areas of work indicate the importance to defining system scope of understanding the flow of information into and out of a system, especially when coupled with an equivalent process flow. These areas of work are the main motivations behind the development of the AWeSOMe tool.

4 Methodology

The AWeSOMe tool is being developed to support investigation of issues around the management of scope and requirements of web systems. The design decisions made when developing AWeSOMe have been guided by these research goals. A brief discussion of these goals follows to show how they have influenced the design of AWeSOMe.

Stage 1: Analysis of existing data on issue resolution and scope refinement: In the first stage of the project, we will consider the question of what system/domain interfaces are associated with those discordances that, when resolved, lead to scope changes. We will analyse the data collected in our earlier research on issue analysis with the specific objective of identifying those issues which related to an identified discordance. These issues will then be analysed to select only those which resulted in a subsequent change or refinement to the perceived or agreed system scope. The AWeSOMe tool provides a realisation of the concepts being discussed in this phase and provides a focal point for discussions. Work on AWeSOMe feeds back into these discussions.

Stage 2: Development of a modelling notation for representing interfaces between a software system and its business domain: Using results from our previous work [16,17] we will develop a rich model that captures the way in which business models and processes inter-relate with IT system designs, particularly in terms of their mutual impacts. This model will be compatible with existing business and system design models (UML for example) and will leverage work that focuses on the impacts IT can have on organisational operation. The modelling notation will be lightweight yet expressive, and be understandable to both developers and clients - thereby facilitating communication between them. Of specific interest will be the i* modeling notation.
and the corresponding framework developed by Yu [19]. The purpose of this modelling is to support identification of misalignments that exist between the IT systems’ core functionality and the organisational workflows (see stage 3). AWeSOMe acts as a tool for prototyping different notations and experimenting with different approaches to modelling. Because it is not tied to any existing notations or frameworks we are free to try out new concepts.

Stage 3: Automated discordance identification algorithms: In this stage we will develop algorithms for analysing the interface models constructed using the notation developed in stage 2. These algorithms will be based on identifying points in the models where the specific modelling semantics have been breached by making changes to the models. In particular, as a model is constructed or changed to incorporate the relationship between a proposed software system and the organisational workflows it supports, the algorithms should be able to identify discordances in these interfaces that have the potential, when resolved, to affect the system scope. We will investigate existing automated reasoning algorithms and technologies to find the most effective one for our purposes. The models created using the AWeSOMe tool are stored in a database in a simple schema that we have created. Using this data model we can create software components that analyse this data using the algorithms developed in this phase. These tools can be standalone or they can be integrated with AWeSOMe.

Stage 4: Implementation of tools to support reasoning: In this component we will develop a prototype tool which supports the construction and evolution of the above models that is suitable for use in industry-based case studies. The AWeSOMe tool will serve as the basis for this implementation, both conceptually and technically. The prototype tool will be interoperable with existing CASE tools so that the models can also form the basis of subsequent modelling (e.g. the models should be able to be exported into skeletons of preliminary UML use case diagrams). Our preliminary design for this tool involves an interface that supports the construction of the as-is model (i.e. existing workflows and information flows) and the subsequent modification of the model to incorporate proposed changes and new system functionality (the to-be model). A rich versioning capability will allow the modeller to wind back or forward the model (or different versions of the changes) in order to consider possible ways in which the scope can be affected. We will also implement in the tool the discordance identification and subsequent issue handling processes. The result will be a composite modelling, versioning and issue tracking CASE tool prototype.

Stage 5: Evaluation and refinement: Supporting all the above project components, and running throughout the project, is a series of user and usage experiments. We will conduct experiments with commercial developers using the model and associated tool to track the extent to which they support effective identification of discordances and raises these as issues to be resolved. Specifically we will consider the extent to which the identified discordances are valid and complete, and the extent to which their resolution affects the agreed system scope (or at least led to valid discussions about the scope). We will also conduct case studies of the use of the tool. Evaluations will be performed both with AWeSOMe and with the tool developed in stage 4.
5 Conceptual Framework

A key purpose of the AWeSOMe tool is to assist with the development of new techniques and methods for building web systems. These techniques and methods are based on the existing body of work relating to Web Engineering. In particular we build on the concept of alignment in systems modelling [1,2,4], existing approaches to modelling systems [14], and existing notations used for modelling systems [14]. The use of these concepts in this work is introduced below.

5.1 Alignment & Discontinuities

A system can be aligned, misaligned, or unaligned. An unaligned system can not be realised. It is a system that has discontinuities. A misaligned system is one where there are gaps between the web system and the organisational processes it is supporting, but these gaps have been identified and steps have been taken to ensure the process still works. Sometimes a physical change such as printing hard copy versions of electronic documents is an acceptable workaround. When a discontinuity is resolved with a workaround rather than with a change to the software scope we say there is a misalignment, but the discontinuity has been resolved. An aligned system is one where the web system totally encompasses the organisational processes. A discontinuity is a discordance that has not been resolved. A model is aligned when all its discontinuities have been resolved. A discordance can be changed either by modifying the software system or the physical system. AWeSOMe is developed to help developers of web systems identify these discordances early in the development cycle, when the cost of rectifying them is lower than it would be otherwise.

5.2 Notations

A key component of this research is developing a notation that can be used to create models of web systems and organisations. There are numerous notations available for modelling web systems and our own research has also developed a useful notation [16]. The AWeSOMe tool seeks to build on this research and support further research into notations that are useful for modelling web systems. In particular, we believe it is essential for many projects to have a simple notation that can readily be applied to real world problems. While BPMN, BPEL, IDEF(n) and the like are extremely useful, there is often a great deal of overhead and cost involved in integrating these modelling notations and techniques into an organisation. We aim to develop a simple notation that will be easily adopted for projects of any size, while still retaining the richness of expression that makes other notations useful.

An example of the kind of notation we are developing is shown in Figures 1&2. This notation describes a system in terms of Actors, Activities, and Artefacts, and the flows between them. The flows indicate, for example, that an Actor can perform an activity, and that an Artefact can be the input or output of an Activity. This notation is not well developed, but with the AWeSOMe tool serving as a prototype we can easily modify notations and evaluate how useful they are for modelling actual systems.
6 Software Framework

The concepts developed in this work are intended to be applied to real world problems. While many of the concepts are widely used in existing tools, we believe our approach to integrating them to be unique. For this reason we have chosen to develop a software framework that supports the creation of tools based on the ideas developed in this work. These tools aim to be useful both as research aids as we develop our ideas, and as tools that can be used in web engineering projects. The following discussion presents our approach to AWeSOMe’s architecture and a brief discussion of its key components.

6.1 4-Layer Architecture

In order to support our research goals we have chosen to base AWeSOMe on a 4-layer architecture as used in UML. Figure 4 shows a linear representation of the UML 4-layer architecture on the left and the corresponding layers used in the AWeSOMe architecture.

![Diagram of 4-layer architectures for UML and AWeSOMe.]

The UML layers are described in the UML 2.0 specification [cite]. In the AWeSOMe architecture, the significance of the layers is as follows:

- M0 - the physical system that is being modelled. This can be an implementation of a web system, or the processes and workflows of a business.
- M1 - a model representing the physical system. The model is an abstraction that allows aspects of the physical system to be represented in a data structure that supports reasoning about the system. The model is expressed using a notation.
- M2 - a notation for describing models of organisations or web systems. Common examples of the kinds of notations describe are BPMN, UML Activity Diagrams, and WIED.
M3 - a model for describing notations. The M3 layer is constructed to be as simple as possible while still allowing a variety of notations to be expressed. The M3 layer describes notations in terms of the entities that can be used in the models and the relationships that can be created between these entities. The M3 layer is expressed as a database schema.

6.2 Modelling Framework

The modelling framework is a collection of database schemas, software interfaces and component designs that support the implementation of a notation independent tool for modelling web systems and organisations. Figure 5 shows a conceptual view of how the 4-layer architecture is realised as a set of software components in our implementation.

6.3 Database Schema

The M3 database schema describes a way of representing notations. We have kept this description as simple and as minimally abstract as possible. In our schema we say that a Notation consists of Entities and Relationships. Both of these can have Attributes associated with them. We also support the notion of superclasses and abstract Entities. In the example notation described above, the Entities are activities, actors, and artefacts. The Relationships are invoke and produce. Relationships are defined as having to and from entities, so when modelling the notation we would say the from-Entity for invoke is actor and the to-Entity for invoke is artefact.
6.4 M2M odeller

The M2M odeller allows us to model notations that are used to model web systems or organisations (or both). Modelling different notations allows us to experiment with different approaches to modelling web systems and organisations. Our goal is to develop a notation that is simple to use but effective for modelling web systems and organisations. The M2M odeller allows us to capture modelling semantics and layout semantics. In our implementation the M2M odeller is a web application developed in Java using the Struts framework and Hibernate. Notations created in the M2M odeller are stored in the Notation Store. The Notation Store is implemented as a Hibernate persistence layer, and in the current implementation also uses a MySQL database.

6.5 M1M odeller

The M1M odeller allows us to model web systems and organisations using any of the notations modelled in the M2M odeller. It also allows us to manage branches and evolutions in the model. The M1M odeller will also be used to interact with software components that help to resolve alignment discontinuities in the model being created. In our implementation the M1M odeller is a 2-tier client-server application. The server layer consists of a collection of web-services that manage the M1 data and layout models. This layer is implemented using Apaches Axis web-service framework and Hibernate. These services are consumed by a C# application that allows the user to create and manipulate models of physical systems using notations that have been added to the system. The model created using the M1M odeller is stored in the Model Store. The Model Store is implemented as a Hibernate persistence layer, and in the current implementation also uses a MySQL database. Figure 5 shows an example of modelling the timesheet system discussed earlier using AWeSOM e's M1M odeller.
7 Further Work

The development of the AWeSOMe architecture and the AWeSOMe modelling tool is being undertaken as part of our ongoing research into the modelling of web-based systems. The architecture and tool serve two key purposes. The first purpose is to provide a framework for the investigation of the various research questions raised above, particularly through use of the tool in case studies. The second purpose is to serve as a discussion point for research questions around requirements engineering, process modelling, and web systems modelling.

7.1 Planned Outcomes.

The AWeSOMe modeller is being developed to support ongoing research into the development of Web Systems. This research intends to have several outcomes, which will be actualised in the AWeSOMe tool and in related tools the investigation identifies as being necessary. These outcomes include:

- An improved understanding (represented as models) of the interdependence of organisational workflows and the software systems which support these workflows, and particularly with the way in which appropriate modelling (based on a composite of workflow modelling and information flow modelling) of these interdependencies can lead to identification of discordances between the workflows and the systems.
- A set of techniques for undertaking reasoning about the models that are constructed and the subsequent automated identification of those discordances between...
software systems and the workflows that are supported which have the potential to affect, when resolved, the agreed scope of the software system.

- A prototype tool which supports the construction of the models and subsequent reasoning about these models, and which thereby assists in identifying misalignments between software systems and the workflows that they support.
- Integration of the prototype tool with existing widely-utilised software product development tools which thereby allow the rapid adoption and leveraging of the models and techniques which will be developed.

7.2 Broader Research Questions

An important goal in the development of the architecture and tools described in this paper is to encourage discussion about issues relating to requirements engineering, web-systems modelling, and the alignment of web systems and organisational models. These discussions are ongoing within our research group, and many of the issues raised are of interest to the broader Web Engineering community. Some key issues that are particularly relevant are:

- What are the modelling requirements for needs driven development and how do these differ from those of vision or goal driven development. What are the implications of these differences for the development process? If we take the view that modelling a web-system is about interpretation of goals into requirements as mediated by the design process, what modelling techniques should be supported?
- Is there an optimal sequencing for the resolution of system requirements that impact on the development and adoption of a system? Can we use this sequence to develop heuristics that allow the identification of high impact issues early in the development cycle and guide the adoption of development pathways?
- What is the impact of the introduction of a new web-based system on existing business processes? Can we identify these impacts early in the development cycle so that any negative impacts can be mitigated?
- How can non-conformances between business processes and web-system requirements be identified before a web-system is deployed in an organisation? Is there a threshold that makes an acceptable level of alignment between a web-system and the business processes it supports?

References


International Conference on Web Engineering 2007

6th International Workshop on Web-Oriented Software Technologies

Organisers
Luis Olsina, University Nacional de La Pampa (Argentina)
Oscar Pastor, University Polytechnic of Valencia (Spain)
Daniel Schwabe, Department of Informatics, PUC-Rio (Brazil)
Gustavo Rossi, LIFIA, UNLP (Argentina)
Marco Winckler, University Paul Sabatier (France) & UcL (Belgium)

Workshop program committee members
Simone Barbosa, PUC-RIO, Brazil
Birgit Bomsdorf, University of Hagen, Germany
Olga De Troyer, Vrije University of Brussel, Belgium
João Falcão e Cunha, FEUP, Porto, Portugal
Peter Forbrig, University of Rostock, Germany
Martin Gaedke, University of Karlsruhe, Germany
Geert-Jan Houben, Vrije Universiteit Brussel, Belgium
Nora Koch, Ludwig-Maximilians-Universität München, Germany
David Lowe, University of Technology, Sydney, Australia
Maria-Dolores Lozano, University of Albacete, Spain
Kris Luyten, Hasselt University, Belgium
Nuno Nunes, University of Madeira, Portugal
Luis Olsina, University Nacional de La Pampa, Argentina
Philippe Palanque, LIHHS-IRIT, University Paul Sabatier, France
Fabio Paternò, ISTI-CNR, Italy
Vicente Pelechano, Polytechnic University of Valencia, Spain
Oscar Pastor, Polytechnic University of Valencia, Spain
Gustavo Rossi, LIFIA, UNLP, Argentina
Daniel Schwabe, PUC Rio de Janeiro, Brazil
Gerd Szwillus, University of Paderborn, Germany
Marco Winckler, LIHHS-IRIT, University Paul Sabatier (France) & ISYS-IAG,
Université catholique de Louvain (Belgium)
Quentin Limbourg, SmalS-MvM, Belgium
Table of Contents

Automatic Display Layout in WebML: a Web Engineering Approach. Sara Comai and Davide Mazza ........................................... 149

A MDA-based Environment for Web Applications Development: From Conceptual Models to Code. Francisco Valverde, Pedro Valderas, Joan Fons and Oscar Pastor Lopez .......... 164

Enriching Hypermedia Application Interfaces. Andre Fialho and Daniel Schwabe ................................................................. 179

Modelling Interactive Web Applications: From Usage Modelling towards Navigation Models. Birgit Bomsdorf ............... 194
Automatic Display Layout in WebML: a Web Engineering Approach

Sara Comai and Davide Mazza
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza L. Da Vinci, 32, Italy
comai@elet.polimi.it
davide.mazza@asp-poli.it

Abstract. In this paper we present an approach for the Automatic Display Layout of Web applications based on the WebML modeling language. The WebML development method is model-driven: its conceptual models contain several elements that can be exploited for defining a procedure for the automatic displacement of page contents. In this paper we describe the development life cycle of our Web Engineering approach and, focussing on the presentation design and on its layout, we highlight which aspects of the Human Computer Interaction approaches are implicitly considered.

1 Introduction

Automatic Display Layout (ADL) is the procedure for automatically positioning the content of a diagram or of a document. In particular, in literature it has been applied for the computation of the layout of nodes and links of graphs, in a similar way for the positioning of electronic components over boards, or for the layout of text, figures, tables, etc. in document presentations [7].

In this paper we consider the Automatic Display Layout of Web pages specified with WebML [16]. WebML relies on a well defined development method that starts from two conceptual models: the data model and the hypertext model. Starting from such models, we have defined a procedure that performs the automatic layout of the contents of the specified Web pages. The concepts defined in both models are exploited to assign a position to each page component. The proposed ADL procedure is based on the specification of layout rules: different classes of rules are defined, taking into account the different elements of the two conceptual models.

The layout of Web applications, and more in general of user interfaces, is a classical problem treated in the human computer interaction (HCI) field: their focus is mainly on the appropriateness of the layout and on the usability of the applications.

In this paper we present a Web Engineering approach that implicitly considers some aspects of the Human Computer Interaction techniques. In particular,
we revise the whole development process adopted in WebML in order to highlight in which phases the aspects that are important for HCI are, at some extent, evaluated.

The paper is organized as follows: Section 2 explains the Web Engineering process followed in the development of data-centered Web applications. Section 3 introduces a running example that has been used as a case study. Section 4 presents the main concepts of the WebML language needed to understand the proposed approach. Section 5 presents the proposed ADL approach in detail and Section 6 presents the test results we have obtained by applying the automatic procedure on the running case. Section 7 discusses related work, and, finally, Section 8 draws our conclusions.

2 The Web Engineering process for Web applications development

The Web Engineering process for the development of data-driven Web applications is in line with the modern methods of Software Engineering, where the different development phases are applied in an iterative and incremental manner, and the various tasks are repeated and refined until results meet the business requirements. Figure 1 shows a diagram that illustrates the different steps of the Web development process.

The input of the process is the set of the business requirements, i.e., of the goals that the application is expected to produce to its users and to the organization who builds it. In particular, the first step of the development process consists in the specification of these requirements: in this phase the business actors are identified and functional and non-functional requirements are collected. The former address the essential functions that the application should deliver to its users; the latter include factors like usability, performance, scalability, and so on. In particular, usability addresses the ease of use of the application, which is determined by the ease of learning of the user interfaces, the coherent use of the interaction objects across all the application interfaces, the availability of mechanisms for orienting and assisting the user, etc.

The second step is represented by the design of the data model, representing the domain of the application. This is a very important phase for data-intensive Web applications: data include the core objects managed by the applications, their relationships, possible auxiliary objects used to classify or specialize the applicative objects (e.g., categories) having the purpose of facilitating access to the application content; also static data may be included.

The third step consists in the specification of the hypertext as the set of links and pages that will form the front-end of the application. The functional requirements specification is transformed into an hypertext model delivering data and services. Usability is a fundamental quality factor that must be considered at this stage: indeed, it is particularly relevant in Web applications, where it is essential to attract users and facilitate them in visiting a site. Hypertext usability may be enhanced by carefully choosing the most suitable patterns of pages and
content [6], based on the user requirements and on the function delivered by a page. Also navigation aids can be provided: they include all the auxiliary mechanisms that help users in exploring the Web site, like navigation bars or shortcuts to most frequently accessed pages. Orientation aids like breadcrumb links, showing the current status of the navigation and the position of the current page within the global structure of the hypertext can also help the orientation of the user.

The fourth step consists in the **presentation design**. This step requires addressing two distinct concerns: **graphic properties** and **layout**. Graphic properties are usually defined by graphic designers and refer to the look and feel of the Web pages. The layout specifies the organization of a Web page, which may include multiple frames, images, static texts, and so on. This part will be explained in more detail in Section 5.

The next steps concern the **deployment** of the Web application: the final application (data and hypertext) must be implemented, tested, and maintained on top of a given architecture, which must also designed and put in place.

In this paper we focus on the layout of Web pages. We will see how it can be specified as an automatic procedure based on the data and on the hypertext models, and will present the results we obtained on a case study.

---

**Fig. 1.** The Web Engineering development process.

---

### 3 Running example

The running case we consider throughout the paper is the webml.org site, which is the official Web site of the reference formalism of this document. Users accessing the site vary from novices, looking for introductory documentation on WebML, to experts, looking for guides or handbooks, to students of courses on Web or Software Engineering. The site provides downloadable didactic material, contains the main research results, and the list of the people who have contributed to the definition of WebML or currently employed in the related
research. Such information are contained in a public area of the site. Beside this public area, there are two other private areas that can be accessed only by registered users or administrators. A first area has been designed for the content management of the site, thus supporting the creation of new contents or the update/deletion of existing data. A second area implements some interaction tools, such as a mailing list, for supporting discussion on the research topics within the WebML community.

4 WebML in a nutshell

In this section we review the main concepts of WebML that are needed to understand the proposed layout procedure. For further details on the WebML model the reader may refer to [2, 3].

WebML is based on two main models: the data and the hypertext model.

The goal of the data model is enabling the specification of the data used by the application. WebML exploits the most successful and popular notation, namely the Entity-Relationship (E-R) model, which includes the essential data modeling concepts sufficient to specify the data schema of a Web application. The ingredients of the Entity-Relationship model are entities, defined as containers of structured data, and relationships, representing semantic associations between entities. Entities are described by means of typed attributes, and can be organized in generalization hierarchies, which express the derivation of a specific concept from a more general one. Relationships are characterized by cardinality constraints, which impose restrictions on the number of relationship instances an object may take part to.

Figure 2 shows a little fragment of the WebML data model of the running example. The entity WebMLConcept contains information about the relevant concepts of the WebML formalism. Each WebMLConcept can have child concepts: this concept hierarchy is implemented using the one-to-many relationship defined on the same WebMLConcept entity. The site contains data about the WebML book, which is organized in parts, represented by entity Part. Each part is formed by one or more chapters (contained in entity Chapter): the containment of chapters in parts is expressed by the one-to-many relationship between the two entities. Entity Chapter contains the abstract and other information about each chapter of the WebML book. Each chapter can refer to one or more WebML concepts, as specified by the many-to-many relationship with the WebMLConcept entity. Finally, the TextChunk entity works as a repository for chunks of static text that have to be shown in pages (such as introductions, instructions on how to complete a form, etc.). This is a support entity, not useful to model the reality of the problem.

The goal of the WebML hypertext model is the specification of the organization of the front-end interfaces of a Web application. It addresses the logical division of the application into modules targeted to different classes of users, the organization of large applications into sub-modules, the definition of the content of the pages to deliver to the users, and the links to support user’s navigation.
and interaction. The key ingredients of WebML are pages, content and operation units for delivering content or performing operations, and links. **Content units** are the atomic pieces of publishable content; they offer alternative ways of arranging the content dynamically extracted from the entities and relationships of the data schema, and also permit the specification of data entry forms for accepting user input. Units are the building blocks of **pages**, which are the actual interface elements delivered to the users. Pages are typically built by assembling several units of various kinds, to attain the desired communication effect. Page and units do not stand alone, but are linked to form a hypertext structure. **Links** express the possibility of navigating from one point to another one in the hypertext, and the **passage of parameters** from one unit to another unit, which is required for the proper computation of the content of a page. A set of pages can be grouped into a **site view**, which represents a coherent hypertext serving a well-defined set of requirements, for instance, the needs of a specific group of users. WebML also allows specifying **operations** implementing arbitrary business logic; in particular, a set of data update operations is predefined, whereby one can create/delete/modify the instances of an entity, and create or delete the instances of a relationship (the latter operations are called connect and disconnect, respectively).

The WebML hypertext model represents a way to specify the tasks to be performed by the user, even if at a low level of abstraction. This is due to the nature of WebML, that requires the specification of the contents of each single page and their connections. Therefore, it allows the definition of tasks at a ready-to-implementation level, rather than a more abstract one, like in the hierarchical approaches usually exploited in HCI [13, 15].

As an example, Figure 3 shows the way in which WebML allows the specification of a selection task. The left-most unit is an index unit showing a list of book parts to the user. The right-most unit is a data unit showing the details about the item selected from the index. The link that connects the two units carries the parameters provided by the source unit and used for computation by the destination unit and is defined as a navigational link. Such link imposes a predefined order in the execution of the units to be displayed; more in general, navigational links can imply the execution order of the users’ tasks.

Fig. 2. A data model example.
Presentation design in WebML is mainly split into two phases: the definition of the style and the specification of the layout. The former is established during the requirement analysis. Style guidelines are specified to define:

- A **page grid**, i.e., a table containing a specific arrangement of rows, columns, and cells where the content must be positioned (see Figure 4).
- The **content positioning specification**, addressing the rules for assigning standard content elements, like banners, menus, and submenus. Such guidelines can reduce the cognitive overhead of the user during the application learning phase, because they force elements with similar semantics to be placed in the same position across different pages.
- **Graphical guidelines**, containing the formatting rules for graphic items, like fonts, colors, borders, and margins. They can be expressed by means of Cascading Style Sheets rules or equivalent specifications. They can refer to the whole page or in WebML they can also be specified for the single types of content units.

Style guidelines are defined by graphic designers and are often embodied into page mock-ups that are sample representations for a specific device and rendition language.

In a second phase and, in particular, once the data and the hypertext model have been defined, the layout of the content units of each WebML page must be specified. In the most common data-intensive Web applications the main content of the pages is usually placed in the central grid, where contents are displaced in multiple rows or columns; further lateral areas on the left and one on the right of the grid may be present, as shown also in Figure 4.

Currently, in the WebML methodology (and also in the commercial tool supporting WebML [17]) the layout of the contents is performed manually by the Web application designer, who positions each unit of each WebML page inside the content grid. In the next sub-sections our focus will be only on this second phase, which can be automatized through the specification of a set of rules exploiting the main concepts of the data and hypertext WebML models. We will present the main classes of rules that have been identified by means of simple examples.
5.1 Exploiting hypertext patterns

We have seen that hypertext design should already consider usability aspects. In particular, it should guarantee consistency: conceptually similar problems should be given comparable solutions. Thus, similar user-tasks should be implemented with the same content/operation units. A consistent use of composition and navigational patterns helps the user build reliable expectations about how to access information and perform operations; by applying past experience he can predict the organization of an unfamiliar part of the applications. Consistency applies not only to composition and navigation, but also to presentation, and in particular to the layout of pages. For this reason a first class of layout rules exploits hypertext patterns.

Common patterns in Web applications have been identified in different works [6, 11]. Here we show a simple example of a modify pattern expressed in WebML. The hypertext fragment in Figure 5 represents the update of a book part. This operation is available in the private area of the application. The page is composed by a data unit Selected book part containing the details of a selected part and an entry unit Modify book part, representing an entry form for collecting the data of the modified part. The data of the selected book are available in the form fields; they are passed from the data unit to the entry unit as parameters on the link between the two units. When the user submits the data entered in the form (represented by the solid-line link exiting the entry unit) a Modify unit modifies the instance of the selected book.

Similar patterns can be found in pages that perform the insertion or deletion of data, or the visualization of information by means of subsequent levels of detail. An overview of the WebML patterns can be found in [6].

Pages that adhere to a specific pattern, should dispose the correspondent components with the same layout: pattern rules associate a layout to a pattern.
Beside the common patterns identified in [6], a Web application may contain other application-dependent recurrent structures. The specification of layout rules can be generalized to any pattern appearing in a Web application.

Figure 6 shows the definition of a pattern-based rule for the Modify pattern of Figure 5. The rule has a left-hand side and a right-hand side. The former represents the hypertext structure a page must contain to apply the rule; the latter represents the corresponding layout in our reference presentation model. Any page containing a data or an index unit, providing content to an entry unit used for modifying the fields of the selected object, will have the layout specified on the right-hand side: the data or index unit is displayed at the top, the entry unit at the bottom.

5.2 Associating a role to components

When a page does not adhere to a pattern, or when the page presents a pattern structure but some other components remain, there is the need to find some general criteria to displace its content. Two further classes of rules are defined, presented in this section and in the next one, respectively.
A set of rules can be specified for the layout of particular components, having a particular role in the global context of the page. We call such rules *role-based rules*. Examples of components associated to a particular role are: static texts to be included in particular positions in the Web pages, e.g., at the beginning of each page; the list of languages in a multi-lingual application which should always appear in the same position across the different Web pages, and so on.

In general, the role of entities having specific semantics cannot be derived from the data and hypertext models. Such information should be specified during the ADL procedure. Particular roles can be associated to units defined over a particular entity or using particular attribute types.

As an example consider the introductory chunks of static text, which in our running example are usually positioned in the first cell in the main grid of each page. These text chunks are stored in the database in a dedicated entity, as for all the other data of the application. In the proposed approach the designer can mark an entity as having a particular role, so that we can make the automatic display layout procedure aware of the particular role of the data contained in that entity. Units based on marked entities can then be positioned in a particular cell.

To assign a position to the units defined on entity `TextChunk` the following rule can be defined: `Textchunk \rightarrow (grid, top)`. It associates the top position in the central grid (in alternative also the left and right column could be specified) to the units defined over entity `TextChunk`. Notice that such rule assigns a *fixed position* inside the grid.

Figure 7 shows an example of application of such role-based rule: it shows the fragment of the data model used in the hypertext, an hypertext instance satisfying the rule (the data unit `The idea`, identified with (1), is based on the entity `TextChunk`), and its final layout. Figure 8 shows the final rendering of the page and highlights the positioning of the data unit `The idea`.

The positioning of the remaining units is explained in Section 5.3.

![Fig. 7. Application of the role-based rule.](image-url)
This example associates a particular role to units based on a particular entity. Particular roles can be associated also to attribute types, e.g., to all BLOB objects or to all the pictures, and so on. For example, all the pdf downloads in the case study always appear in the same position, to guarantee consistency in the whole application.

5.3 Exploiting the topology of the hypertext and the relationships of the data model

When pattern- and role-based rules cannot be applied to the units of a page, the content can be positioned according to the relationships between the entities on which the different units of the page are based. The content of each WebML page can be seen as a set of connected graphs having as nodes the units and as edges the links; each connected graph is treated separately. Among the different units to process in a connected graph, a starting point must be identified: this should be the unit that can be considered the most important within the page. From the analysis of several real applications developed using WebML, such unit can be identified looking at the number of outgoing links: the unit having the maximum number of outgoing links is usually the most important one. This is usually positioned in the main area of the page. Then, for the remaining units a relative position can be identified, by adopting an heuristic: the graph is traversed and the source entity of each child unit is compared with the one of its parent unit in the graph. The cardinality of the relationship between the entities over which the units are defined can be used to determine the relative position as follows:

- if the relationship between the entities is one-to-many, the child node is positioned to the right of its parent unit;
if it is one-to-one, it is positioned in the same area of the parent unit and below the parent unit;
if it is many-to-one, it is positioned in to the left of its parent unit.

Such heuristics have been proven to be valid on several real industrial WebML applications that have been analyzed.

Figure 7 shows an example of relationship-based layout. The considered graph is the one formed by the two units identified with (2) and (3). According to our algorithm The WebML models data unit is positioned first, because it has the maximum number of outgoing links, and it is placed in the central area. The index unit (WebML models) is a child of the former unit, and because of the one-to-many self-relationship on the WebMLConcept entity, it is positioned in the right column. Figure 8 shows the final rendering of this page and highlights the positions of the two units.

6 Validation of the procedure

The procedure has been tested on our running case and on several industrial applications developed with WebRatio. Here we present in detail the results on our running case; they are reported in Table 1. They have been obtained by comparing the layout manually computed by the designer of the application with the layout suggested by our automatic procedure.

The webml.org application is globally composed of 173 pages, containing 640 content units and 155 operation units. Its data model is composed of 34 entities and 80 relationships.

The following layout rules have been defined for this running case:

– The common patterns like the Create, Modify, Delete and so on are defined by default, due to their high usage and diffusion in all Web applications. A layout has been assigned to each of them. Four other application-specific patterns have also been defined.
– Two role-based rules based on entities have been defined: one associated to the TextChunk, representing introductory texts to be positioned at the top of the grid area; another one associated to the News entity (not represented in the fragment of the data model), which positions all the news in the right column.
– The relationship rules have been defined by default according the proposed heuristics.

From the table it can be noticed that in most of the pages the 70-80% of the units are automatically positioned like in the designer’s layout. We have considered both the exact matching and an aesthetical equivalent matching of layouts; the latter allows us to accept an automatically computed layout if its appearance is the same as the one of the manual one: we consider two layouts aesthetically equivalent if the relative positioning of each pair of units is the same, independently on their positioning in the central grid, in the left or in the right columns.
Table 1. Results of the webml.org site test. The number of pages (matched, or total of the siteview) is indicated in brackets.

<table>
<thead>
<tr>
<th>Siteview (Pages)</th>
<th>All rules</th>
<th>No patterns</th>
<th>No roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public area (67)</td>
<td>79.10% (52)</td>
<td>74.63% (50)</td>
<td>56.72% (38)</td>
</tr>
<tr>
<td>Administrative area (87)</td>
<td>77.01% (66)</td>
<td>71.26% (61)</td>
<td>73.56% (63)</td>
</tr>
<tr>
<td>Community area (19)</td>
<td>89.47% (16)</td>
<td>89.47% (16)</td>
<td>89.47% (16)</td>
</tr>
</tbody>
</table>

Column *All rules* shows the results obtained by applying the three classes of rules. Column *No patterns* shows the results obtained without applying pattern-based rules; finally, column *No roles* does not consider the role-based rules. It can be noticed that patterns and roles improved the results (compare the second and third column with the first one, respectively). Relationship-based rules have been applied in the 40% of the pages.

The public area, compared to the private areas, makes a heavy use of role-based rules. The administrative area instead is based on patterns; however, such patterns are quite simple and often match also the relationships layout rules. Many pages in the community area contain very few units, and therefore are easily positioned in the grid, also using only the relationships layout rules.

The cases where the layout is different in the manual and in the automatic approaches (10-25%) regard pages containing a great number of units, where neither pattern rules nor role-based rules can be applied. From the analysis of the manual layouts no logics in the positioning of the units can be derived and therefore specified with the third class of rules introduced in Section 5.3. Moreover, for such cases it is also very hard to decide which layout can be considered correct between the automatic one and the manual one.

The obtained layouts can be considered as an initial layout, and a further manual refinement is still possible by the developer.

7 Related work

In literature the ADL problem has been applied in different fields. Early approaches have been applied to the computation of the layout of graphs, electronic components on boards, or document presentation. Two possible approaches are worth notice: a *constraint-based* approach and a *rule-based* one.

Constraint-based approaches are based on the specification of a set of constraints given in input to a solver which takes them into consideration during the computation of the final layout. This is usually the path followed by graph layouters, where constraints typically consist in the minimization of edge crossings or of occupied space. Algorithms used in this approach vary from the classical Constraint Satisfaction Problem [1] to the evolutionary techniques like the genetic algorithms [9].

Rule-based approaches are based on a set of rules, fixed or customizable by the user, which are subsequently applied until all elements to displace have been
positioned. Like in our approach, rules usually have a left-hand side (LHS), which specifies the application conditions, and a right-hand side (RHS), which specifies the operations to do or the state to which the current situation has to evolve. This approach is usually applied to the document presentation. A survey of the approaches developed for information presentation can be found in [7].

Solutions of the ADL problem applied to Web-oriented documents are instead less diffuse. To our knowledge, those that can be found in literature have been considered especially from an HCI point of view. All of them mainly focus on the concept of task as operation to be done by the user. Instead, no Web engineering methodology copes in detail with this problem.

However, the different classes of rules considered in our approach blend together aspects already treated in literature.

First of all we have considered the usage of patterns. Navigational patterns are a set of navigational macro-structures largely used in nowadays Web applications (e.g. Shopping basket, News, etc.) [11]. In [11] they are defined at a higher level of abstraction, taking more into consideration the usage that each pattern allows to do, rather than its actual implementation. Our approach starts from navigational patterns specified in an actual design language and focuses on their layout.

In an analogous way, [13] considers the domain model (similar to our data model), the interaction of the user and the task to be done in a user interface. The paper identifies some ready-to-use solutions as presentation patterns, each one for a specific task situation. Each solution has been defined with care to usability. Compared to this work, our focus is on the realization of the final layout, rather than on presentation schemas.

The work [10] proposes another approach for the design of a Web application based on the tasks the user has to do with the application. The information derivable from the data and navigational models is not enough expressive to represent the mental model the user has in approaching and using the Web application. Therefore they first model the users and the tasks to be done by each single user, using a notation called CTT; subsequently, the navigational model is specified using the StateWebCharts notation [14], which exploits the previous definitions of users and related tasks. Also this work does not focus on a specific implementation of the application, while our main objective is the creation of the actual layout of the applications. The SWC notation and WebML present some commonalities: in particular, SWC is formed by different states in which the user can be and the navigation from one state to another are triggered by users’ actions; the formal model of execution of WebML is very similar [4]. The CTT approach, proposed in [15], expresses the user-task point of view with a dedicated notation, while in our approach the user-centered aspect is implicitly considered by the patterns (and more in general by the WebML hypertext specification), that can be seen as another way to model the user tasks.

Other works try to give a measure of the usability of the layouts. [12] proposes a way for measuring the usability of a user interface by detecting the sum of all
the distances the user mouse pointer has to cover in order to completely perform a task. The layout is therefore defined as appropriate if it allows to minimize the covered distance. [8] is instead a work based on the notion of visual balance: it considers how the human eye first looks at some details of the image rather than others by noticing its features (dimensions, colours, shape, ...) and by weighting them in the global context of the image. According to this notion, it builds a weighted map of the contents to displace and positions them in order to balance the overall weight of the layout. Our approach does not consider any measure of the usability of the layouts. However, it is possible to analyze the quality and usability of the WebML hypertext specification as shown for example in [6, 5], on which we base our layout algorithm.

8 Conclusions

In this paper we have illustrated a procedure developed to perform the automatic layout of the content of Web application pages, considered from a Web Engineering point of view. The proposed approach is based on patterns, which allow to define similar layouts to similar tasks, on the definition of role-based rules, which allow to constrain the position of particular pieces of content, and on the exploitation of the relationships existing among the different units of a page. Layout computation is strongly related to usability, because of the vital importance of the user interfaces, especially in Web applications. The presented approach is not explicitly based on HCI concepts. However, many aspects typical of HCI are considered during the requirement analysis phase and reflected in some way (e.g., by means of the usage of patterns) in the hypertext model, which is then taken as basis for specifying the layout rules. The way in which WebML units are positioned with our rules guarantees consistency in the presentation of the user interfaces, which in large applications cannot be guaranteed by manual layouts. This is especially true for pattern-based rules: however, since the layout is assigned by the designer and can be arbitrarily complex the usability for such pattern should be carefully evaluated by the designer defining the rules. Role-based rules, which constrain some contents to be positioned in predefined places, are instead a way to highlight the most important information. Finally, it can be noticed that the third class of rules, which considers the relationships between the different units and clusters related information, allows the user to easily catch such relationships.

The proposed approach represents a first experiment in the automatic display layout of the WebML methodology. Our point of view originates from our experience in the Web Engineering field, but its models and the proposed approach implicitly consider HCI concepts such as usability and user-tasks. We believe that there are several common aspects between a Web Engineering methodology and the HCI techniques and that discussion between the two communities could actually improve our Web Engineering process and the automatization of some tasks concerning the user interface, like the layout of the Web contents.
References

A MDA-based Environment for Web Applications Development: From Conceptual Models to Code

Francisco Valverde¹, Pedro Valderas¹, Joan Fons¹, Oscar Pastor¹

¹ Department of Information Systems and Computation, Technical University of Valencia, Camino de Vera S/N, 46022 Valencia, Spain
{fvalverde, pvalderas, jjfons, opastor}@dsic.upv.es

Abstract. Nowadays, MDA is gaining popularity as a feasible way to develop software in Web environments. As a consequence, several tools from both academic and industrial contexts, offer their own MDA processes for producing Web Applications. OO-Method is an object-oriented method that produces software systems by means of its MDA implementation, OlivaNOVA. This tool has been broadly tested in industry with real desktop applications. However, it lacks the expressivity needed to accurately describe Web Systems. OOWS is the web-oriented extension of OO-Method with was developed to solve this problem. This work, presents a MDA development environment that combines OO-Method and OOWS. This environment produces fully functional Web Applications that integrate the business logic generated by OlivaNOVA with a Web Interface produced from OOWS models. The followings tools are introduced to support OOWS development process: (1) an Eclipse-based modeller to edit OOWS models visually (2) a Web Interface Framework that is based on Software Factories philosophy in order to reduce the abstraction level between conceptual models and the code to be generated and (3) a set of Model-to-Text transformations that allows the automatic generation of a Web Interface from models. This work also describes a strategy to include the OlivaNOVA development process into the new MDA development environment.

1 Introduction

Model-Driven Software Development (MDSD) is a discipline that is starting to provide promising results. There are several optimistic indicators regarding the evolution and the industrial implementation of this development philosophy. For example, the increasing popularity of the MDA standard proposed by the OMG [17] or the Software Factories philosophy [9] promoted by Microsoft. Perhaps, the most promising indicator is the continuous development of tools and technologies for building CASE Tools to support MDSD, such as the Eclipse Modelling Project [6], and the DSL Tools [13] that are integrated into Microsoft Visual Studio. Other commercial tools with explicit support for MDSD such as OptimalJ [21] or AndroMDA [3], also provide a promising research scenario.

¹ This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.
The Web engineering community is aware of this trend and several approaches have emerged to provide support for Model-driven Web Applications development. These approaches introduce conceptual models to describe the different aspects that define a Web Application in an abstract way. The use of model-to-model and model-to-text transformations is proposed to obtain code that unambiguously represents the Web Application conceptual mode. Several web engineering methods that follow this approach are: OOHDM [24], WebML [5], WSDM [16], UWE [15] and OOH [10]. All these methods have been widely accepted by the web engineering community and have been proved and validated in several applications. Some methods also provide support tools. These tools define models that capture the structural, navigational and presentation aspects of Web Applications and provide code generation support.

In this context, OO-Method [22] is an automatic code generation method that produces the equivalent software product from a system conceptual specification. Like the methods mentioned above, it has a tool that implements its process: OlivaNOVA [18]. This tool is the commercial application of the OO-Method, which has been developed by Care Technologies S.A.\(^2\) It automatically generates code in multiple platforms (.NET 2.0, Java, Visual Basic) from an OO-Method model. This tool has been tested in real enterprise applications and is therefore a reliable development environment. However, some usability issues regarding aesthetic aspects and navigation were detected when web systems are modelled. For this reason OOWS [7], has been defined as an extension for the OO-Method to improve Web Applications modelling. OOWS extends OO-Method conceptual models by introducing two new ones. The goal of these models is to accurately capture the navigational and presentational aspects of Web Applications.

In this work, we present a MDA development environment that is able to produce Web Applications automatically from Conceptual Models. The data and business layer is generated by OlivaNOVA since its quality has been proven. On the other hand, the OOWS development process that is described in this paper extends OlivaNOVA, providing an accurate Web Interface that interacts with the business logic. In order to comply with Care Technologies business policies, the OOWS development process was implemented without modifications to OlivaNOVA. As a consequence, backwards compatibility with previously generated applications is guaranteed. Therefore, the main contributions of this work are:

1. A MDA development environment that extends an industrial MDA tool (OlivaNOVA) with the conceptual models (OOWS) needed to develop Web Applications.
2. A visual editor based on the Eclipse Platform that: 1) defines OOWS models at the PIM level according to its metamodel and 2) provides a strategy to integrate OO-Method models defined in OlivaNOVA with models defined in OOWS Visual Editor.
3. A Code generation process that integrates business logic and persistence (provided by OlivaNOVA) and a web interface generated by the OOWS Model Compiler.

The rest of the paper is organized as follows: section 2 presents both commercial and academic approaches that develop Web applications from a MDA/MDD point of view. Section 3 briefly introduces an overview of the development process proposed

\(^2\) Care Technologies S.A. www.care-t.com
here. Section 4, describes the conceptual models used at the PIM level and Section 5
details the strategy followed to generate Web applications code. Section 6, discusses
lessons learned and identified weaknesses of the proposal. Section 7 presents our
conclusions.

2 Related Work

Due to the proven benefits of MDA many methods have emerged in the field of web
engineering. Several methods have also been implemented to support their ideas.
Currently, there are many model-driven tools to generate web applications from both
the academic and industrial sectors.

For many academic approaches such as UWE or OOH, UML is the starting point.
UWE defines a development process that is able to semi-automatically generate Web
Applications. To achieve this goal, UWE provides a navigation and presentation
model. Since its metamodel is defined as a conservative extension of UML 1.5, an
important advantage is that the analysts are very familiar with the notation. To create
UWE models an ArgoUWE [11] editor based on an open source tool (ArgoUML) is
provided. The transformation code process generates an XML file for deploying a
Web Application into a XML publishing Framework. However, the transformation
process is currently in a very early development stage, and does not yet generate a
complete Web Application.

A similar approach to UWE is the OOH method and its tool VisualWade [8]. This
tool is based on three views to model Web Applications: a class diagram, a navigation
diagram, and a presentation diagram. An advantage is that web page look and feel
can be designed inside the tool. In addition, the tool has a transformation engine that can
generate PHP applications. An inconvenience is that although the notation is UML-
based, it cannot be exported. Furthermore it also lacks a proper definition of behav-

iour because only CRUD operations can be defined thus requires manual code.

WebRatio [1] is a good example of an industrial tool that comes from an academic
methodology, WebML. Its main advantages is that since it focuses on the entire web
development process it covers requirements gathering, data and hypertext design,
testing, implementation and maintenance. This tool has been tested in several indus-
trial projects and has advanced features to model web presentation aspects. Its data
model is based on UML class diagrams, however some notation from Hypertext
Model is not strictly UML compliant. In spite of the fact that WebML are continu-
ously evolving, the behavioural aspects are not yet fully supported. The behaviour
that can be modelled is mainly focused on create, modify, delete operations and rela-
tionships between objects. More complex behaviour must be imported from external
sources (like web services) or implemented manually.

Another interesting approach is AndroMDA [3], an Open Source MDA generator
for multiple platforms. It supports UML Models and is extensible by means of car-
tridgess. A cartridge provides a metamodel, a set of transformation rules and code
templates to define how UML models are transformed to a specific platform code. It
defines transformations for web platforms as Struts. Even though it uses UML mod-
els, AndroMDA does not provide a UML Case tool. Therefore an external tool is
needed to define UML models in a XMI format that AndroMDA can understand.
Hence AndroMDA can be defined as an aid to MDA development but it cannot be considered as a MDA tool.

Finally a good example of a Model-Driven industrial tool is Optimal J. This tool follows a MDA approach which starts in a domain model (PIM) that is UML compliant. This domain model is transformed into an application model (PSM), from which the final code is generated. Even though OptimalJ [21] is a model driven tool, its PSM is clearly based on the J2EE platform and is related to technological concepts. Therefore its code model emphasizes manual changes in code by means of protected blocks. As a consequence, the entire code cannot be generated from the domain model.

The main difference between these approaches and our MDA approach is that it emphasizes the use of conceptual models for generating both presentation and behaviour aspects. Currently some minor details are still to be fixed directly in the final code. We are analyzing all these manual changes, looking for their corresponding conceptual primitive. Our final goal is to provide a true Model Compiler for Web Application generation.

![Fig. 1. OOWS MDA Development Process](image)

3 MDA Development Process: Overview.

The new development process that this work introduces is summarized in Fig.1. The left side represents the OO-Method development process, which is supported by OlivaNova. The OlivaNova Modeller allows us to graphically define the different views
that describe a system (the structural, dynamic and functional models explained in section 4). A set of OlivaNOVA Transformation Engines, then compiles these views and translates the conceptual primitives defined in the PIM into a specific implementation language. The final result is a three-tier software application: logic, persistence, and presentation. This development process is summarized in Figure 1 (left).

The code generation process implemented by OlivaNOVA must also be extended in order to automatically generate code from OOWS models. In order to perform this extension, we have defined a parallel translation process which generates code from the OOWS models (Figure 1 right). The integration of the two translation processes (OO-Method and OOWS) are performed at the implementation level. To achieve this goal two tools to support Web development have been implemented: the OOWS visual editor and the OOWS model compiler. Therefore, the MDA development process proposed is composed of two stages:

1. Conceptual Modelling: First, the PIM model that defines the specific aspects of our Web application is built. Static and behaviour aspects, which are collected by OO-Method models, are defined in the OlivaNOVA Modeller. Additional information about this tool can be found in [18]. The navigational and the presentation models are defined by the OOWS Visual Editor. This tool and the strategy defined to solve the integration with OlivaNOVA models are described in detail in section 4.

2. Code Generation: In this state, two parallel code-generation processes are carried out. The OlivaNOVA transformation engine produces code based on a three-tier architecture from OO-Method models. The OOWS model compiler generates a Web Interface by means of model-to-text transformations whose target is a new Web Interface. This transformation step, which is described by MDA as an Automatic Transformation, is introduced in section 5.

The MDA development process proposed produces fully functional Web Applications. Each stage of this MDA development environment is detailed in the following sections.

4 Conceptual modelling

The first step in a MDA development process is to define the PIM (platform-independent models). In this work our PIM is based on both the OO-Method and the OOWS metamodels. This section describes how Conceptual Models are created. Since these models are defined in different tools (OlivaNOVA Modeller and OOWS Visual Editor) an integration mechanism to share data between them is introduced.

4.1 Modelling behaviour: OO-Method Models

OO-Method provides a UML-based PIM where the static and dynamic aspects of a system are captured by means of three complementary views, which are defined by the following models:

- a Structural Model that defines the system structure (its classes, operations, and attributes) and relationships between classes (specialization, association, and aggregation) by means of a Class Diagram.
- a Dynamic Model that describes the valid object-life sequences for each class of the system using State Transition Diagrams. Object interactions (communications between objects) are also represented by Sequence diagrams in this model,
- a Functional Model that captures the semantics of state changes to define service effects using a textual formal specification.

These models are created in the OlivaNOVA Modeller. The OO-Method models are created in an XML file that is used as input for the desired transformation engine. Figure 2 illustrates models screenshot’s from the OlivaNOVA Modeller. Extended information about these models and their semantics can be found in [22].

4.2 Extending OO-Method for the Web: OOWS Models

OOWS introduces two new models into the previous OO-Method metamodel in order to support the particular navigational and presentation aspects of a web application in an adequate way. These models are:
- User Model: A User Diagram allows us to specify the types of users that can interact with the system. The types of users are organized in a hierarchical way by means of inheritance relationships which allow us to specify navigation specialization.
- Navigational Model: This model defines the system navigational structure. It describes the navigation allowed for each type of user by means of a Navigational Map. This map is depicted by means of a directed graph whose nodes represent navigational contexts and whose arcs represent navigational links that define the valid navigational paths over the system. Navigational contexts are made up of a set of Abstract Information Units (AIU), which represent the requirement of retrieving a chunk of related information. AIUs are made up of navigational classes, which represent views over the classes defined in the Structural Model. These views are represented graphically as classes that are stereotyped with the «view» keyword and that contain the set of attributes and operations that will be available.
to the user. Basically, an AIU represents a web page of the Web Application at the conceptual level.

- **Presentation Model:** Thanks to this model, we are able to specify the visual properties of the information to be shown. To achieve this goal, a set of presentation patterns is proposed to be applied over our conceptual primitives. Some properties that can be defined with this kind of patterns are information arrangement (register, tabular, master-detail, etc), order (ascendant/descendent), or pagination cardinality.

In order to support OOWS models definition, a visual editor based on the Eclipse Modelling platform has been built. This tool supports two main features:

- **Model definition and persistence:** OOWS models must be created in accordance with their metamodel and be serialized in a standard language. To accomplish these requirements, we have used a set of tools provided by the Eclipse Modelling Framework (EMF) [6]. From an OOWS metamodel definition described in Ecore (a subset language from MOF), EMF provides a framework to define and edit conceptual models according to the OOWS metamodel. Moreover, EMF stores conceptual models as XMI documents, which is the OMG Standard to interchange conceptual models between MDA tools. Thanks to this feature, our tool will be able to interact with other modelling tools in the future.

- **OOWS Visual Editor:** To simplify OOWS models definition for system analysts, a visual editor tool has been developed. This tool is similar to other CASE tools that are on the market. To accomplish this task, the Eclipse Graphical Modelling Framework (GMF) has been used. This framework allows the automatic generation of visual editors from an Ecore metamodel. The editor is created from a specification that associates each conceptual primitive to a textual or graphical representation in our editor.

A screenshot of our visual editor is shown in Figure 3. This figure depicts the OOWS navigational model of IMDB Lite Web application. In this context, the model defines several navigational contexts to show information about movies, actors, direc-

![Fig. 3. OOWS Visual Editor](image.png)
tors, etc. For instance, TopMovies shows a movie ranking created by users whereas MovieInformation provides a detailed description about a selected movie.

4.3 Integration at the PIM level

To define an OOWS Model, an OO-Method class diagram is needed because a navigational context is composed by views over classes. This diagram also defines the interface for calling business logic services. Since the class diagram is created in OlivaNOVA, the OOWS Visual Editor needs a mechanism to retrieve information from it. Therefore, integration at the PIM level takes place at the class diagram synchronization. To perform the integration two mechanisms are needed: (1) an export mechanism to share a class diagram defined in OlivaNOVA (2) an import mechanism to simplify the import process to OOWS Visual Editor.

The export mechanism is currently provided by OlivaNOVA because every model can be stored as standard XML. The import mechanism must be implemented inside OOWS Visual Editor, taking into account that a XML document is going to be imported. Figure 4 illustrates this mechanism:

- Firstly the OO-Method class diagram is defined by means of an Ecore metamodel. Since Ecore is a main component in EMF (see section 3), the OOWS Suite developed in Eclipse can easily understand Ecore models. So that, the current OOWS metamodel is extended with the new Ecore class diagram metamodel. As a result of this, the OOWS Visual Editor can understand and manage information from the OO-Method Class Diagrams defined in OlivaNova.

- Then, a model-to-model transformation is defined to transform an OlivaNOVA class diagram (store as XML) to its Ecore representation (store as XMI). When this transformation is finished, the resulting model can be loaded into the OOWS Visual Editor. To define the transformation we have chosen XSLT because is the standard language to provide transformations between XML documents.

![Diagram](image.png)

**Fig. 4.** Import mechanism from OlivaNOVA Modeller
5 Code generation process

The MDA code generation process proposed in this work is composed of two parallel processes: 1) one to generate business logic from OO-Method models and 2) another one to generate Web interfaces from OOWS and OO-Method models. In this section, firstly the first code transformation process in OlivaNOVA is briefly introduced and the OOWS Model Compiler that supports the second transformation process is described. Finally how is integrated the code from the two processes is explained.

5.1 From OO-Method to Code: OlivaNOVA Transformation Engines

According to MDA, each OlivaNOVA Transformation Engine is a tool that automatically performs PIM-to-PSM transformations and PSM-to-Code transformations. The input for a transformation engine is a model created in the OlivaNOVA Modeller. Each transformation engine is composed of four elements that define its code generation strategy:

1. A common Conceptual Model, which is based on OO-Method that shares all transformation engines. This model can be seen as the OlivaNOVA PIM.
2. An Application Model (PSM) for each target platform (Java, .Net, ASP) that represents its technological aspects. The application model does not need to be modified by analysts because there is a clear relationship between Conceptual Model elements and Application Model elements. For this reason, it is hidden inside the transformation engine.
3. A set of correspondences that defines relationships between Conceptual Model elements and Application Model elements. This set can be described as a function whose domain is the Conceptual Model and whose image is an instance from the Application Model.
4. A set of transformations that establishes how to obtain the application code from an Application model. Each transformation takes an element from the Application model and generates the associated portion of code.
   Following this generation strategy, a transformation engine generates an application that is functionality equivalent to the model.

5.2 From OOWS to Code: OOWS Model Compiler

The OOWS Model Compiler is composed of two main elements: 1) a Web Interface Framework that has been developed to simplify transformations complexity and skip PIM-to-PSM step. This framework provides high level conceptual primitives for building a Web Application. 2) a set of model-to-text transformations for OOWS Models that produces code from conceptual elements.

5.2.1 Web Interface Framework

The main goal the Web Interface Framework is to simplify model-to-code transformations. PHP 5 has been chosen for implementation purposes because it is an accepted and widely used language for building Web frameworks. To define this framework some
principles from the Software Factories approach have been adopted. For instance, high
level primitives that abstract web page implementation have been provided. Thanks to
these high level primitives, the semantic gap between PIM models and code is dra-
stically reduced. Therefore, PIM-to-PSM transformation step is avoided because the re-
lation ship between PIM concepts and framework primitives is very clear. Advantages and
disadvantage of this approach are discussed in [2]. These primitives are defined as a set
of classes which represent common concepts on Web Application development such as:
web page, link, navigation menu, service etc. Following this approach, a Web Applica-
tion can be defined as a set of objects that specifies which kind of functionality should
be provided. The principal objects that make up our Web application are:

• Application: This object, which is unique in an application, contains global infor-
mation. For example, the Role method defines different user types that can access the
application while the AllowAnonymous property provides anonymous access. It’s
possible to select a presentation style through the Default Style method. In addition,
to create the Page objects that make up the application, the AddPage method is
provided.

• Page: Each Page object is related to a Web Page implementation. Following the
usability criteria described in [19], this object defines a web page as an aggregation
of content zones. A zone can be defined as an independent piece of information or
functionality. The most important content zones that can be defined are:
- Navigation Zone. It provides the user with a set of navigable links that can be ac-
tivated within the web page.
- Location Zone. It provides the user information about where the user is and the
navigation path (sequence of navigational pages) that s/he has followed to reach
that location.
- Information Zone. It provides information about the system (usually in a data-
base).
- Services Zone. It provides access to the operations that can be activated. This
zone is contained inside an information zone, and all the operations are related to
that information.
- Data Entry Zone. It is responsible for providing the user with a form to input
data to execute certain operations. Then, a submit-like button links the input data
with the associated functionality..
- Custom Zone. It contains information regarding other types of contents that can-
not be catalogued in the other zones. This zone is normally used for domain-
dependent content, such as advertisements, other related web applications, ex-
ternal applications, etc.

The Page object provides a set of methods to add content zones such as AddNavi-
gationZone or AddInformationZone.

• Zone: a Zone object defines any content zone mentioned above. For example, if we
define an information content zone, this object provides the AddField method to
select the attributes that are going to be recovered and also provides the AddDetail
method to show information recovered through a relationship (association, compo-
isition or inheritance). Mechanisms to filter and index recovered information are
specified with the DefineFilter and the DefineIndex methods.

Since these concepts are not related with OOWS, this framework can be used by
other web engineering methods in their generation processes. Moreover, the frame-
work facilitates the inclusion of aesthetic aspects by means of presentation templates than can be adapted and reused as can be see at [25].

Figure 5 shows a web page from IMDB Lite case study (see section 6) where framework objects’ and its visual representation are related. When a user makes a web request to our application, the Web framework creates the set of objects that make up the requested Web page (Application, Page, Zone etc.). These objects produce XHTML code that is sent to the user Web browser. It’s interesting to note that the framework abstractions are not closely related to OOWS Models. Due to this fact, Web Interface Framework can be used as target the platform by other MDA process or Web engineering methods.

![Image](image_url)

**Fig. 5.** Web Interface Framework code example

### 5.2.2 Model-to-code transformations

To define a transformation from model to code several approaches have been proposed. Many authors suggest graph-based transformations [25], others support template languages [14], and there are even some defenders of XSLT transformations [12]. We have chosen openArchitectureWare (oAW) [20], a model-driven support framework. The main advantage of oAW with regards to other solution is that it provides a good Eclipse support, so that can be easily integrated into the MDA Environment proposed. Among other tools, oAW provides the xPand language to produce code from conceptual models. Using this language, transformation rules whose input is a conceptual primitive can be defined. From this conceptual information, a code template is completed.

In order to define the OOWS code generation process, a set of oAW rules that takes OOWS conceptual primitives as input has been created. Each OOWS element has a rule that produces the Web Interface Framework code that represents its functionality. From an MDA point of view, this is called Automatic Transformation because an in-between PSM is not needed. The following example illustrates this process. The navigational model primitive (described in section 4.2) has an associated rule called *ModelNavigationRule* (see Figure 6). This rule generates the code that adds the user
owner of the navigational model. To achieve this task the Role method from the Application object is used. Then, a new web page is created for each context by calling the Page method.

\begin{verbatim}
<DEFINE NavigationalModelRule FOR NavigationalModels
  $Application->Role|"$UserRol.id","$UserRol.className","$UserRol.parents"|;
<FOREACH Navigational Context AS NC
  $Application->Page|"NC.id","NC.alias","NCRol.id","NCReachability"|;
<EXPAND NavigationalContextRule FOR NC>
<ENDFOREACH>
<ENDDEFINE>
\end{verbatim}

Fig. 6. Navigational Model transformation rules

5.3 Code Integration

A Web interface is implemented from OOWS conceptual models using the framework introduced in section 5.2.1. However, OlivaNova business logic can be generated using a different technological platform and requires a mechanism to integrate business logic and the interface code.

For reasons of brevity, we only focus on the .NET code generated by OlivaNOVA. In this generation process business logic is encapsulated as a COM+ object. The communication between an interface and this component is performed by interchanging messages. To request data or execute a service an XML message is sent to the COM+ object and the response is made with another message. Therefore, the Web interface must communicate with business logic by sending suitable messages. To carry out this communication process, a business facade named OlivaNovaFacade has been developed. This facade presents a group of methods that can be used by the Web Interface Framework to build XML messages to be sent. Figure 7 shows how this integration mechanism works. This solution provides a transparent mechanism for using OlivaNOVA business logic without having to modify previously generated code.

6 Lessons learned.

In this section, we present the experiences extracted from the implementation of two Web applications developed using the MDA development environment presented here. These Web applications are:

- Water control application: This case study is based on a management application from a water supply company, “Aguas del Bullent”\(^3\). Its information system is implemented on a desktop application that manages every functionality needed (customers management, bills, resource assignment etc.). New business goals required migrating some features from the main application to a web environment. This new

\(^3\) http://www.aguasdelbullent.com/index_en.html (only corporate information)
Web application needed to be perfectly integrated with the desktop application in order to be able to share the same data.

- **IMDB Lite**: This case study is based on an online repository of information related to the movie world: the IMDB Internet Movie Database (IMDB). To illustrate our approach (see Figure 5) we implemented a Web application for some functionalities from the IMDB. For instance to see information about movies and actors, etc. and to make comments about them. All the code from this example (except aesthetic properties such as colour, font-family, background etc.) was automatically generated from OlivaNOVA and the OOWS Model Compiler.

### 6.1 Benefits

- Nowadays, to develop a Web application, System Analysts must deal with many technologies and concepts (client-server, web request, etc.) when designing the system. For this reason, Web development is more expensive and slower than traditional software development. A key advantage of this MDA development approach is that it is independent from the technological platform. Since business logic can be generated in several languages, programming expertise is not a problem. Analysts need only to focus on how to design the system and not on how to implement it.

- The transformation process from a PIM-to-PSM and from PSM-to-code is not trivial. The introduction of the Web Interface Framework has two advantages: 1) since a PSM is not required the model-to-model transformation step is eliminated and the process is simplified. 2) Technological aspects related to code generation can be delegate to the framework without introducing them at the PIM level.

- This MDA development environment has been defined using several standards that have been accepted by MDA community. The PIM metamodel is defined using MOF. OOWS/OO-Method models are based on UML, which is a widely used notation, and can be exported as XMI. As a consequence, these models can be imported/exported from/to other MDA tools.
6.2 Identified weaknesses

- We have detected a few usability issues in the generated interfaces. These usability problems are related to complex interactions between user and interface. For example is not possible to specify how errors are shown to the user or model wizards to aid the user. However we are studying on how to incorporate usability patterns on OOWS Models in order to generate interfaces more usable to users.

- Currently our approach is based on two different tools OlivaNOVA Modeller and OOWS Visual Editor. The main reason is to preserve the successful code generation capabilities of OlivaNOVA. However, models definition is a hard task because the analyst has to export/import models from two different environments. Our final objective is to have a unique environment, where the OOWS navigational and presentation models must be included.

- In the case studies presented here all business logic was generated by OlivaNOVA. Even though this MDA approach seems to be limited in use, the same concepts can be used with other business logic providers. Interaction with other business logics that come from external sources such as web services or business process are topics that will be include in future works.

- Non functional-requirements such as security and performance are not taken into account using this approach. Currently, the security implemented by OlivaNOVA is considered to be adequate; in a very restricted environment other security protocols should be analyzed. With regard to performance no strict empirical study has been done. Though business logic has been validated at the industrial level, the produced Web interface has not yet been tested in a high-demand Web Application. We are studying how to carry out a performance test and how to improve it if required.

7 Conclusions

A MDA-driven environment for Web Applications development has been presented in this paper. This environment provides several tools to support the OOWS/OO-Method development process to automatically generate Web Applications. Its main advantage is that it combines experiences from both the academic and the industrial world. On the one hand the PIM level is supported by OOWS and OO-Method Conceptual Models, which describe Web applications systems precisely. On the other hand business logic generation is delegated to OlivaNOVA which is a commercial tool that produce high quality code and has been verified at industry.

The development process for obtaining a Web Interface from OOWS Models is described in detail. This new development process is based on open source tools and OMG standards such as MOF, XMI or UML which are broadly accepted in MDA community. An integration strategy has been defined in order to combine these parallel processes in a satisfactory way. This MDA environment has been applied and verified in two case studies. Several Web Applications are currently being developed with this approach. The experiences and feedback from these works will help us to improve the code generation process.
References


Enriching Hypermedia Application Interfaces

André T. S. Fialho, Daniel Schwabe

attfialho@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 through concrete interfaces by applying rhetorical style sheets, which define actual animation schemes. The resulting applications have different transition animations defined according to 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 multimedia, in the form of sound, animations and video. In particular, animation is increasingly becoming an integral part of Web application interfaces, after the advent of AJAX technologies, as exemplified, for instance, the Yahoo Design Patterns Library (for transitions) [13].

So far, the emphasis has been in animating individual interface widgets, as a way to enhance or emphasize actions taking place during the interaction. A more complex kind of animation is involved when considering entire interface changes that occur during hypermedia navigation. Such interface changes are intrinsic to hypermedia systems, where there is a transition as a result of navigational changes which commonly also changes the information items displayed. As such, this type of interface change is a prime candidate for the application of animation techniques.

The actual use of such techniques has remained an artistic endeavor up to now. Designers use their sensibility and previous experience, sometimes guided by design patterns such as [13]. This paper presents an approach for systematically enriching hypermedia applications by extending the SHDM (Semantic Hypermedia Design Method) approach [9]. 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.

* This is an expanded version of a short paper accepted to ICWE 2007.
Section 3 presents the proposed approach, and Section 4 presents a discussion about the results and conclusions.

2. Background

2.1. Advantages of animation

There have been several studies analyzing the advantages of using animation in interfaces. A study [2] of the use of animation for spatial information, shows how animations affect the mental map formed by the user, it is shown that animation helps to maintain consistency and aids in reconstructing spatial information, without performance loss.

Evaluations of these kinds are difficult to carry out. Gonzales [5] argues that inconsistencies in experiments result stem from the comparison between textual versus graphical display of information, making it difficult to isolate the specific effects of animation. Another factor is due to the tasks used – they are frequently simple, well known and understood tasks that pose no real challenge or difficulty to the user. A third aspect is the focus on information presentation as opposed to interaction with the user.

With these considerations in mind, the authors in [5] carried out an experiment to evaluate the effect of images, transitions and interaction styles on two renditions of an interface – one on a real interface and a second on a mock-up. It was observed that real images improve decision making, turning the interaction into a more pleasurable experience for the user. However, these effects vary greatly depending on the kind of task being carried out. It was also observed that smooth animations are preferred over more abrupt ones, also improving decision making. Parallel interactions are also preferred by users, improving the quality of decisions when compared to sequential interactions.

2.2. Animation in application interfaces

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 user experience through smooth transitions (Media Browser [4], Visual Decision Maker [12], Time Quilt [6], MacOS, Windows Vista, etc). Over the web, animation is also widely applied mostly due to the increased acceptance of technologies such as flash, dynamic html and dynamic interfaces with asynchronous communication (Ajax). These animations are mostly applied through a graphic design process, requiring an artistic view for decision making in the process. As already mentioned, patterns and libraries have been developed to aid the design and implementation process, facilitating the use of animations.
While classification of the animation types can be quite variable, depending on the abstraction level used, we can distinguish animation effects that are applied to elements in the following categories: Entrance; Emphasis; Exit; Motion Path and Object Action.

The first three are straightforward, representing the entrance, emphasis or exit of an element, the motion path repositions an object, moving it along an invisible path, and the object action represents an internal behavior of the object such as the execution of multimedia content.

Beacker and Small [1] divide animation in three classes according to their use. The first, are structure animations that correspond to animations made under three dimensional environments, with the purpose of simulating, previewing and exploring different position views and environments. The second are process animations, applied for the visualization of internal processing of a function, task or application. This type is commonly used for simulating and visualizing the functioning of algorithms and programs. The third class are function animations, which helps the user comprehend the interface, minimizing complexity and guiding through interactions. This kind of animation represents best what we are trying to accomplish. Table 1 below identifies what functions this type of animation can provide, next to the problem it helps to solve.

<table>
<thead>
<tr>
<th>Function</th>
<th>Answers the question</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>What is this?</td>
<td>Identify an application when they are invoked.</td>
</tr>
<tr>
<td>Transition</td>
<td>Where did I come from?</td>
<td>Illustrate changes in system state that change the interface. For example, closing a window might be accompanied by an animation of an outline for the new window as it shrinks until it disappears.</td>
</tr>
<tr>
<td>Choice</td>
<td>What can I do?</td>
<td>Animate menus in order to improve efficiency of display, indicating the relationship between items.</td>
</tr>
<tr>
<td>Demonstration</td>
<td>What can I do with this?</td>
<td>Improve the clarity of description of a function associated to an icon.</td>
</tr>
<tr>
<td>Explanation</td>
<td>How can I do this?</td>
<td>Present steps necessary to achieve a goal as an animated &quot;Guided tour&quot;</td>
</tr>
<tr>
<td>Feedback</td>
<td>What is happening?</td>
<td>Show the degree of completion of a task as progress bars and other indicators.</td>
</tr>
<tr>
<td>History</td>
<td>What have I done?</td>
<td>Replay user activities in order to explain the steps that were taken to arrive at a particular state.</td>
</tr>
<tr>
<td>Guidance</td>
<td>What should I do now?</td>
<td>Animations showing what would be the result of possible actions.</td>
</tr>
</tbody>
</table>

Several animation effects that can be used in interfaces originate from traditional cinematography techniques[1]. Cinematographers learned over time which dynamic
scenes are easily assimilated by the spectator and which aren’t. This type of knowledge is a resource of principles that can be applied to human computer interaction [11].

Another relevant influence comes from traditional cartoon animations techniques [3]. These techniques offer enough information without confusing the spectator or demanding greater efforts for understanding the animation, therefore aiding the interpretation of changes that occur in the interfaces. In these animations, elements don’t simply disappear or change position instantly; certain animation principles apply. Among the most useful, are solidity, reinforcement and exaggeration, as postulated by Johnston and Thomas [7].

Another important factor is timing. Determining the appropriate timing is a non-trivial task, since there is a trade-off between the amount of effects to be presented and the total duration of the animations since the duration of an animation should not affect the time to achieve a task. On the other hand, an animation executed too quickly defeats its purpose, since it becomes too hard for the user to follow. Experiments indicate that a duration that satisfies these constraints should be between half to one second [2].

2.3. Abstract Interfaces in SHDM

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.

The abstract interface model is built by defining the perceptible interface widgets. Interface widgets are defined as aggregations of primitive widgets (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.

It is useful to design interfaces at an abstract level, to achieve, among other things, independence of the implementation environment.

![Diagram](image)

**Fig. 1** Abstract widgets ontology

The most abstract level is called the Abstract Interface, focusing on the type of functionality played by interface elements. The Abstract Interface is specified using the Abstract Widget Ontology, which establishes the vocabulary, shown in Fig. 1.
This ontology can be thought of as a set of classes whose instances will make up a given interface.

An abstract interface widget can be any of the following:

- SimpleActivator, which is capable of reacting to external events, such as mouse clicks;
- ElementExhibitor, which is able to exhibit some type of content, such as text or images;
- VariableCapturer, which is able to receive (capture) the value of one or more variables. This includes input text fields, selection widgets such as pull-down menus and checkboxes, etc... It generalizes two distinct (sub) concepts;
- IndefiniteVariable, which allows entering hitherto unknown values, such as a text string typed by the user;
- PredefinedVariable, which abstracts widgets that allow the selection of a subset from a set of pre-defined values; oftentimes this selection must be a singleton. Specializations of this concept are ContinuousGroup, DiscreetGroup, MultipleChoices, and SingleChoice. The first allows selecting a single value from an infinite range of values; the second is analogous, but for a finite set; the remainder are self-evident;
- CompositeInterfaceElement, which is a composition of any of the above.

It can be seen that this ontology captures the essential roles that interface elements play with respect to the interaction – either they exhibit information, or they react to external events, or they accept information. As customary, composite elements allow building more complex interfaces out of simpler building blocks.

The software designer, who understands the application logic and the kinds of information exchanges that must be supported to carry out the operations, should design the abstract interface. This software designer does not have to worry about several usability issues related to the look and feel, which will be dealt with during the concrete interface design, normally done by a graphics (or "experience") designer.

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. Concrete widgets correspond to those usually available in most runtime environment, such as labels, text boxes, combo boxes, pull down menus, radio buttons, etc... Fig. 2 and Fig. 3 show examples of concrete interfaces, the first for a page describing a movie and the second describing an artist. In sequence, Fig. 4 and Fig. 5 show abstract representations for these interfaces.

As represented below, a concrete interface is based on an abstract interface model. During development of the application we construct the concrete interface by mapping each abstract widget into a real widget. In this example each element can be mapped to a text or image. The dispositions of the elements are presented according to the designer’s choice and the layout details are added later, or bound to the widgets as CSS styles.
**Fig. 2** Interface that represents a page of a movie

**Fig. 3** Interface that represents a page of an actor
Fig. 4 Abstract interface of a page that describes movies

Fig. 5 Abstract interface of a page that describes actors
3. Introducing Animations in Hypermedia Applications

In this section we cover the systematic process for the insertion of animations during the design of interface of the application.

The process is composed of four stages illustrated in Fig. 6, in which each of the stages produces a specific output necessary to the next stage.

![Fig. 6 Steps to produce an animated interface](image)

The animations proposed in this work are displayed to the user during interactions that define a change in the navigational state. Each of these changes is made between a pair of distinct origin/destination interfaces that represent these states. We call this process a transition, which can be seen as an intermediate animated interface between two other interfaces. Note that 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 transformations that changes the source interface into the destination interface. Therefore, the transformation to be applied to each widget has to be specified. Fig. 7 illustrates this process.

![Fig. 7 Transition representation](image)
3.1. Interface Pairs

As already stated, the first requirement for setting up an interface animation is the identification of the widgets that compose each interface. After describing the abstract interface we will then need to specify which pairs of interfaces will define the transition, which is determined by the navigational structure of the application and associated abstract interfaces, as specified in the SHDM model of the application.

Each pair will be composed by a source and a destination interface. The source identifies the originating state of the transition and the destination the ending interface displayed after the transition.

As an example we can consider the interfaces described earlier in the document by Fig. 4 and Fig. 5 as a source/destination pair of interfaces. The running example will represent a transition from the source Movie interface to an Actor destination interface.

3.2. Transition Specification

Once we have a list of which elements are available in the source and destination interfaces, we can identify the existing relation between them and define which animations may be specified for the transition.

For each defined transition we need to compare the existing widgets in each interface description, considering their respective mappings to the application model. The goal is to identify widgets that are mapped to the same element in the model, or to related elements. As a result, we identify which widgets remain, which disappear, which appear, and define which 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 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 determine the transition specification for the navigational change. Note that each possible change will be triggered during interaction by a specific widget, which is as an activator element, so it is safe to assume that the specifications can be made for every activator element of each interface.

The transition specification is made considering a pre-defined range of animation functions. Three main actions can be applied to a widget: A removal, an insertion or a transformation. After analyzing and experimenting with which animations identify the various alternatives, we came up with the list of animation actions below:

- **Insert** – An insertion animation in which a new widget is added during the transition;
- **Remove** – A remove animation that removes an existing widget from the interface during the transition;
- **Match** – For widgets that remain 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 is responsible for matching these parameters. We can identify which widgets correspond by insuring that the widgets are mapped to a same element instance and attribute in the model.

- **Trade** – A transformation animation responsible for exposing the relation between two distinct related widgets (i.e. widgets mapped to different attributes of a same element instance, therefore represented differently) during the transition, as for example, a morphing effect.
- **Emphasize** – A transformation animation that alters specific parameters such as size or color of a widget to emphasize it.

To exemplify how the animations actions are chosen we will initially consider a transition formed by the source/destination pair of interfaces represented by Fig. 4 and Fig. 5.

Comparing the two interfaces we notice that there are certain widgets that will appear only in the Actor interface (destination) such as the ActorDescription widget - in this case we can apply an insertion action. Similarly, there are widgets that appear only in the Movie interface (source) such as the DirectorComposite, on which we should apply a remove action. For the match action we consider widgets that are mapped to the same attribute and element of the model in both interfaces, such as the ActorName and IdxActorName. The emphasis transition is an auxiliary animation usually applied to give feedback to the user. This animation could be assigned to an activator widget triggered by the user, such as the ActorName on the source interface, which activates the transition from Movie to Actor.

To exemplify the trade action we will consider a transition between Movie interfaces (illustrated in Fig. 4), in which we navigate between objects in the same context (e.g., next Movie) changing only the element of the model presented, but maintaining the interface layout. A trade animation can be represented in this transition, exposing the relation between distinct widgets such as two CharacterPhoto widgets when they represent the same element of the model (i.e. different characters of a same actor element).

These functions will also have properties that describe the duration, the effect to apply (fade, push, grow, etc) and the order in which they should occur within the transition. The specific properties are specified according to their role in the transition, described next.

### 3.3. 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. This sequence in which the animations are presented has great importance since it influences how the transition will be interpreted by the user. Not only the sequence is important, but also the duration of each of these actions and the chosen effects during the actions.

Since several transformations can occur in a transition, an improper sequence, timing or choice of effects could surely confuse the user, defeating the purpose of any kind of cognitive aid during interaction.
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) [10], as it has been used for generating animation plans ([8]). 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 is occurring.

The rhetorical structure is specified in terms of rhetorical categories, which classify the various possible animation structures. A possible set of rhetorical categories is:

- Removal – Set of all animations that achieve an element removal (widgets that disappear). 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 to the user that the request made has been received, and that the application is about to respond to it;
- Layout animations – Set of animations that changes (inserts or transforms) 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 decide which animation functions that have been identified in the previous step fit into each of these categories. 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 to 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. This is an important step in the process, as it defines the sequence of animations during the transition. This sequence can be built in several distinct ways and is one of the main factors that affect the way the user perceives the transition. Different sequences can be arranged for each type of navigation. In our example, illustrated in Fig. 8, we have used one possible sequence using these rhetorical categories.

![Timeline](image)

**Fig. 8** Rhetorical animation structure
This sequence follows the rationale that first the screen should be cleared of elements that will disappear, and simultaneously a feedback of the activated widget should be given. 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 elements.

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. This is necessary since each category has a different role in the rhetoric animation structure and must be presented differently to express the desired semantics to the user. For instance, since the main animation category behaves as the most important part of the transition, we should present the composing animations in a way that most gathers attention of the user, such as exaggerated movements or presenting the animations that compose this category one at a time, avoiding overlaps.

The specification is done through a set of styles defined as a Rhetorical style sheet. The choices of which effect to apply in each action, the duration and what will be the sequence of execution inside each category can be done in several different ways, and is a result of the designer preferences.

This process can be guided by the use of specific patterns that gather solutions to common transition problems within a specific context. These patterns can help the designer identify how the actions should be defined to attract a correct degree of attention of the user. For instance we can use the following set of general rules described in the Yahoo Pattern Design Library [13]:

- The more rapid the change the more important the event.
- Rapid movement is seen as more important than rapid color change.
- Movement toward the user is seen as more important than movement away from the user.
- Very slow change can be processed without disrupting the user's attention.
- Movement can be used to communicate where an object's new home is. By seeing the object moving from one place to another we naturally understand where it went and therefore will be able to locate the object in the future.
- Transitions should normally be reflexive. If an object moved and collapsed to a new spot, you should be able to open it and see it open up with the reverse transition. If you delete an object and it fades, then if you create an object it should fade into place. This reinforces the concept of 'symmetry of action' (the opposite action is implied by the initial action.)

Nevertheless, it is still necessary to experiment with different rhetoric style sheets and examine which would better represent the transition.
3.4. 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. This process is technology dependent and can be done for any kind of hypermedia representation. In this work we use an environment for supporting animation on web documents, in which HTML web pages represent the different types of interfaces. In our example, the environment was implemented with javascript technology using dynamic HTML for the animations.

The process of animation will initiate once a user has triggered a transition by clicking on a widget on the source interface. The activating widget has a transition specification bound to it, redirecting the user to a destination interface. In this case the distinct interfaces are also distinct web pages. Once the element is clicked a request to the destination page and transition specification are made, providing access to all the necessary widgets and actions. When the elements are available the transition specification, which is formed by a list of animation function calls, is interpreted, executing each animation as specified. In our case the functions are described by a javascript library and rendered by the browser.

In our environment the animations are all made on the source interface. This approach is feasible because the appearance of a new destination interface can be considered as a set of transformations over the source that results in a final interface. In other words, when the execution of all animations in the specification is completed the result is the destination interface of the transition. However, since we are dealing with HTML documents, in this approach the final interface would be displayed in the source url, so we redirect the user to the destination interface url, at the end of the transition. This is a necessary step, since the user can then reference the state of the navigation via its url.

The diagram that represents our environment is illustrated below (Fig. 9). In the diagram first the user interacts with an activating widget in the source interface, then the destination interface, followed by the specification, are loaded and executed. Finally the user is redirected to the destination interface.
We have developed a prototype of a flash application of a hypermedia movie database with the approach described on this document to validate and exemplify the use of animation during interactions, where the user can step through each phase in the rhetorical structure, to better understand what is happening (hence the use of flash instead of the implemented javascript). This example can be accessed at http://www.inf.puc-rio.br/~atfialho/hmdb/hmdb.html (requires a flash plugin 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.

Another goal is to allow partial animations in web pages, supporting AJAX style of interactions.

Acknowledgement: Daniel Schwabe was partially supported by a grant from CNPq.

5. References

5. Gonzalez, C., Does animation in user interfaces improve decision making?, Proceedings of the SIGCHI conference on Human factors in computing systems: common ground, p.27-34, April 13-18, 1996, Vancouver, British Columbia, Canada
Abstract. This paper presents the WebTaskModel approach, by which task model concepts are adapted for the purpose of modelling interactive web applications. The main difference to existing task models is the introduction and build-time usage of a generic task lifecycle. Hereby the description of exceptions and error cases of task performance is on the one hand appended to the task while, on the other hand, being clearly separated. Furthermore, it is shown how an initial navigation model is derived from a WebTaskModel, whereby also the objects attached to a task description are considered.

Keywords: usage-centred design, task model, navigation model, model-driven development

1 Introduction

Both Web Engineering (WE) and Human-Computer-Interaction (HCI) follow a user-centred modelling approach, even though, with different emphasis on various aspects. In the field of WE, model-based approaches, e.g., WebML [5], OOHDM [18], and WSDM [15] have their origin in the development of hypermedia as well as information systems, incorporating more and more methods and techniques known from Software Engineering (SE), or, as in the case of UWE [8], being strongly related to SE. In general, the WE development process starts with requirement analysis. By this the objective of the web site, the intended user groups and their tasks are identified. This information is described, e.g., by means of narrative scenarios, use cases and a kind of task model.

However, task models as applied in the HCI field [7, 9, 11, 12, 14, 17] provide richer concepts, e.g., describing constraints and temporal relations defining conditions and sequencing of task execution. Task models are used as formal input to the subsequent development steps, such as user interface modelling. In contrast to this, in web modelling task models are used primarily as a means to explore requirements and are limited to high-level task descriptions only. The documents are taken mainly as input to derive the conceptual domain model and in some cases the (data-centred) navigation model. Hence, the focus is shifted away from the users and their tasks to the objects to be provided to them. The user-centred view is not kept up during subsequent activity to the degree as in HCI.
All in all, WE and HCI include task and domain modelling – but with different importance and impact on subsequent modelling steps. Also the question by which concept the modelling process should start is answered unequally. Up-to date approaches have to support both starting with a conceptual model of the domain as well as with a task model. Developing state-of-the-art web applications includes designing access to an information space as well as to a function space – both are needed in different combinations, not only within the whole web site but also within single pages. For example, when a customer visits an online book store information about books and relations between them may be in the foreground. Once the customer wants to check out, the activities to be performed are dominating. From the users’ point of view, however, the distinction in accessing the information space or the function space is of no interest. They simply want to reach their goals easily – which has to be accomplished by an appropriate design.

In our point of view the development process should start with both task model and conceptual model of the domain. Models taken for the high-level description of web applications should provide concepts so that designers can choose and alternate between behavioural and data centric modelling as well as combining them. In the following we present how this can be supported by combining task and object models as known from HCI with conceptual modelling known from WE. Throughout the paper we focus mainly on task modelling aspects and its adaptation for the purpose of web processes. In section 2 we first clarify different perspectives taken during modelling of the processes to point out where our approach fits in. Task model concepts cannot be applied straightforwardly due to the differences between traditional user interfaces and web applications. In section 3 and 4 we present WebTaskModel, an adapted web task model. We also propose a combined specification of simplified and extensively specified objects and their relation to task models. Afterwards, in section 5, it is shown how that model can be transformed into an initial navigation model.

## 2 Modelling of Web Processes

The importance of processes in web applications is increasing. The inclusion of their specification in existing modelling approaches leads to the adoption and adaptation of different models, whereby business processes, workflow or user task models are most commonly utilized. In principle they provide similar concepts, but usage in existing approaches differs. The differences are often small, being one reason leading to misunderstandings – not only between WE and HCI but also within each community. The consideration of the development perspective that is taken within modelling supports a first clarification. Web applications, as considered here, are characterized by three kinds of processes:

*Usage-oriented processes* represent from the perspective of the users how they perform tasks and activities by means of the web application. *Domain-oriented processes* result from the domain and the purpose of the web application. Examples of such processes are business processes in e-commerce
or didactical design in e-learning. The process specification reflects the view point of the site owner and his goals. System-based processes are specified from the perspective of the developer aiming at implementation. The description is conditioned by, e.g., business logic and system internal control information. This group of processes also includes the models of web services which are specified by business processes as well. The processes investigated and specified by these perspectives are highly interlinked, being partially orthogonal while in other parts building intersections. Both WE and HCI provide answers of how to model such web processes but with different emphasis of the usage perspective and different utilization of the resulting specifications in subsequent design steps.

In OO-H and UWE [8] business processes are described at conceptual level (by means of UML activity diagrams this time) and subsequently transformed into the design model. In OO-H the process model is mapped onto the navigation model; thus the processes are expressed at this level in terms of the (already defined) concepts. In contrast to this, within OOHDM [16] and UWE the navigation model is expanded by new node and link types representing processes and interconnections with the traditional data-centric navigation model. The navigation process model describes a decomposition structure that is comparable to task hierarchies. The links to the parts of the data-centric navigation nodes showing points where the user may access such a process and thus control is passed over.

While in UWE system operations are already introduced by object methods, in OOHDM the conceptual model comprises process objects with temporary lifetime for specifying business processes and hierarchical decomposition. WebML [4] introduces new symbols in their notation for describing workflows, which show flow of data objects and their manipulation by system operations.

WSDM [6] has a strong emphasis on the user-centred view. For describing user tasks the CTT notation [9] is used in a slightly modified version. The modifications allow specification of transactions as a grouping mechanism for tasks. Similar to the work by [10], task models are used as design models and refined up to the level of dialog models. Afterwards, the initial structure of the navigation model is adopted from the task model structure. i.e., the task model is translated into the notation of the navigation model.

In [19] the CTT task notation is applied as well but more strictly than by WSDM in regarding task types. As a result, in [19] the leaf tasks describe the dialog (by alternating user and system tasks). These are transformed into the navigation model that is described in a states chart based notation. States represent pages containing graphic or executable objects. However, the objects are not modelled within the states but the inner control structure according to the state’s function. In contrast to this, in WSDM the behaviour is described by means of object inspection and modifications as relevant in the task context.

As a rule of thumb, task models concern mainly usage-oriented processes, whereas business process models and workflows are more used to cover the domain- and system-oriented perspective. Generally, process/workflow models focus more on responsibilities and task assignment, whereas the concept tasks relate more to user goals. Control structures used in process/workflow models are basically adopted from programming languages, whereas constructors in task models are more influenced by
the domain and the users. The prevalent focus in modelling differs as well. Task models put the decomposition structure into the foreground which is typically denoted by means of a hierarchical tree notation. Process models lay the primary focus on the sequencing information, formulated in most cases in terms of UML activity diagrams.

The work presented in the following sections focuses on the description of user-oriented processes at an abstract level. It is based on task modelling as well but with extensions aiming at web applications.

3 Modelling High-Level Usage Behaviour with WTM

The WebTaskModel (WTM) presented here enhances our previous work on task based modelling [3]. We extend the modelling concepts to account more appropriately for characteristics of interactive web applications. In contrast to other approaches of task modelling, we assume the developer not to describe the complete application by means of a connected task model; instead task modelling can be applied whenever usage-centred aspects are under investigation. In the case information (objects and their semantic relations) is dominating the modelling focus the well-known models and notations (such as UML and Entity-Relationship diagrams) can be applied. The resulting specification consists of several task models, interrelated and linked to data-centric model parts. Since from this a first navigation structure is derived, neither the task nor the content structure dominates the entire web user interface but only those parts where appropriate.

3.1 Basic Task Description

As an example of task modelling, figure 1 shows parts of the model of an online travel agency. As in general, the task hierarchy, showing decomposition of a task into its subtasks, and different task types are modelled. In the specification of high-level usage behaviour we distinguish cooperation tasks (represented by □) to denote pieces of work that are performed by the user in conjunction with the web application; user tasks (□) that denote the user parts of the cooperation and are thus performed without system intervention; system tasks (≡) to define pure system parts. Abstract tasks (□), similarly to [9] are compound tasks the subtask of which belong to different task categories. This is very similar to the CTT notation, but since it is not alike in every particular a slightly different notation is used here.

Figure 1 depicts three separate task models specifying the login/logout procedure, the booking of a flight and a hotel, and the single-task model get tourist information. We define no dependency between these models to allow a user to switch between the tasks, e.g., to perform the login process at every point within the booking process. At this modelling stage, all isolated task models are conceptually related by the web application (here Flight Application). The position in the final site and thus inclusion of the related interaction elements into pages depends on the navigation and page design.

The number of task executions is constrained by cardinalities of the form (min, max), whereby no label indicates mandatory performance, i.e, card=(1,1). The task perform login process is marked with (1..*) to denote that the user can repeat it
as often as he wants. Labels indicating min=0 define optional tasks (in the example alter shipping data and alter payment data). Additionally, the label \( T \) is used to define transactional tasks, i.e., task sequences that have to be performed completely successfully, or not at all (payment in the example).

**Fig. 1. Examples of Task Models**

The order of task execution is given by temporal relations. In contrast to CTTs [9], we assign them not individually between sibling tasks but to the parent task, so that the same temporal relation is valid for all of the subtasks. Relations typically used in task models are sequential task execution, parallel task execution, and selection of one task from a set of alternative tasks. Further relations are described in [9] and [3]. In the notation used in figure 1, temporal relations are denoted by abbreviations. The tasks find a flight, choose a hotel and payment have to be performed strictly one after the other (denoted by \( \text{Seq} \)) in the specified order (denoted by \( \rightarrow \)).

Tasks of an arbitrary sequence, such as provide departure and provide arrival or alter shipping data and alter payment data, are performed one after the other in any arbitrary order (denoted by \( \text{ASeq} \)), so that at one point in time only one of the tasks is under execution. \( \text{SeqB} \) is an extension we made to describe task ordering that often exists in web applications: going “back” systematically to an already executed task of a sequence. Hereby, the results of that task or of the complete sequence up from that task are rolled back and the tasks can be performed again. In the example, the user is guided through the tasks of payment. Before he accepts the conditions or confirms he is allowed to go back to re-perform alter data and accept conditions, respectively. Since validate data is a system task, the user cannot step back to it, but it is performed automatically after each execution of alter data. Guided tours as traditionally implemented in web sites provide similar behaviour but with different effect. Visitors
are guided through an information space enabling them to leave the tour at any arbitrary point without any effect on the information space or domain model.

3.2 Object Views and Domain Objects

We make use of simplified object models as well as of detailed models describing conceptual domain objects. The descriptions are detailed as needed during modelling of abstract behaviour, refined and completed in subsequent development steps. A simplified object model is used particularly for describing task objects. Such objects represent information units as observed and used by the user (i.e., by the different roles) during task performance. They are not considered as irredundant information sources, but rather as views on the information and function space. In the example, the customer should be able to choose a hotel depending on the selected flight. Thus a list of all hotels located at the place of arrival should be made available to the user (see figure 2). The connections between tasks and task objects are denoted by involves relations, which may be refined by specifications of object manipulations (s. below). Additional information, such as constraints or derivation from the domain objects, is attached informally. In the example, in the underlying database we would store each hotel by a single object, based from which the hotel list can be dynamically derived and inserted into a web page.

Properties of task objects and task object types, respectively, are described by means of attributes while their life cycles are specified by means of a finite state machine. Hereby, only those aspects are modelled that are relevant to the user while performing a task and interacting with the system, respectively.

![Diagram](image)

**Fig. 2.** Examples of Task Objects

A user may want to get some information about the city or region he is going to fly to. Let us assume the site owner wants to provide short descriptions in combination with related books the user should be able to buy through the web site as well. In addition, they should be able to get further information about books and authors. Therefore, we have been inserting the task get tourist information in figure 1. Since the actions the user may perform in this context are basically content-driven, we decide to describe this part by means of views and the conceptual domain model. We apply UML diagrams by which the model can be described in a very detailed way. Figure 3 outlines only extracts from the domain and view model as related to get tourist information. As described by the view model, the city description will be presented to the user. By means of an index list (denoted by the index symbol \[\square\] we adopt from web modelling) the user can select a book to get detailed information.
about its content and its authors. The derivation of the view from the domain model may be described informally or by SQL like statements (not shown here).

![Diagram of Domain and View Model]

**Fig. 3.** Parts of the Domain and View Model

### 3.3 Conditions

Task execution may depend also on business rules, context information and results of tasks performed so far. These dependencies are specified by pre- and post-conditions. A pre-condition is a logical expression that has to hold true before starting the task; once it is started the condition is no longer checked. A post-condition is a logical expression which has to be satisfied for completing the task. In contrast to pre- and post-conditions, temporal relations decide on ordering of task execution. Once a task could be performed because of the sequencing information, the conditions can be evaluated to determine if an execution is actually enabled and may be finalized, respectively.

Condition specifications can be given differently, e.g., in terms of logical expressions. In the following a condition refers to the result of its evaluation (that basically yields “true” or “false”). In the WebTaskModel approach structuring and composition of conditions are separated from their usage in the context of tasks. A condition may be relevant for more than one task, possibly assigned to a task as a pre-condition while being a post-condition for another one. Since conditions are formulated separately from tasks and objects they can be attached flexibly to tasks.

### 4 Task State Information

#### 4.1 Generic Task Lifecycle

Tasks undergo different states while they are performed. The states are also significant to users since in their planning of follow-up activities they take into account current task situations. It is important to a user, whether he can start to work
on a task (initiated) or not because of unfulfilled conditions, or if he is already performing a task and thus its subtasks (running). Further task states and the possible transitions between them are given in figure 5. Start, End, Skip, Restart, Suspend, Resume and Abort can be used to represent events resulting from user interactions or internal system events. The coordination of the usage-oriented processes with the domain-oriented and system-based processes is realized by means of these events. Start, End and Skip are particularly significant in designing the web user interface since they signalize the need of interaction elements.

The global events timeout, navigate_out, navigate_in are generated from “outside” and valid for all states but the end states (skipped, completed and terminated). The timeout event is a pure system event introduced here to deal with the occurrence of session timeouts. In contrast to user interfaces of traditional applications the Client/Browser provides additional navigation interactions (e.g., Back-button, navigation history). The WebTaskModel provides the events navigate_in and navigate_out to deal explicitly with unexpected user navigations by which he leaves or steps into a predefined ordering of task execution. Such user behaviour as well as session timeouts have to be considered at the level of task modelling since they may significantly impact the predefined processes. First of all the relevance of a global event for a specific task is to be decided: Should something happen at all or should a global event cause no special behaviour of the “task”. If it is relevant, the impact on further task executions and on the results reached so far is to be fixed: Should a task be aborted, be interrupted or should occur nothing? Should modifications on objects remain or is a rollback to be performed? Reaction in each case is in general a matter of application design (examples are given below).

Fig. 5. Generic Task State Machine

The generic task life cycle originally served as part of the control component of the final run time system [2, 1] and within our model simulation tool. It was planned not to be visible explicitly in a task model editor. The developer should be able to attach specifications of error and special cases to it by using a terminology which is related to the cases and not to states and transitions. What do we mean by this?

One extension of our web-task model is given by the explicit specification of interruptions from the users’ point of view. Here we distinguish three phases: The

<table>
<thead>
<tr>
<th>State</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>initiated</td>
<td>if all preconditions are fulfilled the task can be started</td>
</tr>
<tr>
<td>skipped</td>
<td>the task is omitted</td>
</tr>
<tr>
<td>running</td>
<td>denotes the actual performance of the task and of its subtasks, if applicable</td>
</tr>
<tr>
<td>completed</td>
<td>marks a successful task execution</td>
</tr>
<tr>
<td>suspended</td>
<td>the task is interrupted</td>
</tr>
<tr>
<td>terminated</td>
<td>indicates an abort</td>
</tr>
</tbody>
</table>
**prologue** description contains the information presented to the user and the required behaviour when the task is going to be suspended. Similarly, the **epilogue** (phase of resuming a task) description shows the information to be presented to the user and required behaviour to continue. The phase of the interruption is called **within interruption**. Referring to the task life cycle, prologue specifications are assigned to the **Suspend** transition, **within interruption** specifications are assigned to **in_state** and epilogue specifications are assigned to **Resume** transition. The assignments were planned to be performed internally by the editor software while the developer models in terms of prologue and epilogue.

We applied our extended task model in small projects (in industry as well as in students’ projects) before implementing an editor. The experiences show that the models are more structured and concise in the cases the developers could make use of the task state machine directly. Although we do not regard this as a representative evaluation, it motivates us to re-design our first editor conception. As a result, the main task structure is modelled as usually by means of a hierarchical tree notation while additional behaviour can be assigned explicitly to states and transitions.

The actions of a behaviour may affect tasks, objects and/or conditions. An action affects
- a task by sending a global or specific task event to it (task-action),
- an object by sending an event to the object (by which involves relations are described in more detail) (object-action),
- a condition by setting its value (condition-action).

The actions are triggered
- either by a global task event or a specific task event (event-trigger),
- or by entering or leaving a state (on-entry, on-exit) or while the task is in the state (state-trigger).

As an abbreviation we use here the notation: task.task-state.task-event → action where task-event is either an event-trigger or a state-trigger, and action is a task-action, an object-action or a condition-action.

### 4.2 Example: Specific Behaviour Specification

Figure 6 shows the behaviours defined so far for the task **select flight**.

![Image](image_url)

**Fig. 6. Task Behaviour Specification**
navigate_out occurring while the task is running is treated in this example as an interruption, which is formulated by

\[ \text{select flight.running.navigate_out} \Rightarrow \text{send Suspend to task select flight} \]

Further behaviours are:

- \[ \text{select flight.running.navigate_out} \Rightarrow \text{send store to object myFlight} \]
- \[ \text{select flight.suspended.Resume} \Rightarrow \text{send restore to object myFlight} \]
- \[ \text{select flight.suspended.Resume} \Rightarrow \text{send flight_selection_incomplete to object message} \]

The specification does not describe from what user interaction the navigate_out results. For example, it may be generated because the user starts to browse through the tourist information:

\[ \text{get tourist information.running.on_entry} \Rightarrow \text{send navigate_out to task select flight} \]

In general, the specification of how to handle an event is uncoupled from its occurrence. The reactions are described locally in the context of each task. A navigate out, however, cannot be detected in all cases (due to the HTTP protocol). The user may navigate to an external web site leaving the current task process open. At a predefined point in time the web application server will terminate the user session, whereby the timeout event is generated. We could make use of this event to formulate the same behaviour as defined for navigate_out:

\[ \text{select flight.running.timeout} \Rightarrow \text{send Suspend to task select flight} \]

However, if the user is not logged in we do not know how to assign to him the data collected so far. So we model a system task handle_timeout that differentiates the cases:

\[ \text{select flight.running.timeout} \Rightarrow \text{send Start to task handle_timeout} \]

All in all, there are diverse causes and different ways of detecting navigations beyond task sequencing. As shown by these few examples, the task life cycle model can be used in a flexible way to describe complex behaviour of high level tasks. The events timeout, navigate_out and navigate_in are used only if they impact high-level behaviour. If, for example, two tasks are represented and accessible, respectively, by the same page, it is rather useless to attach reactions to the navigate_out and navigate_in events. Furthermore, it is oversized to have a less complex task controlled by the generic life cycle. Our experience so far shows, that in particular navigate_out specifications are not very often defined at the task abstraction layer, but if so they are effective in keeping the web application behaviour consistent over all web pages presenting the same task.

### 5 Initial Navigation Model

The Navigation model, as specified in the field of WE, shows the information elements the navigation space is made of and possible ways of how a visitor may access these elements. Typically it is expressed in terms of nodes and links. The objective is to define the content and logical structure of the pages, as well as accessing criteria and kinds of navigation.
In HCI focus is on models describing access to system functionality. In our work we make use of dialog modelling, which we adapted for the concerns of web user interface development [1]. The task model and the dialog model are two separate models at different levels of abstractions. They largely cover the same information: the organization of the user’s interaction with the application. While the task model concentrates on the tasks/activities that need to be performed, the dialog model focuses on how these activities are organized chronologically in the user interface. Following the tradition of model-based design in the field of HCI, we base the dialog description on the task model. An initial navigation model is used as an intermediate step in this process providing the transition from the view on task decomposition to the view on chronological task flow. In the remaining of this section we present how the navigation model can be derived from the task and the object models leading to an integrated WE-HCI-navigation model.

5.1 Task Access Units

The navigation meta-model is extended by the concept of a task access unit. Leaf tasks of the task hierarchy basically denote the places interactions have to be designed for. Therefore, a unit is derived for each leaf task. Figure 7 depicts the basic symbol of a task unit and the possible navigation transitions (task links) resulting from the task states completed and skipped. These links are inserted systematically into the navigation model depending on the cardinality constraints:

<table>
<thead>
<tr>
<th>cardinality</th>
<th>(i) completed transition</th>
<th>(ii) iterative completed transition</th>
<th>(iii) skip transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(0,1)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>(1,<em>), (0,</em>)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(k,k), 1 &lt; k</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>(i,k), i &lt; k</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ transition to be inserted - transition not to be inserted

An iterative completed transition represents the case that the task is executed successfully but can be iterated. Completed and skip transitions end at the symbol of the subsequent task. Determination of this “next task” is based on the task structure and the temporal relations. In the case of a sequence this is quite simple: The task units are connected by links according to the given order. Figure 8 presents the initial navigation model for the task models shown in figure 1. The C-link from the login unit to the logout unit results from the Seq relation. Since the entire process can be repeated, the outgoing completed transition of logout ends at login. The link perform login process.S results from the unbounded repetition number of the superior task. All in all, this little navigation sequence holds the same information as the task model: The user can login and afterwards logout. He can repeat this sequence as often as he
likes or omit it. For a SeqB relation (as defined for payment) the back-links are additionally inserted. The task book flight and hotel consists also of subtask to be performed in a sequence. This time the transformation from the decomposition structure into the chronological structure is a little bit more complex since different temporal relations have to be converted.

Some temporal relations would result in a complex link structure if we want to denote all possibilities by means of simple task units and links. As a kind of abbreviation we define new access structure units. Two of them are used in the example navigation model. \( \Rightarrow \) represents the arbitrary sequence. The dotted arrows link all task units to it which can be visited under that temporal relation. Both the C-links of provide departure and of provide arrival point back to the symbol, denoting that after visiting and completing one of them the next one can be performed. After completion of the entire sequence the user is directed to the next task, which is represented by the C-link from the \( \Rightarrow \) unit to the choose a flight unit. \( \Rightarrow \) represents the possibility to switch between different task groups. In the example it is used to connect the task navigation structures resulting from the given task models. The task access units are derived systematically from the WebTaskModel similarly as simple task units. All in all, the respective task symbols can be interrelated by C-, S- and T-transitions, as well as by trigger-transitions resulting from task triggers. Instead of going into the details of this, we shift the focus to the attachment of objects.

![Initial Navigation Model](image)

**Fig. 8. Initial Navigation Model**

### 5.2 Object Access Units

The view model in figure 3 depicts a part of an object-driven navigation model as used by our approach. Similarly to navigation classes in UWE or object chunks in
WSDM, and like in OOHDM we describe nodes as views on the domain model. These nodes represent objects and content, respectively, to be included in the web pages of the site. The example in figure 8 shows how the task flow description can be linked easily to the specification of the data-driven navigation. Such links are derived from the involves-links, i.e., from the object-actions that are specified in behaviours attached to task states and transitions. Hereby, task access units are linked to access units derived from the object model. The other way round, tasks can be linked to object units to denote tasks the user may perform in the context of the object (as shown by the example of buy book). Starting with simple or informal object descriptions and replacing them by sophisticated view specifications, the involve relation is refined by mapping method calls to the object-actions.

In addition, we adopt the access structure elements as defined by web modelling approaches, e.g., to denote an index, a guided tour and access by means of a search. Access structures, which are based on data structures stemming from the domain model, represent a useful abstraction from the specification of single units and links. Task access structure units as introduced above do the same based on task performance structure. All in all, task-centred and object-centred access can be described by similar concepts, but with slightly different meanings. For example, an index access structure implicitly defines required interactions. In the example the user has to select a book from a list to access more information by means of book details (see figure 9).

Similarly, the city information object requires the city name for which the information is to be provided. Thus, the task get tourist information has to be refined appropriately, e.g., by an interaction tasks “fill in city name”. We postpone further refinements to the interaction and dialog modelling since they describe the interface behaviour rather than domain and application specific high-level usage.

Overall, object-driven views on the domain model are basically decomposed into their data-driven and interaction-driven parts. Hereby, the model describes task-based and object-based access in a unified and very granular way. Based on this generic access components are defined by means of compound access units. These units describe the building blocks of the access space, which are specialized through the user interface design. We recommend this approach particularly if the design of the concrete user interface is under investigation or the access space should support multiple interfaces.

6 Conclusion

The modelling proposed in this paper supports equal modelling of tasks and objects (content), so that none of the models dominate per se the web site structure. All in all, we approach the design of the access space from both the task model and the object model and integrate them in the access model. Hereby object views can be refined up to the level of the included content information and interactions.
The main extension introduced by the WebTaskModel consists of the explicit description of task performance by means of state information. The modifications aim particularly at developing web applications but are applicable to traditional interactive systems as well, e.g., the concepts for modelling interruptions from the users’ point of view. Needless to mention that hereby also system functionality required for handling the interruptions can be referenced.

In our approach we distinguish between domain objects and views on them. This is also applied in WE modelling approaches - even sometimes not mentioned explicitly. Different in our approach is that we also make use of less detailed described task objects as defined in HCI. Similarly to [13] we describe modifications on task objects by means of state-transition-diagrams. In that work the task models are only used as an informal input to derive the object transitions. We retain the task model and bind it to the objects by pre-/post-conditions and events triggering the transitions. This is more flexible and allows using both models as formal input within the subsequent navigation and interaction design. First experience showed that developers tend to favour only one of the object types at a time depending on their background.

This paper also presents our steps towards generating an initial navigation model. This work basically combines two approaches: The derivation of an initial dialog model as followed by the HCI community, whereby focus is on the orchestration of activities - and the derivation of a navigation structure model as applied in WE, whereby focus is on the arrangement of content. It was shown by the notation used above, how the task flow structure can be translated into the node-link-terminology of navigation models. The task access structure units may be also useful in dialog modelling to visualize the initial dialog model. OOHDM [18] make use of User Interaction Diagrams (UIDs) focussing exclusively on the information exchanged between the application and the user. UIDs provide a notation for data-oriented dialog descriptions comparable to dialog graphs. Each state is specified by the data exchanged between a user and the web application. It is used as an informal input for navigation modelling later on, which is supported by guidelines. Furthermore, it is not formally integrated with the OOHDM business process modelling [16]. In contrast to OOHDM, on the one hand our approach is based on task performance that relates to users’ goals. On the other hand, the WebTaskModel is used formally to derive the initial access model.

Task models can be refined down to the dialog level, e.g., as done in WSDM and [10]. Alternatively, the dialog can be described by a separate dialog model, e.g., as introduced in [19]. Currently, we investigate both directions. In [1] a refined WebTaskModel is combined with so-called Abstract Dialog Units (ADUs). The task units are used as placeholders for objects (content, interaction and control objects). This is similar to WSDM but our approach allows attaching object views not only to leaf task. Hereby, we can define views relevant for several tasks.

The general objective of our work is to provide a modelling and runtime support for multiple user interfaces. WebTaskModel is used at the build time to generically define the task and domain specific behaviour of the site. The resulting models are transformed into a runtime system, whereby the task state machines becomes part of the controller, i.e., of a task-related controller [1, 2].
References

Model-Driven Web Engineering (MDWE 2007)

Geert-Jan Houben¹, Nora Koch² and Antonio Vallecillo³
¹Vrije Universiteit Brussel, Belgium, and Technische Universiteit Eindhoven, The Netherlands
²Ludwig-Maximilians-Universität München, and FAST GmbH, Germany
³Universidad de Málaga, Spain

G.J.Houben@tue.nl, kochn@pst.ifi.lmu.de, av@lcc.uma.es

http://wise.vub.ac.be/mdwe2007/
Foreword

Web Engineering is a specific domain in which Model-Driven Software Development (MDSD) can be successfully applied. Existing model-based Web engineering approaches already provide well-known methods and tools for the design and development of most kinds of Web applications. They address different concerns using separate models (content, navigation, presentation, business processes, etc.) and are supported by model compilers that produce most of the application’s Web pages and logic based on these models. However, most of these Web Engineering proposals do not fully exploit all the potential benefits of MDSD, such as complete platform independence, explicit and independent representation of the different aspects that intervene in the development of a Web application, or the ability to exchange models between the different tools.

Recently, the MDA initiative has introduced a new approach for organizing the design of an application into different models such that portability, interoperability and reusability can be obtained through architectural separation of concerns. MDA covers a wide spectrum of topics and issues like MOF-based metamodels, UML profiles, model transformations, modeling languages and tools. Another MDSD approach, Software Factories, provides concepts and resources for the model-based design of complex applications. At the same time, we see a trend towards the incorporation of emerging technologies like the Semantic Web and (Semantic) Web Rule Languages, which aim at fostering application interoperability. However, the effective integration of all these techniques with the already existing model-based Web Engineering approaches is still unresolved.

This workshop, which builds on the success of the preceding 2005 and 2006 Workshops (held, respectively, in Sydney jointly with ICWE 2005, and in Menlo Park jointly with ICWE 2006) aims at providing a discussion forum where researchers and practitioners on these topics can meet, disseminate and exchange ideas and problems, identify some of the key issues related to the model-based and model-driven development of Web applications, and explore together possible solutions and future work.

The main goal of the MDWE 2007 workshop is to offer a forum to exchange experiences and ideas related to Model-Driven Software Development in the Web Engineering field. Accordingly, we invited submissions from both academia and industry about the wide list of topics of interest stated in the Call for Papers.

In addition to the Workshop Organizers (Geert-Jan Houben, Nora Koch and Antonio Vallecillo), a selected program committee was set up to help reviewing and selecting the papers to be presented at the workshop. Members of the MDWE2007 Program Committee were Luciano Baresi, Jean Bézivín, Olga De Troyer, Piero Fraternali, Martin Gaedke, Athula Ginige, Jaime Gómez, Gerti Kappel, Esperanza Marcos, Maristella Matera, Pierre-Alain Muller, Alfonso Pierantonio, Vicente Pelechano, Gustavo Rossi, and Hans-Albrecht Schmidt.

In response to the call for papers, a total of 10 submissions were received. Each submitted paper was formally peer reviewed by at least two referees, and six papers were finally accepted for presentation at the Workshop and publication in the proceedings.
The selected papers focus on the following topics, which constitute the basis for the discussions of the three workshop sessions.

Freudentstein et al. discuss in their paper the use of domain specific languages and a model-driven approach based on the construction of models of Web application and the interpretation of these workflow-based models.

Kraus et al. discuss in their paper both an interpretational and a translational approach. For static aspects of Web applications the authors propose the use of transformation rules; for workflows of Web application instead the use of a virtual machine that provides a seamless bridge between models and code.

The paper by Wimmer et al. presents an approach for the integration of Web modeling languages using model transformation rules. The authors selected three Web modeling methods and the model transformation language ATL for the proof of concept.

In the paper of Pau Giner et al. the focus is on model-to-model transformations also specified in ATL and which are applied to BPMN resulting in Web applications written in BPEL.

De Castro et al. propose the use of a graph transformation language for the transformation of platform independent models. The authors focus on service-oriented Web applications.

The final paper, by Toffetti, focuses on modeling collaborative Web applications. He proposes the extension of a Web modeling language to support enriched interactions following the new RIA paradigm.

An additional non-reviewed paper reports on the recent MDWEnet project. MDWEnet is an initiative in the scope of MDWE and that counts on the participation of many of the people and groups from the Universities of Alicante, Málaga, Linz, Munich, Vienna and the Politecnico Milano related to the MDWE workshop from its origins.

The presented papers, together with the Call for Papers, and all the information relevant to the workshop, is available at the web site of the event, whose URL is http://wise.vub.ac.be/mdwe2007/ This web site contains the final workshop proceedings, and the latest information about the activities performed during the workshop.

Acknowledgements. We would like to thank the ICWE 2007 organization for giving us the opportunity to organize this Workshop, especially to the General Chair, Piero Fraternali, and the Workshops Chair, Emilia Mendes. They were always very helpful and supportive. Many thanks to all those that submitted papers, and particularly to the contributing authors. Our gratitude also goes to the reviewers and the members of the Program Committee, for their timely and accurate reviews and for their help in choosing and improving the selected papers. Finally we would like to specially thank Marco Brambilla, the ICWE 2007 Local Organization Chair, for his continuous assistance and support with all local arrangements, with the electronic submission system, and with the production of the workshops proceedings.

Como, Italy, July 2007
Geert-Jan Houben, Nora Koch, Antonio Vallecillo
MDWE 2007 Organizers
# Table of Contents

Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages.
*by Patrick Freudenstein, Jan Buck, Martin Nussbaumer and Martin Gaedke* 215

Model-Driven Generation of Web Applications in UWE.
*by Andreas Kraus, Alexander Knapp and Nora Koch* 230

*by Antonio Vallecillo, Nora Koch, Cristina Cachero, Sara Comai, Piero Fraternali, Irene Garrigós, Jaime Gómez, Gerti Kappel, Alexander Knapp, Maristella Matera, Santiago Meliá, Nathalie Moreno, Birgit Pröll, Thomas Reiter, Werner Retschitzegger, José E. Rivera, Andrea Schauerhuber, Wieland Schwinger, Manuel Wimmer, Gefei Zhang* 246

On the Integration of Web Modeling Languages.
*by Manuel Wimmer, Andrea Schauerhuber, Wieland Schwinger and Horst Kargl* 255

Bridging the Gap between BPMN and WS-BPEL. M2M Transformations in Practice.
*by Pau Giner, Victoria Torres and Vicente Pelechano* 270

Model Transformation for Service-Oriented Web Applications Development.
*by Valeria de Castro, Juan Manuel Vara and Esperanza Marcos* 284

Modeling data-intensive Rich Internet Applications with server push support.
*by Giovanni Toffetti Carughi* 299
Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages

Patrick Freudenstein¹, Jan Buck¹, Martin Nussbaumer¹, and Martin Gaedke²

¹ University of Karlsruhe, Institute of Telematics, IT Management and Web Engineering Research Group, Engesserstr. 4, 76128 Karlsruhe, Germany {freudenstein, buck, nussbaumer}@tm.uka.de
² Chemnitz University of Technology, Faculty of Computer Science, Distributed and Self-organizing Computer Systems Group, Straße der Nationen 62, 09107 Chemnitz, Germany gaedke@informatik.tu-chemnitz.de

Abstract. The requirements for Web applications concerning workflow execution, interaction, aesthetics, federation and Web service integration are steadily increasing. Considering their complexity, the development of these “rich workflow-based Web applications” requires a systematic approach taking key factors like strong user involvement and clear business objectives into account. To this end, we present an approach for the model-driven construction and evolution of such Web applications on the basis of workflow models which is founded on Domain-specific Languages (DSLs) and a supporting technical framework. We describe our approach’s core DSL for workflow modeling which supports various modeling notations like BPMN or Petri nets and outline a set of DSLs used for designing workflow activities like dialog construction, data presentation and Web service communication. In conclusion, rich workflow-based Web applications can be built by modeling workflows and activities and passing them to the associated technical framework. The resulting running prototype can then be configured in detail using the presented DSLs.

Keywords: Web Engineering, Workflow, Domain-specific Languages, Reuse, Evolution, Web Services, SOA, EAI

1 Introduction

The World Wide Web has evolved from a decentralized information medium to a platform for basic e-commerce applications. Currently, the next step in its evolution cycle towards a platform for sophisticated enterprise applications and portals with strong demands regarding workflow execution, rich user interaction, aesthetics, and strong Web service integration is taking place [16, 17]. Especially in the context of Enterprise Application Integration (EAI), Enterprise Information Integration (EII) or Business-to-Business (B2B) scenarios, these workflow-driven Web applications are gaining more and more importance. In order to cope with the immense increase in
these Web applications’ complexity and their permanent evolution, a dedicated engineering methodology is required.

Besides specific requirements resulting from this new type of applications, a suitable engineering approach must also consider key factors like strong user involvement and clear business objectives arising from a project management’s perspective. Their strong influence on a project’s success was proved in comprehensive studies [20] and taken on in agile software development methods [2], being reason enough to soundly integrate them in today’s Web engineering methods.

Facing these challenges, we present an evolutionary approach for the model-driven construction of rich, Web service-based Web applications on the basis of workflow models. The approach is based on our previous work, namely the WebComposition approach [8], the WebComposition Service Linking System (WSLS) [7] and our latest approaches towards DSL-based Web Engineering [14, 15]. By providing dedicated Domain-specific languages (DSLs) and an underlying technical framework, stakeholders and domain experts with diverse backgrounds and knowledge are enabled to directly contribute to the development effort. They can easily understand, validate and even develop parts of the solution which in turn leads to a much more intense collaboration and lowers the possibility of misunderstandings.

In section 2, we introduce a business process from a real-world scenario within a large-scale EAI-project to which we will refer to throughout the paper. We elaborate the particular requirements a systematic engineering approach for the described problem scope must fulfill. Section 3 gives a comprehensible overview of our evolutionary, DSL-based engineering method. In section 4, we describe the core DSLs for workflow modeling in detail and outline supporting DSLs for rich dialog construction, data presentation and Web service communication. Moreover, we present the supporting technical framework being able to interpret a workflow model and to assemble a corresponding Web application. Based on the presented scenario, we show exemplarily how a workflow modeled in e.g. BPMN notation and enriched with little annotations can directly be transferred into a running Web application. In section 5, we give an overview of related work. Finally, section 6 concludes the paper and outlines future work.

2 Challenges with Developing Workflow-based Web Applications

In the following, we first present a real-world scenario from a large-scale EAI project that serves as a running example throughout the paper. Subsequently, we introduce a general core set of requirements an engineering methodology for the systematic construction and evolution of modern workflow-based Web applications should meet.

2.1 The KIM Project - An Example EAI Scenario

We have been collaborating in the project “Karlsruhe’s Integrated InformationManagement (KIM)” [13], a university-wide EAI-project, for several years now. One of the main challenges in this project is the extraordinary decentralized organizational structure of a university.
Technical challenges: On the one hand, there exists a huge diversity of heterogeneous IT systems which have to be integrated in order to enable a uniform access to information. In order to cope with these integration challenges, the KIM project is founded on a multi-layered Service-oriented Architecture (SOA). Therein, canonical Web service wrappers provide homogeneous access to existing heterogeneous legacy systems. These Web services are then orchestrated to realize value-added functionalities which are also exposed via Web service interfaces. Finally, the portal layer comprises mainly Web portals providing a centralized user interface for accessing the highly distributed Web services.

On the other hand, business processes are mostly spanning several organizational units and IT systems, thus suffering from media discontinuity issues. To improve the efficiency and quality of these processes, there is an urgent need for a support platform allowing for their integrated and uniform execution.

Communication challenges: Besides these technical aspects, we found communication problems making up the second major problem area. Stakeholders belong to different faculties and departments with entirely different education and professional background. Hence, when specifying business processes with stakeholders from all over the university, each group uses its own “language”. For example, some of them prefer Petri nets as a means of communication as they play a major role in their research context. Others favor the Business Process Modeling Notation (BPMN) [21] or UML Activity Diagrams for the same reason. And people with a background in humanities often like a notation in natural language better. However, assuring efficient, non-ambiguous and intense communication is especially in phases like requirements engineering and conceptual design a key factor [20].

2.2 The Master Thesis Business Process Example

One of our main goals within the KIM project was the development of a Web portal for all students of the university serving as a uniform access point to all study-relevant information and business processes. Within this paper, we will focus on a workflow-driven feature which supports the complete Master Thesis business process for all involved parties: The student, the advisors, the examination office and the library.

![Fig. 1. Excerpt from the Master Thesis business process](image-url)
Fig. 1 shows an excerpt from the Master Thesis business process modelled in BPMN. We chose BPMN as the involved stakeholders found this notation rather intuitive. However, in the following chapters, we will also outline how other notations, e.g. Petri nets, can be employed within our approach. Our example starts with the student submitting the final version of her thesis. Next, the advisor downloads and reviews it and submits her expertise. The associated professor reviews the expertise, which is then submitted to an existing exam management system via a Web service wrapper and further processed by the examination office. The university library approves the electronic thesis document, usually a PDF file, whereupon a dataset for the thesis is created in the library’s central catalogue.

This process excerpt is a typical example found in advanced workflow-driven Web applications, especially in EAI, EII and B2B scenarios. It comprises different organizational units and roles, it contains typical building blocks like complex dialogs, data rendering and Web service communication, and various stakeholders are involved in the aspired Web application’s specification. Moreover, due to the permanent restructurings taking place in the context of the Bologna Process [6] as well as the current merger of the University of Karlsruhe towards the “Karlsruhe Institute of Technology (KIT)”, our business processes underlie frequent changes.

2.3 Requirements for a Workflow-Driven Web Engineering Methodology

From the general requirements for Web Engineering methods found in literature (e.g. [5, 9]) as well as based on our experiences in real-world projects, we identified the following requirements to be particularly important for a methodology targeting the construction and evolution of workflow-based Web applications. While the first three requirements concern the development process and the methodology itself, the last three aim at technical characteristics of modern workflow-based Web applications.

**Agility & Evolution**: Web applications in general and workflow-based Web applications in particular underlie a continuous evolution due to frequent changes, e.g. adjustments in the business process’ structure, integration of new partners or presentational changes. Thus, agility in terms of supporting short revision lifecycles and the efficient adoption of such changes is essential. To this end, a model-driven approach seems to be a good solution as it allows for comparatively easy changes in the models which are then automatically propagated to the actual implementation. However, assuring consistency between models and implementation is crucial.

**Reuse**: With respect to requirements from the fields of evolution support, development efficiency and software quality, the systematic reuse of all kinds of artifacts throughout the development process plays an important role. Regarding workflow-based Web applications, especially the reuse of workflow models in whole or part as well as typical workflow building blocks, like e.g. dialogs, Web service communication and data rendering is of great interest. Thus, an engineering method should address reuse as a guiding principle throughout the development process.

**Strong Stakeholder Involvement**: In order to assure clear business objectives and to avoid misunderstandings between the developers and the business, stakeholders should be strongly incorporated in the development process. Especially regarding workflow-based Web applications, the future end-users know the underlying business
processes best. Thus, a dedicated engineering methodology must take into account the
great diversity of stakeholders with different backgrounds and skills. Therefore, the
methodology should allow for dedicated modeling languages and notations hiding
unwanted complexity and being tailored to specific stakeholder groups [14].
Moreover, the ability to provide running prototypes from the very beginning of the
development process further supports the communication between the end-users and
the developers. Discrepancies between the requirements and the realization can be
identified in the early stages of the development process and cost-efficiently resolved.

Rich User Interfaces: As dialogs play a dominant role in workflow-based Web
applications, their usability has a great influence on how efficiently process
participants can complete their tasks and thus contribute to the business process. Due
to the increasing complexity of these tasks and the underlying data models, rich user
interfaces supporting the users by reducing the cognitive overload are required.
Therefore, these dialogues should be highly dynamic and offer guidance by e.g.
showing only relevant options and providing immediate feedback and hints. Thus,
their usability can be considerably improved [13]. Beyond that, aspects from the field
of accessibility, i.e. providing accessible interfaces for people with disabilities have to
be considered [22], especially in the public sector due to recent legal regulations.

Federative Workflows: Workflows based on business processes are - in contrast
to simple page flows - long-running and affect different people or roles respectively.
Advanced workflow scenarios (e.g. B2B) even span multiple companies. This means,
they involve people from different organizations and rely on multiple, distributed
information systems. The integration of these systems is usually realized via Web
service interfaces. Thus, supporting long-running and federative workflows as well as
comprehensive Web service support are key requirements.

Multimodal Participation: In advanced workflow scenarios, e.g. in supply chain
management, parts of a workflow take place outdoors or away from computers. Then,
process participants must be able to collaborate with other devices, e.g. PDAs or
smart phones. Beyond that, some tasks are better conducted in dedicated, task-specific
applications, e.g. a spreadsheet application. Thus, even though trying to offer one
integrated, browser-based user interface is a desirable objective, workflow-based Web
applications must also allow for completing tasks off the browser.

3 An Evolutionary DSL-based Engineering Approach

In this section, we give an overview of our evolutionary engineering methodology for
the model-driven construction of workflow-based Web applications starting from
business process models. The details of the methodology as well as the associated
technical framework will be explained in section 4. Both were designed and
implemented with strong adherence to the requirements identified in section 2.

3.1 The Workflow DSL

The model-driven construction is based on previous work in the fields of DSL-based
Web Engineering [15] and thus is founded on a dedicated Workflow DSL as a core
A DSL can be seen as programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. By providing various graphical notations and accompanying editors, each of them being as intuitive as possible for a particular stakeholder group, the usability of a DSL can be further improved. According to this definition, the Workflow DSL is an executable specification language for workflow-based Web applications which allows the use of various graphical notations known from the Business Process Modeling field, e.g. BPMN, Petri Nets, UML activity diagrams etc. as well as custom notations. By providing stakeholder-specific notations according to their individual skills and preferences, stakeholders can easily understand, validate and even specify parts of the solution being constructed. Following our DSL-based Web Engineering concept, the Workflow DSL consists of three core elements:

**Domain-specific Model (DSM):** The DSM represents the formal schema for all “DSL programs” that can be described with the DSL. With respect to our requirement to support various process modeling notations, the DSM can also be seen as a “Process Intermediate Schema”, representing an (as far as possible) common denominator of multiple existing process modeling languages. Beyond business process information, the DSM comprises dedicated modeling constructs necessary for the transition from a pure business process model to a running workflow-based Web application. We chose the XML Process Definition Language (XPDL) [19] as a basis for the DSM. Serving both as an interchange format for process definitions and as a definition language for executable workflow specifications (including human interaction aspects) belonged to the major design goals of XPDL, making it an ideal foundation for our DSL. The extensibility mechanisms provided by XPDL were used to shape our DSM.

**Domain Interaction Model(s) (DIM):** Based on the DSM, a DIM comprises a dedicated (graphical) notation being as intuitive as possible for a particular stakeholder group. By using a DIM, stakeholders can understand, validate and even create DSL programs without being confronted with complicated source code. Within the Workflow DSL, multiple DIMs for various stakeholder groups could be defined. Thereby, a DIM could either be derived from a well-known business process modeling notation like e.g. BPMN or Petri nets, or defined from scratch based on a custom notation. According to different incremental stages of the Web application construction process, DIMs can also cover only parts of the DSM. Accompanying editors support stakeholders in creating DSL programs based on a DIM notation.

**Solution Building Block (SBB):** A SBB is a software component being capable of executing programs developed with the DSL. Therefore, the SBB does not generate code but rather adapts its behavior according to a given DSL program. Thus, the SBB of the Workflow DSL can be configured with an XML-based specification of a workflow-based Web application, i.e. a Workflow DSL program. Thereupon, it constructs at runtime an associated workflow-driven Web application prototype. This immediately executable prototype employs SBBs from other DSLs to realize workflow activities like e.g. dialog construction, data presentation or Web service communication. These SBBs were initialized with a minimum configuration set derived from the workflow model and can then be configured in detail using the associated DSLs or their DIMs respectively.
3.2 The DSL-based Process Model

Fig. 2 gives an overview of our methodology’s underlying evolution- and reuse-oriented process model as well as the involved roles. It is based on the WebComposition approach [8] and consists of three phases in a continuous evolution.

**Business Process Modeling:** In this first phase, the business process to be realized by the workflow-based Web application is modeled using pure business process modeling constructs. Thereby, stakeholders representing the involved process participants and knowing the business process best as well as a process analyst supporting the modeling itself are involved. Moreover, the ‘Reuse Librarian’ role advises the modeling team regarding possibilities for reusing existing process models in whole or part. The resulting business process model is created by employing adequate DIMs and associated editors from the Workflow DSL.

**Workflow Modeling:** In this phase, the business process model from the previous phase is supplemented with workflow execution-relevant information, the ‘Concern Configuration’. Thereby, to each activity in the process model a corresponding activity building block is assigned, whereby each activity building block has an associated DSL for its configuration. In our experience, a small core set of building blocks for dialog construction and processing, data rendering and Web service communication was sufficient for the most cases. If no suitable building block exists, the ‘Developer’ role designs and implements a new one. In addition to selecting the activity building block type, a minimum set of configuration information has to be provided. This set contains properties from the different concerns of a Web application, i.e. data, navigation, presentation, dialog, process, and communication. This minimum configuration set assures the automatic setup of a running prototype of the workflow-based Web application in the next phase. The detailed configuration of the activity building blocks is usually performed in the next phase by means of the associated DSLs. Like the previous phase, the Workflow Modeling phase is again conducted by stakeholders who are supported by an application designer and the reuse librarian. The application designer is experienced in workflow modeling and knows the activity building blocks, the associated DSLs and the required minimum
configuration sets. She assists the modeling team in related issues. The reuse librarian advises the team concerning the reuse of existing Concern Configurations and activity building blocks from the reuse repository. The Workflow modeling phase is also supported by adequate Workflow DSL DIMs and accompanying editors, usually the same as used in the previous phase but extended with Concern Configuration facilities. The result of this phase is a valid Workflow DSL program in form of an XML document, whereby process structure information and Concern Configuration are loosely coupled, thus easing reuse and evolution.

Physical Design & Execution: This phase deals with the transformation of the DSL program into a running prototype of the aspired workflow-based Web application. Therefore, the DSL program is passed to the Workflow DSL’s SBB, which configures an associated workflow-driven Web application prototype. This prototype can then either be configured in detail using the activity building blocks’ associated DSLs or directly be used for creating and processing workflow instances. The WebComposition Service Linking System (WSLS) [7] serves as our approach’s technical platform and facilitates the assembly and configuration of SBBs. As in the previous phases, stakeholders can strongly participate in this phase, assisted by an ‘Application Configurator’ role who is experienced in WSLS and its configuration capabilities as well as the activity building blocks’ associated DSLs.

Evolution: In case of changing or new requirements, our method provides strong support for adopting changes, either in the business process model or the Concern Configuration or both. Changes in the business process can easily be performed in the Business Process Modeling Phase while keeping the Concern Configuration in the Workflow Modeling phase unchanged. Changes in the Concern Configuration can either be performed on model-level in the Workflow Modeling phase or directly in the Physical Design & Execution phase by using appropriate DSLs. Our approach and the technical platform preserve model consistency throughout all phases.

4 The Workflow DSL Approach Applied - Realization Details

This section describes our approach in detail based on the example excerpt from the ‘Master Thesis’ business process presented in section 2 (Fig. 1), which could be the output of the Business Process Modeling phase and serves as a starting point. To ease the understanding of the following subsections, section 4.1 outlines a selection of activity building blocks for the realization of workflow activities. Section 4.2 focuses the Workflow Modeling phase and shows how an appropriate Concern Configuration could look like. We present our XPDL-based DSM, an adequate DIM editor and excerpts from the resulting DSL program. Finally, section 4.3 covers the Physical Design & Execution phase, and presents the approach’s underlying technical platform for constructing, configuring and executing workflow-based Web applications.

4.1 Activity Building Block DSL Catalogue

A major design goal of our activity building blocks was that one activity from a business perspective can be realized by one activity building block, and must not be
split up into several activities from a system perspective. Thereby, the business process model’s structure can be kept throughout the construction process, easing the collaboration with stakeholders. Thus, as we especially aim at service-based Web applications, the Web Service Communication building block can be integrated with other building blocks. Therefore, the DSLs allow for the loosely coupled integration of external code in their programs which is forwarded at runtime to the appropriate SBB. For example, a Dialog Construction DSL program can thus submit a filled form to a Web service.

**Web Service Communication:** This DSL allows the specification and execution of Web service calls. The DSM is represented by an XML Schema which defines, amongst others, elements for specifying the Web Service endpoint, WSDL URL, operation name, input parameters and security policy information based on the WS-Security Policy standard (to be submitted to OASIS). The DIM editor is realized in form of a property editor. The SBB generates a SOAP message according to the DSL program, sends it to the Web service and returns the received response.

**Dialog Construction:** This DSL is used for the specification of highly interactive and dynamic dialogs. The DSM is based on the W3C XForms standard. We defined a DIM notation on the basis of the XForms user controls as well as Petri net-based structures for modeling the dialog’s dynamic behavior. The SBB is capable of automatically creating an XForms-based form prototype from an XML Schema specification or a WSDL document. Moreover, it renders and processes the XForms document by means of a JavaScript library. A browser-based DIM editor allows the detailed design of the form, including the verification of accessibility guidelines [12].

**Data Presentation:** This DSL addresses the presentation of data. The DSM provides elements for referencing the data to be displayed as well as an XSL document specifying the data transformation to the desired output format. If a DSL program contains no reference to an XSL document, the SBB automatically generates a prototypical XSL transformation into XHTML. A browser-based DIM editor allows the detailed design of XSL stylesheets on the basis of Cascading Style Sheets (CSS), again with integrated verification facilities for accessibility guidelines.

### 4.2 Workflow Modeling

The Workflow DSL’s DSM and the associated DIMs constitute the basis for the Workflow Modeling phase. The DSM is based on XPDL which we extended by dedicated modeling elements as depicted in Fig. 3.

![Fig. 3. WebComposition XPDL extensions appended to existing ApplicationTypes](image-url)
The XPDL schema contains so-called “ApplicationTypes” for modeling applications used for performing activities. We extended the XPDL ApplicationTypes “Form”, “WebService” and “Xslt” in order to integrate properties for the configuration sets required by the activity building blocks presented above. Thereby, the “WebComp_Dialog_Extension” serves for the Dialog Construction building block, the “WebComp_WS_Extension” for the Web Service Communication building block and the “WebComp_DataPresent_Extension” for the Data Presentation building block. Beyond the minimum configuration set, we are currently working on a full coverage of the DSLs’ modeling elements in form of attributes. Thus, fully configured building blocks resulted from prior Physical Design & Execution phases and stored in the Reuse Repository could be already reused in the Workflow Modeling phase.

To provide support during the Workflow Modeling phase, we adapted Microsoft Visio as a visual editor for a BPMN-based DIM notation (Fig. 4). The editor provides shapes according to the BPMN notation (left pane) as well as a property editor for annotating the Concern Configuration, i.e. the selection of an activity building block and its corresponding minimum configuration, to process activities (bottom pane). In the picture, the ‘Create Expertise’ activity is currently modeled as a Dialog Construction building block and a data schema for the dialog as well as attributes regarding the Web service the form shall be submitted to are provided.

Fig. 4. A BPMN-based DIM Editor in the Workflow Modeling Phase

Having completed the Workflow Modeling, the workflow model can be exported as a valid Workflow DSL program. Thereby, the mapping of the BPMN notation and the annotated Concern Configuration to a DSM-conform DSL program is performed. The mapping of BPMN symbols to XPDL elements is described in the XPDL specification [19]. Regarding other graphical notations, additional mappings have to be defined. We are currently working on a DIM based on the Petri Net notation and to be supported by the process modeling tool INCOME [11]. As our support for various notations aims primarily at improving stakeholder collaboration by providing intuitive notations, their usability and simplicity is more important than covering even the most
complex semantic aspect of a DIM notation’s underlying modeling language. Having analyzed a great variety of process models from the KIM project, we found that XPDL provides a sufficient (and extensible) set of generic business process modeling elements, being suitable to accomplish a variety of DIM notations derived from popular modeling languages like e.g. Petri nets and UML Activity Diagrams.

The following code snippets are part of the resulting Workflow DSL program. Extract (1) shows the representation of the ‘CreateExpertise’ activity. According to the business process model shown in Fig. 1, it is assigned to the ‘Advisor’ role. The activity is linked to the ‘CreateExpertiseDialog’ application definition which defines a dialog according to the Concern Configuration shown in Fig. 4.

4.3 Physical Design & Execution – The Technical Platform

Fig. 5 depicts the approach’s underlying technical platform, mainly consisting of the Workflow Web Service and the Workflow SBB running on the WSLS framework.
The Workflow Web Service has two core functions: First, it is used to manage workflow definitions, i.e. Workflow DSL programs, and workflow instances, via a CRUDS (Create, Read, Update, Delete, Search) interface. Second, it can be used to participate in a workflow by retrieving the current tasks for a particular role (GetTaskList) or the actual input parameters for a given task of a workflow instance (GetTaskData) as well as sending the results of a completed task back to the workflow instance (CommitTaskData). For the realization of these functions, the Microsoft Windows Workflow Foundation (WF) is used as workflow engine. When creating a new workflow definition, the Web service converts the process structure from the DSL program into an executable WF library which serves as input for the engine. By encapsulating these functionalities in a Web service, all kinds of clients from any platform can participate in a workflow across organizational borders.

The WSLS framework supports the systematic development and evolution of Web applications by assembling and configuring components with respect to the ‘Separation of Concerns’ principle at runtime. It aims at realizing the ‘Configuration instead of Programming’ paradigm and thus at making the process of evolution faster and more flexible. The Workflow SBB is a WSLS component that can be configured with a Workflow DSL program which is then sent to the Workflow Web Service in order to create a new workflow definition. As all DSL programs are accessible via the Web service, other WSLS installations can easily retrieve them, thus enabling federation scenarios. The Workflow SBB uses the Concern Configuration information contained in the DSL program to instantiate and configure a child component for each process activity. Therefore, it employs the SBBs of the presented activity building blocks which possess automation features enabling fast prototypes with minimal
configuration as well as a ‘Commit Activity’ component. The latter is used for confirming the completion of tasks performed offside a PC like shipping a package. Furthermore, the Workflow SBB instantiates two default child components: ‘Workflow Management’ for starting and managing workflow instances, and ‘Task List’ (Fig. 6a) for displaying a cross-workflow task list for the currently logged in user. From this moment on, all child components can be configured in detail (at runtime) using the associated DSLs and the comprehensive WSLS configuration facilities. The changed configurations are propagated back to the DSL program to preserve consistency between physical design and the workflow model. Fig. 6c shows the detailed design of the ‘Create Expertise’ dialog using the Dialog Construction DSL’s DIM editor. If the user selects a task, the Workflow SBB retrieves the task input parameters from the Web service and displays the child component associated with the task (Fig. 6b). After the completion of a task, the Workflow SBB sends the results back to the Web service which passes it to the workflow engine. Afterwards, the Workflow SBB retrieves the new task list for the current user and the current workflow instance and displays it or - in case of only one outstanding task - directly switches to the page containing the associated child component.

5 Related Work

Having recognized the increasing importance of workflow-based Web applications, established Web Engineering approaches like e.g. OOHDM [18], UWE [10], and WebML [3] were extended towards modeling workflow-based Web applications. In the following, we will point out the differences compared with our approach based on the requirements presented in section 2.

All of them are model-driven design approaches, i.e. they incorporate business processes in their various Web application models and thereof generate the Web application’s code. Thus, they support agility and evolution in terms of adopting changes on a model-level. However, changes in the generated code are not propagated to the model-level and thus get lost when regenerating an application.

Model reuse is considered in the above mentioned approaches. However, reuse in terms of component reuse, i.e. reusing common activity building blocks, is not covered. Thus, the design models derived from business process models still require intensive design effort regarding the concrete realization of process activities. In [1], an advanced generic workflow modeling approach considering also application logic contained in activities is presented. Though, they focus primarily on data processing and navigation and leave out other concerns like presentation or interaction.

The strong involvement of stakeholders throughout the development process by providing dedicated notations for individual stakeholder groups has not been in the focus of current Web Engineering approaches yet. However, by means of model transformations, they could also support multiple process modeling notations. The integration of stakeholders in later stages beyond business process modeling has also to be investigated further. Audience-driven approaches, e.g. from DeTroyer [4], rather consider the future end-users, their characteristics and information requirements in their design process in order to develop user-centered Web applications.
Presentation modeling naturally plays a major role in most of today’s Web Engineering approaches. However, only few of them, e.g. WebML, have also presented concepts concerning the modeling of dynamic and adaptive user interfaces required for reducing the cognitive overload caused by complex dialogs. The integrated consideration of accessibility guidelines in the development process is also an open issue in most approaches.

Regarding technical requirements like federative workflows and multimodal participation, most approaches provide only limited support. OOHDM, for example, mentioned the need for supporting federative workflows in their above cited paper’s outlook. WebML and UWE already support long-running workflows with different roles, but so far their technical platform do not allow for federated participation scenarios, e.g. from other Web portals or multimodal access from diverse clients.

Beyond the only limited fulfillment of the above requirements, each of the existing Web Engineering approaches has other unique characteristics and ideas worth considering and learning from. WebML, for example, presented inspiring work in the field of exception handling and constraint modeling. Likewise, OOHDM’s concepts for adapting business processes to given contexts and users is also an interesting topic not yet covered by our approach. The UWE approach, for example, is very strong in the field of meta-modeling and model-transformations. Their ideas and concepts serve as a valuable source for the transformations required in our approach.

6 Conclusions & Future Work

Facing the challenges found in the development and evolution of advanced workflow-based Web applications, we presented a methodology for their model-driven construction employing Domain-specific Languages. DSLs can define various modeling notations, each of them being as intuitive as possible for a particular stakeholder group and thereby improving communication and collaboration throughout all stages of the construction process. Dedicated software components can be configured at runtime with DSL programs and adapt their behavior accordingly.

Our approach places emphasis on evolution and reuse and enables rapid prototypes automatically derived from slightly annotated business process models. This is supported by a set of activity building blocks for the realization of rich dialogs, data presentation and Web service communication, each of them being able to work with a minimum configuration set derived from the workflow model. Associated DSLs allow their detailed design at runtime, thereby assuring consistency between models and implementation. The approach’s technical platform provides strong support for evolution and reconfiguration and enables the execution of federative workflows and multimodal participation.

As the presented approach is the result of more than two years research, not all aspects could be described in detail. Thus, we are working on further publications focusing e.g. the formal transformations from different process modeling notations to our XPDL-based Process Intermediate Language and from there to the specification formats of various workflow engines. This is achieved by employing model transformation techniques like XSLT, QVT and ATL. Moreover, we will present the
details of our activity building blocks, especially the Dialog Construction DSL, in a separate paper. Beyond that, interesting extensions were given in section 5.

References

Model-Driven Generation of Web Applications in UWE

Andreas Kraus, Alexander Knapp, and Nora Koch

Ludwig-Maximilians-Universität München, Germany
{krausa, knapp, kochn}@pst.ifi.lmu.de

Abstract. Model-driven engineering (MDE) techniques address rapid changes in Web languages and platforms by lifting the abstraction level from code to models. On the one hand models are transformed for model elaboration and translation to code; on the other hand models can be executable. We demonstrate how both approaches are used in a complementary way in UML-based Web Engineering (UWE). Rule-based transformations written in ATL are defined for all model-to-model transitions, and model-to-code transformations pertaining to content, navigation and presentation. An UWE run-time environment allows for direct execution of UML activity models of business processes.

1 Introduction

Model-driven engineering (MDE) technologies offer one of the most promising approaches in software engineering to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively [18]. MDE advocates the use of models as the key artifacts in all phases of the software development process. Models are considered first class entities even replacing code as primary artifacts. Thus, developers can focus on the problem space (models) and not on the (platform specific) solution space [18]. The main objective is the separation of the functionality of a system from the implementation details following a vertical separation of concerns.

The area of Web Engineering, which is a relatively new direction of Software Engineering that addresses the development of Web systems [8], is as well focusing on the MDE paradigm. Most of the current Web Engineering approaches (WebML [5], OO-H [6], OOWS [20], UWE [9], WebSA [15]) already propose to build different views of Web systems following a horizontal separation of concerns.

The most well known approach to model driven engineering is the Model Driven Architecture (MDA) defined by the Object Management Group (OMG)². Applications are modeled at a platform independent level and are transformed by model transformations to other models and (possibly several) platform specific implementations.

¹ This research has been partially supported by the project MAEWA “Model Driven Development of Web Applications” (WI841/7-1) of the Deutsche Forschungsgemeinschaft (DFG), Germany and the EC 6th Framework project SENSORIA “Software Engineering for Service-Oriented Overlay Computers” (IST 016004).

The development process of the UML-based Web Engineering (UWE [9]) approach is evolving from a manual process (based on the Unified Process [7]) through a semi-automatic model-driven process (based on different types of model transformations [9]) to a model-driven development process that can be traversed fully automatically in order to produce first versions of Web applications. A table of mapping rules is shown in [9]. The UWE approach uses recently emerged technologies: model transformation languages like ATL\(^3\) and QVT\(^4\). In this paper we present a fully implemented version using the ATLAS environment and based on ATL transformations.

UWE applies the MDA pattern to the Web application domain from the top to the bottom, i.e. from analysis to the generated implementation ([9][12]). Model transformations play an important role at every stage of the development process. Transformations at the platform independent level support the systematic development of models, for instance in deriving a default presentation model from the navigation model. Then transformation rules that depend on a specific platform are used to translate the platform independent models describing the structural aspects of the Web application into models for the specific platform. Finally, these platform specific models are transformed (or serialized) to code by model-to-text transformations. In a complementary way platform independent models describing the business processes are executed by a virtual machine thus providing a seamless bridge between modeling and programming. The UWE approach is completely based on standards – in the same way as MDA – facilitating extensibility and reusability.

The remainder of this paper is structured as follows: First we give a brief overview on model-driven Web engineering. Next we present the UWE approach to MDWE and focus on the model-based generation of Web application using the model transformation language ATL and the run-time environment for executable business processes. We conclude with a discussion on related work, some remarks and an outline of our future plans on the use of model transformation languages.

2 Model-Driven Web Engineering

Model-driven Web Engineering (MDWE) is the application of the model-driven paradigm to the domain of Web software development where it is particularly helpful because of the continuous evolution of Web technologies and platforms. Different concerns of Web applications are captured by using separate models e.g. for the content, navigation, process and presentation concerns. These models are then integrated and transformed to code whereas code comprises Web pages, configuration data for Web frameworks as well as traditional program code.

2.1 Variants in Model-Driven Approaches

Model-driven Web development is an effort to raise the level of abstraction at which we develop Web software. MDWE processes can be achieved either by code genera-
tion from models or by constructing virtual machines that execute models directly. The translational approach is predominant and is supported by so-called model transformations. An interpretational approach offers the benefits of early verification through simulation, the ability to separate implementation decisions from understanding of the problem, and the ability to execute the models directly and efficiently on a wide variety of platforms and architectures.

In addition, following the classification of McNeile there are two interpretations of the MDE vision named “elaborationist” and “translationist” approaches [14]. Following the “elaborationist” approach, the specification of the application is built up step by step by alternating automatic generation and manual elaboration steps on the way from a computational independent model (CIM) to a platform independent model (PIM) to a platform specific model (PSM) to code. Today, most approaches based on MDA are “elaborationist” approaches, which have to deal with the problem of model and/or code synchronization. Some tools support the regeneration of the higher-level models from the lower level models, also called reengineering. In a “translationist” approach the platform independent design models of an application are automatically transformed to platform specific models, which are then automatically serialized to code. These PSMs and the generated code must not be modified by the developer because roundtrip engineering is neither necessary nor allowed. UWE is moving from an “elaborationist” to a “translationist” approach.

2.2 Role of Model Transformations

Transformations are vital for the success of a model-driven engineering approach for the development of software or in particular of Web applications. Transformations lift the purpose of models from documentation to first-class artifacts of the development process. Based on the taxonomy of transformation approaches proposed by Czarnecki [4] we classify model transformation approaches in model-to-code and model-to-model approaches, which differ on providing or not a metamodel for the target programming language.

Model-to-model transformations translate between source and target models, which can be instances of the same or different metamodels supporting syntactic typing of variables and patterns. Two categories of rule-based transformations are distinguished: declarative and imperative and it is worth to mention two relevant subcategories of declarative transformations: graph transformation and relational approaches. Graph-transformation rules that consist in matching and replacing left-hand-side graph patterns with right-hand-side graph patterns. AGG$^5$ and VIATRA$^6$ are examples of tools supporting a graph model transformation approach. Relational approaches are based on mathematical relations. A relation is specified by defining constraints over the source and target elements of a transformation. A relation has no direction and cannot be executed. Relational approaches with executable semantics are implemented with logic programming using unification-based matching, search and backtracking. QVT and ATL support the relational approach and provide impéra-

---

tive constructs for the execution of the rules, i.e. they are hybrid approaches. A different approach is the declarative and functional transformation language XML for transforming XML documents. As MOF compliant models can be represented in the XML Metadata Interchange format (XMI), XSLT could in principle be used for model-to-model transformations and transformed to code using languages such as MOFScript that generate text from MOF models. This approach has scalability problems, thus it is not suited for complex transformations.

3 The UWE Approach to MDWE

The approach followed in UWE consists of decoupling the construction of models using a UML profile, defining transformation rules using a model transformation language, and providing a run-time environment, respectively. Model type transformations map instances from a source metamodel (defining the source types) to instances of a target metamodel (defining the target types). Therefore, the MDE approach is based on metamodels and transformations. All metamodels share the same meta-metamodel (MOF). The proposed solution for the executable models is the use of a platform specific implementation of the platform independent abstract Web process engine. A transformation maps the process flow model to XML nodes, which represent the corresponding configuration of the runtime environment. This runtime process engine is part of the platform specific runtime environment.

There is no restriction on the employed modeling tool as long as it supports UML profiles and stores models in the standardized model interchange format. We selected ATL as a Query/View/Transformation language and the ATLAS transformation engine. The Spring framework was selected as runtime environment engine.

3.1 UWE Process

Applying the MDA principles (see Sect. 2), the UWE approach proposes to build a set of CIMs, PIMs, and PSMs as results of the analysis, design and implementation phases of the model-driven process. The aim of the analysis phase is to gather a stable set of requirements. The functional requirements are captured by means of the requirements model. The requirements model comprises specialized use cases and a class model for the Web application. The design phase consists of constructing a series of models for the content, navigation, process, presentation and adaptivity aspects at a platform independent level. Transformations implement the systematic construction of dependent models by generating default models, which then can be refined by the designer. Finally, the design models are transformed to the platform specific implementation. This UWE core process is extended with the construction of a UML state machine – called “big picture” in our approach – that integrates the design models. The objective of the big picture model is the verification of the UWE
models by the tool Hugo/RT, a UML model translator for model checking and theorem proving [10]. In addition, architectural features can be captured by a separate architecture model using the techniques of the WebSA (Web software architecture) approach [15] and further integrated to the so far built functional models.

In this work we focus on the UWE core process depicted in Fig. 1 as a stereotyped UML activity diagram. Models are represented with object nodes and transformations as stereotyped activities (special circular icon). A chain of transformations then defines the control flow.

![Fig. 1. UWE core process for CIM to PIM and PIM to PIM](image)

### 3.2 UWE Metamodel

The metamodel is structured into packages, i.e. requirements, content, navigation, process, presentation and adaptation. It is defined as a “profileable” extension of the UML 2.0 metamodel providing a precise description of the concepts used to model Web applications and their semantics. We restrict ourselves to illustrate the approach by means of an excerpt of the presentation metamodel, for further details on the UWE metamodel see [11] and [12].

![Fig. 2. UWE presentation metamodel](image)
The presentation metamodel defines the modeling elements required to specify the layout for the underlying navigation and process models. A PresentationClass is a special class representing a Web page or part of it and is composed of user interface elements and other presentation classes. UIElements are classes that represent the user interface elements in a Web page. The presentation metamodel is depicted in Fig. 2. Anchors for example represent links in a Web page, and optionally a format expression may be defined for specification of the label that the anchor should have.

OCL class invariants are used to define the well-formedness rules for models, i.e. the static semantics of a model, to ensure that a model is well formed before executing a transformation.

### 3.3 Model to Model Transformations in UWE

Requirement specification is based on UML use cases for the definition of the functionality of a Web application. The design phase consists of constructing a series of models for the content, navigation, process and presentation aspects at a platform independent level. Transformations implement the systematic construction of dependent models by generating default models, which then have to be refined by the designer. The approach uses the ATL transformation language in all phases of the development process.

We illustrate our approach by means of a simple project management system, which is part of the GLOWA Danube system, an environmental decision support system for the water balance of the upper Danube basin. The project management system allows adding, removing, editing and viewing of two types of projects, user projects and validation projects. At CIM level the UWE profile provides different types of use cases for treating browsing and transactions (static and dynamic) functionality. UWE proposes the use of stereotypes «navigation» and «web process», to model navigation and process aspects, respectively. A content model of a Web application is automatically derived from the requirements model by applying a transformation Requirements2Content. The developer can refine the resulting default content model by adding additional classes, attributes, operations, associations etc. Such a refined content model is represented as a UML class diagram (see Fig. 3).

![Fig. 3. Content model of project management system (simplified)]
The navigation model represents a static navigation view of the content and provides entry and exit points to Web processes. Nodes stereotyped as «navigation class» and «process class» like Project and RemoveProject (Fig. 4) represent information from the content model and use case model (requirements). They are generated by the rules ContentClass2NavigationClass and ProcessIntegration respectively. Nodes stereotyped as «menu» allow for the selection of a navigation path.

Links («navigation link» and «process link») specify the navigation paths between nodes. The transformation rule CreateProcessDataAndFlow automatically generates the process data and a draft of the process flow for Web processes. Process flows are represented as UML activity diagram where «user action»s, such as RemoveProjectInput in Fig. 5, are distinguished to support a seamless application of transformation rules. The transformation NavigationAndProcess2Presentation automatically derives a presentation model from the navigation model and the process model. For each node in the navigation model and each process class that represents process data a presentation class is constructed and for each attribute a corresponding presentation class is constructed and for each attribute a corresponding presentation class is constructed and for each attribute a corresponding
tation property is created according to the type of a user interface element. For example a «text» stereotyped class is created for each attribute of type String and a «anchor» stereotyped class is created for an attribute of type URL. An excerpt of the results is shown in Fig. 6.

4 UWE-Based Generation of Web Applications

The main effort of an MDE approach to Web application generation is bridging the gap between the abstractions present in the design models (content, navigation, process, and presentation) and the targeted Web platform. In fact, this gap is quite narrow for UWE content and presentation models: The UML static structure used for describing content in UWE (if we may neglect more arcane modeling concepts like advanced templates, package merging and the like) has a rather direct counterpart in the backing data description techniques of current Web platforms, like relational database models, object-relational mappings, or Java beans. Similarly, an UWE presentation model, being again based on UML static structures, can be seen as a mild abstraction of Web page designs, using, e.g., plain HTML, Dynamic HTML, or Java Server Pages (JSPs). This abstraction gap is comparable for UWE navigation structures using only navigation nodes and access primitives. The situation, however, changes for UWE business process descriptions, which use a workflow notation. In particular, the token-based, Petri net-like interpretation of UML activities and their combination of control and data flow, which is especially well-suited for a declarative data transport, differs notably from the more traditional, control-centric programming language concepts supported by current Web platforms.

Thus, for generating a Web application from an UWE design model, we employ on the one hand a transformational and on the other hand an interpretational approach: Transformation rules are adequate for generating the data model and the presentation layer of a Web application from the UWE content, navigation structure and presentation models, where the differences in modeling and platform abstraction is low. The higher differences in abstraction and formalism apparent in the process models can be more easily overcome interpreting these executable models directly in a virtual machine. For concreteness, we describe the Web application generation process from UWE models by means of a single Web platform, the Spring framework\(^\text{10}\), but, by exchanging the model transformations and adapting the virtual machine, the principal ideas could be easily transferred to other Web technologies like using simply Java Server Pages (JSPs) or, more heavy weightily, ASP.NET.

Spring is a multi-purpose framework based on the Java platform modularly integrating an MVC 2-based\(^\text{11}\) Web framework with facilities for middleware access, persistence, and transaction management. Its decoupling of the model, view, and controller parts directly reflects the general structure of the UWE modeling approach and also supports the complementary use of transformation rules and an execution engine in Web application generation: Minimal requirements are imposed by the Spring framework on the model technology; in fact, any kind of Plain Old Java Ob-

\(^{10}\) Spring Framework, http://www.springframework.org/
jects (POJOs) can be used, and the access to the model from the view or the controller parts only relies on calling get- and set-methods. We illustrate the approach defining transformation rules from an UWE content model into Java beans. The view technology is separated from the model and the controller part by a configuration mechanism provided by Spring; thus technologies like JSPs, Tiles, or Java Server Faces can be employed. We define transformation rules from an UWE presentation model into JSPs. Finally, the controller part provides a hook for deploying a virtual machine for business process interpretation, as it can be customized through any Java class implementing a specific interface from the Spring framework. Configuration data for the virtual machine are generated from the UWE process and navigation model.

An overview of the transformations and the execution engine we describe in the following is given in Fig. 7. The virtual machine for business process execution and the integration into navigation is wrapped into a runtime environment that is built on top of the Spring framework and also encapsulates the management of data and the handling of views.

4.1 Runtime Environment

The structure of the runtime environment is shown in Fig. 8. The Spring framework is configured to use a specific generic controller implementation named MainController. The controller has access to a set of objects with types generated from the content model and one designated root object as entrance point for the application. Model objects are accessed by their get- and set- methods and the operations defined in the content model. Additionally, the controller manages a set of NavigationClassInfo objects which contain information about the navigation structure regarding inheritance between navigation classes and are generated from the navigation model. A set of ProcessActivity objects generated from the process model represents the available Web processes; for each session at most one process can be active at a time.
Views, i.e. Web pages, are not explicitly managed by the runtime environment, only string identifiers are passed. The Spring framework is responsible for resolving these identifiers to actual Web pages that were generated from the presentation model.

The method `handleRequest` handles incoming Web requests by modifying the model and returning a corresponding view. When this method is called, it first checks if a process is active in the current session. If it is, then the execution is delegated to the process runtime environment as detailed in Sect. 4.3. If not, then the next object from the content model that should be presented to the user is resolved by its identifier. Finally, a view identifier is returned and the corresponding Web page is shown to the user.

![Diagram](image)

**Fig. 8.** Runtime environment

### 4.2 Content and Navigation

The transformation of the content model into Java beans is rather straightforward. The following example illustrates the outcome for the transformation rule applied to `ProjectManager`:

```java
public class ProjectManager {
    private List<Project> projects;
    public List<Project> getProjects() {
        return projects;
    }
    public void setProjects(List<Project> projects) {
        this.projects = projects;
    }
    public void removeProject(Project project) {
        // to be implemented manually
    }
}
```

The navigation model does not have to be directly transformed into code because in the transformation of the presentation model the references to elements from the navigation model are resolved so that the generated pages directly access the content.
model. Nevertheless, a minimum knowledge about the navigation model is needed in the runtime environment to handle dynamic navigation. For instance, in the navigation model of Fig. 4 a process link leads from the process class AddProject to the abstract navigation class Project with the two navigation sub classes UserProject and ValidationProject. Thus, when following the link from AddProject to a created project then the presentation class for the navigation subclass for the dynamic content object type should be displayed.

4.3 Process

Because of the complex execution semantics of activities based on token flows we integrate a generic Web process engine into the platform specific runtime environment presented in Sect. 4.1. The basic structure of the Web process engine is given in Fig. 9. A process activity comprises a list of activity nodes and set of activity edges. Activity nodes can hold a token which is either a control token, indicating that a flow of control is currently at a specific node, or an object token which indicates that an object flow is at a specific node. Activity edges represent the possible flow of tokens from one activity node to another. Multiple tokens may be present at different activity nodes at a specific point in time. The method acceptsToken of an activity node or an activity edge is used to query if a specific token would currently be accepted which then could be received by the method receiveToken. An activity has an input parameter node and optionally an output parameter node which serve to hold input and output object tokens.

The control nodes supported by the process engine are decision and merge nodes, join and fork nodes, and final nodes. The object nodes supported are pins representing input and output of actions, activity parameter nodes for the input and output of process activities, central buffer nodes for intermediate buffering of object tokens, and datastore nodes representing a permanent buffer. The implementation of these nodes corresponds to the UML 2.0 specification.

Before starting the execution of a process activity it has to be initialized by calling the method init. This results in initializing all contained activity nodes and placing an object token in the input parameter node as illustrated by the following simplified Java code lines:

```java
public void init(Object inputParameter) {
    // initialize all activity nodes
    for (ActivityNode n : activityNodes)
        n.init();
    // place new object token in input parameter node
    inputParameterNode.receiveToken(new ObjectToken(inputParameter));
    finished = false;
}
```

The complete execution of a process activity comprises the handling of user interactions, like RemoveProjectInput and ConfirmRemoveProjectInput in Fig. 5. Thus, when a process activity contains at least one user interaction then it cannot be executed completely in one step. The method next of a process activity is called from the runtime environment to execute the process activity until the next user interaction is encountered or the process activity has finished its execution. Moreover, either the
The following code lines give an outline to the implementation of the method `next`:

```java
public Object next() {
    // process input requested after last method call
    for (ActivityNode n : activityNodes) {
        if (n.isWaitingForInput()) {
            n.processInput();
            break;
        }
    }
    // token passing loop
    while (true) {
        for (ActivityNode n : activityNodes) {
            n.next();
            // return in case of waiting for user input
            if (n.isWaitingForInput()) {
                return n.getInputObject();
            } else if (n == outputParameterNode) {
                if (n.hasToken()) {
                    return outputParameterNode.getObjectToken().getObject();
                } else if (n instanceof ActivityFinalNode) {
                    return null;
                }
            } else if (n instanceof ActivityFinalNode) {
                return null;
            }
        }
    }
}
```

First the method `processInput` of the first activity node that was waiting for input in the last step is called to process the user input that is now available in the user in-
teraction object. Then all activity nodes are notified to execute their behavior by calling the method next. If a node then indicates that it is waiting for input the method returns with the user interaction object returned by this node. If a token arrives either at an activity output parameter node or at an activity final node the execution of the process activity terminates and the method returns.

4.4 Presentation

We outline the transformation from the presentation model to Java Server Pages. The metamodel of JSPs is shown in Fig. 10. For every user interface element of type X in the presentation metamodel a corresponding ATL transformation rule X2JSP is responsible for the transformation of user interface elements of the given type.

![Fig. 10. JSP metamodel](image)

Each presentation class is mapped to a root element that includes the outer structure of an HTML document. All mappings of user interface elements are included in the body tag.

```atl
rule PresentationClass2JSP {
  from pc : UWE!PresentationClass
to jsp : JSP!Root=
    documentName <- pc.name + '.jsp',
    children <- Sequence{ htmlNode },
    htmlNode : JSP!Element=
      name <- 'html',
      children <- Sequence{ headNode, bodyNode },
    headNode : JSP!Element=
      name <- 'head',
      children <- Sequence{ titleNode },
    titleNode : JSP!Element=
      name <- 'title',
      children <- Sequence{ titleTextNode },
    titleTextNode : JSP!TextNode=
      value <- pc.name,
    bodyNode : JSP!Element=
      name <- 'body',
      children <- Sequence{ pc.ownedAttribute->collect(p | p.type) }
}
```

Each user interface element of type form is mapped to an HTML form element. All contained user interface elements are placed inside the form element. The action attribute points to the target of the corresponding link associated to the form by using a relative page name with the suffix .uwe.

```atl
rule Form2JSP {
  from
```
uie : UWE!Form

to
jsp : JSP!Element(name <- 'form',
    children <- Sequence{ actionAttr, uie.uiElements }),
actionAttr : JSP!Attribute(name <- 'action',
    value <- uie.link.target.name + '.uwe')
}

All remaining UI elements, like anchors, text, images are transformed similarly [12].
The JSP rendering of the presentation class ProjectManager of Fig. 6 is shown in Fig. 11.

![Screenshot of JSP for the presentation class ProjectManager](image)

**Fig. 11.** Screenshot of JSP for the presentation class ProjectManager

5 Related Work

Other methodologies in the Web engineering domain are also introducing model-driven development techniques in their development processes. For example, the Web Software Architecture (WebSA [15]) approach complements other Web design methods, like OO-H and UWE, by providing an additional viewpoint for the architecture of a Web application. Transformations are specified in a proprietary transformation language called UML Profile for Transformations, which is based on QVT.

MIDAS [3] is another model driven approach for Web application development based on the MDA approach. For analysis and design it is based on the content, navigation and presentation models provided by UWE. In contrast to our approach, it relies on object-relational techniques for the implementation of the content aspect and on XML techniques for the implementation of the navigation and presentation aspects. A process aspect is not supported by MIDAS yet.

The Web Markup Language (WebML [5]) is a data-intensive approach that until now does use neither an explicit metamodel nor model transformation languages. The corresponding tool WebRatio internally uses a Document Type Definition (DTD) for storing WebML models and the XML transformation language XSLT for model-to-code transformation. WebML transformation rules are proprietary part of its CASE tool. Schauerhuber et al. present in [17] an approach to semi-automatically transform the WebML DTD specification to a MOF compatible metamodel.
A recent extension of the Object Oriented Web Solution (OOWS [19]) supports business processes by the inclusion of graphical user interfaces elements in their navigation model. The imperative features of QVT, i.e. operational mappings are used as transformation language. In [20] OOWS proposes the use of graph transformations to automate its CIM to PIM transformations. A similar approach is used in W2000 [1]. SHDM [13] and Hera [21] are both methods centered on the Semantic Web. HyperDE – a tool implementing SHDM – is based on Ruby on Rails extended by navigation primitives. Hera instead only applies model-driven engineering for the creation of a model for data integration.

Another interesting model driven approach stems from Muller et al. [16]. In contrast to this work, a heavyweight non-profilable metamodel is used for the hypertext model and the template-based presentation model, nevertheless UML is used for the business model. A language called Xion is used to express constraints and actions. A visual model driven tool called Netsilon supports the whole approach.

6 Conclusions and Future Work

We have presented an MDE approach to the generation of Web applications from UWE design models. On the one hand, model transformation rules in the transformation language ATL translate the UWE content and presentation models into Java beans and JSPs; on the other hand, a virtual machine built on top of the controller of the Spring framework executes the business processes integrated into the navigation structure. These are the first steps towards a “translationist” vision of transformations of platform independent models to platform specific models in UWE.

The combination of a translational and interpretational approach offers a high degree of flexibility to generate Web applications for a broad range of different target technologies. The approach presented in this work is further extended in [12], including a detailed description of computational independent models (CIMs) and platform independent models (PIMs) as well as transformations from CIM to PIM and PIM to PIM, which are also expressed as ATL transformation rules. The ATL transformations of this work are easily transferable to QVT.

Our future work will focus on applying the model transformation approach to other Web applications concerns, such as adaptivity and access control. An aspect-oriented modeling approach is used to model these concerns in UWE and still needs the definition of appropriate transformations. We also plan to analyze the applicability of an MDE approach to Web 2.0 features, e.g. Web services and Rich Internet Applications (RIAs) using AJAX technology in the model-driven development process of UWE.

References


245
MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods

Antonio Vallecillo¹, Nora Koch², Cristina Cachero⁴, Sara Comai³, Piero Fraternali³, Irene Garrigós⁴, Jaime Gómez⁴, Gerti Kappel², Alexander Knapp², Maristella Matera³, Santiago Meliá⁴, Nathalie Moreno¹, Birgit Pröll⁶, Thomas Reiter⁶, Werner Retschitzegger⁶, José E. Rivera¹, Andrea Schauerhuber⁵, Wieland Schwinger⁶, Manuel Wimmer⁵, Gefei Zhang²

¹Universidad de Málaga, Spain
²Ludwig-Maximilians-Universität München, Germany
³Politecnico di Milano, Italy
⁴Universidad de Alicante, Spain
⁵Technical University Vienna, Austria
⁶Johannes Kepler Universität Linz, Austria

mdwenet@pst.ifi.lmu.de

Abstract. Current model-driven Web Engineering approaches (such as OO-H, UWE or WebML) provide a set of methods and supporting tools for a systematic design and development of Web applications. Each method addresses different concerns using separate models (content, navigation, presentation, business logic, etc.), and provide model compilers that produce most of the logic and Web pages of the application from these models. However, these proposals also have some limitations, especially for exchanging models or representing further modeling concerns, such as architectural styles, technology independence, or distribution. A possible solution to these issues is provided by making model-driven Web Engineering proposals interoperate, being able to complement each other, and to exchange models between the different tools. MDWEnet is a recent initiative started by a small group of researchers working on model-driven Web Engineering (MDWE). Its goal is to improve current practices and tools for the model-driven development of Web applications for better interoperability. The proposal is based on the strengths of current model-driven Web Engineering methods, and the existing experience and knowledge in the field. This paper presents the background, motivation, scope, and objectives of MDWEnet. Furthermore, it reports on the MDWEnet results and achievements so far, and its future plan of actions.

1 Introduction

Model-Driven Engineering (MDE) advocates the use of models and model transformations as the key features in all phases of software development, from system specification and analysis over design to implementation and testing. Each model usually
addresses one concern, independently of the rest of the issues involved in the construction of the system. Thus, the basic functionality of the system can be separated from its final implementation; the business logic can be separated from the underlying platform technology, etc. The transformations between models enable the automated implementation of a system right from the different models defined for it.

Web Engineering is a specific domain in which model-driven software development can be successfully applied [1]. Existing model-driven Web Engineering approaches (such as OO-H [2], UWE [3] or WebML [4]) already provide a set of suitable methods and tools for the design and development of most kinds of Web applications. They address different concerns using separate models (navigation, presentation, business logic, etc.) and come with model compilers that produce most of the application’s Web pages and logic based on these models. However, most of these Web Engineering proposals do not fully exploit all the potential benefits of MDE, such as complete platform independence, or tool interoperability. In addition, these proposals also have some limitations, especially when it comes to exchanging models or expressing further concerns, such as architectural styles or distribution.

Recently, the OMG’s Model-Driven Architecture (MDA) initiative [5] has introduced a new approach for organizing the design of an application into different models so portability, interoperability and reusability can be obtained through architectural separation of concerns. MDA covers a wide spectrum of topics and issues ranging from MOF-based metamodels to UML profiles, model transformations and modeling languages.

However, the effective integration with the already existing model-driven Web Engineering approaches has been only partially achieved. The most interesting issue is the interoperability of models and artifacts designed using the different existing development methods to enable the use of synergies. The vision is, at the end of a long way, to count on either one unified method based on the strengths of the different methods, or interoperability bridges (transformations) between the individual models and tools that would allow their seamless integration for building Web applications.

Many groups of the Web Engineering community share these objectives. Lively discussions took place at both Model-Driven Web Engineering (MDWE) workshops in Sydney (2005) and Menlo Park (2006). A small number of groups decided to reinforce discussions on workshops with a set of planned activities in order to get concrete solutions to the current problem of interoperability of model-driven Web Engineering approaches. The initiative is called MDWEnet and started its activities in December 2006. This paper provides an overview of the motivation and background of this initiative (Section 2), its scope and objectives (Section 3), activities (Section 4), and future plans (Section 5).

2 Background and Motivation

The growing interest in Model-Driven Web Engineering has produced quite a significant number of results, which have materialized into a concrete set of MDWE approaches. As mentioned above, they provide suitable methods and tools for the design and development of Web applications, but they also present some limitations. So far,
each group is mainly working on progressively improving their own proposals in an independent manner, with the exception of a couple of bilateral collaborations. One alternative solution is provided by the possibility of making Web proposals interoperate, being able to complement each other, and to exchange models between the different tools. This is precisely one of the goals of MDWEnet.

The authors of this paper met for the first time in Munich in December 2006, with the objective of coordinating the current efforts being carried out by individual groups in the field of MDWE. They are members from five of the groups that work on these topics, including the UWE, OO-H and WebML teams from the Universities of Munich, Alicante and Politecnico di Milano, respectively. The other two groups are from the University of Malaga, and from a joint cooperation between the Technical University of Vienna and the Johannes Kepler University of Linz, contributing with their knowledge on frameworks, metamodels and model transformations in the Web field \[6,7,8\]. The intention is to harmonize their efforts in order to be more effective, to avoid duplicated work, and to align their targets and goals. The plan was to start with a small number of groups first, and then to open to the rest of the MDWE community as soon as the first results were tangible and could be shown.

Several discussions took place during the meeting, most of them being representative examples of the topics and issues of current interest to the MDWE community. First, the current activities and work in progress of each group were presented. Then, a SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis of the situation of MDWE in the fields of MDE and Web Engineering was conducted, to provide a clear picture of the context and the current position from where to start. The following sections describe these issues, because they not only can be of help to the MDWEnet group, but can also be interesting to the whole MDWE community.

### 2.1 Work in Progress

The following list shows the topics and issues that each individual group has recently addressed:

- **a) Addressing new concerns in Web application development**
  - Software architecture (OO-H)
  - Personalization (OO-H), Adaptation (WebML)
  - Workflows (UWE, WebML)
  - Integration with external Web Services (WebML)
  - Requirements (UWE)

- **b) Quality evaluation**
  - Effort estimation (OO-H, WebML)
  - Usability logs for analysing usage patterns and validation of navigational designs (WebML)

- **c) Metamodel profiling and integration**
  - Definition of a global framework (Málaga)
  - WebML profiles (various)
  - Metamodel integration (Wien/Linz)
d) Other
- Semantic annotations (OO-H)
- Automatic client-side code generation (WebML)
- Test derivation of applications (WebML)
- Analysis, validation and verification of models (WebML, UWE)
- Use of Aspect-Oriented Software Development techniques (e.g., for adaptation/access control) (UWE, Wien/Linz)

2.2 SWOT Analysis

A SWOT analysis was conducted to gain a better understanding on the Strengths, Weaknesses, Opportunities and Threads of current MDWE practices and approaches. The results are very illustrative, and show a field with plenty of possibilities and opportunities to grow and provide interesting benefits to the Web Engineering community.

a) Strengths
- Tool-supported methods…that work in practice!
  Significant improvements on productivity
- Tested and validated by real industrial usage
  Large companies
  Many projects (both privately and publicly funded)
- Wide knowledge and experience in Web Engineering
- Many groups working on interesting and useful extensions (see Sect. 2.1)

b) Weaknesses
- Of those approaches not using OMG standards
  Use of proprietary notations (many customers don’t like them)
  Tools not aligned with MDA (yet)
- Of those using UML
  Tool support (for modeling and code generation)
- No interoperability of models and tools between individual proposals
  No reusability of efforts and developments
  No “core competencies” approach possible
- Current Web modeling languages…
  …are model-driven to a limited extent (e.g., the majority of approaches have not defined their metamodels, do not rely on model transformations, etc.)
  …partly provide concepts for modeling customization but no comprehensive support
- Customization of functionality cannot be captured separately but is scattered across all levels of a Web application model
c) Opportunities

- Web Engineering is a domain where MDE ideas can be successfully applied
- There is a current need for MDWE solutions in industry
  - Real interest from customers
  - Research funds (National and European)
- There is an interest in academia
  - Journals, conferences
- MDE and MDA are fashionable now
  - Claimed to be supported by everybody (OMG, IBM, Microsoft, Customers, etc.)
  - Model transformation languages are becoming mature
- There is a group of people willing to co-operate to make it work
  - MDWEnet is a concrete example
- Use the repositories of previous projects for conducting empirical studies on performance, quality, etc.

d) Threats

- MDE/MDA fails to deliver because of
  - No tool support
  - Customer dissatisfaction or frustration (probably due to too high expectations)
- We fail to deliver because of
  - Result is worse than individual proposals, or
  - Resulting method, techniques and/or notation are too complex,
  - Learning is too difficult, or usability is not good enough
  - No real applications (very complex) can be built
- Real goals not addressed; they are
  - Too academic, or
  - Too pragmatic

3 Scope and objectives of MDWEnet

The scope of the MDWEnet initiative is the model-driven development of Web applications, using different methods and tools, while ensuring the interoperability of their artifacts and models.

The overall objective is to improve current practices and tools for the model-driven development of Web applications, by making use of the strengths of current model-driven Web Engineering methods, and the existing experience and knowledge in the field.

The way in which we decided to reach this goal is by investigating the interoperability of model-driven Web Engineering methods, i.e., by trying to explore how Web proposals could interoperate, be able to complement each other, and exchange models between the different tools.
Two clear phases in the process were distinguished: (1) proof of concept and validation; and (2) application of the interoperability approach.

The first phase is focused on investigating how this interoperability can be achieved at a basic level (i.e., over the fundamental set of elements and functionality that any MDWE method should cover), and on its validation for three MDWE methods: OO-H [2], UWE [3] and WebML [4]. This phase is based on an incremental and iterative process, starting from a very small set of features and functionality that the different methods should deal with, which are progressively extended until the basic functionality offered by any MDWE approach is covered.

Once we achieve the required interoperability between the individual methods at that basic level, the second phase will use a set of representative Web applications to progressively extend these modeling elements and features, being able to deal with both static and dynamic aspects of Web application design.

4 Activities

During the workshop different possibilities to achieve the objectives were discussed, as already mentioned, focusing on two options: to use or not to use a common metamodel. In order to be able to define precise actions, the MDWEnet group had to make a set of decisions related to the technologies and tools to be used for implementing the actions. Some of these decisions were not easy to make, as described below. A plan of concrete actions was defined, relying on a strong commitment of the teams of all groups.

4.1 Possibilities

In general, there are many ways to achieve these goals, especially in the MDE field—which is neither fully mature nor well established yet. For instance, we had the following choices for tackling the problem of the interoperability between different MDWE approaches.

- Taking the best of each approach and try to define an integrated approach (in a similar way in which the UML was originally defined)?
- Developing a common metamodel?
- Preserving the nature of each web method and try to concentrate the efforts to process transformations between models?

We decided to initially explore two possibilities, and, once we have some concrete results, to look back and decide based on the pros and cons of each one. These possibilities, together with their advantages and disadvantages (a priori) are as follows.

Option 1: Definition of a metamodel for each individual approach and of the transformations between the different metamodels.

- Assumptions
  - There exists no common metamodel, or
– No agreement is reached w.r.t. a common metamodel, or
– The common metamodel is not expressive enough, or
– Transformations are possible between all individual metamodels

• Benefits/advantages
  – Individuality is respected
  – Tools are readily available
  – Zoos (model repositories) can be “easily” built and maintained to share models

• Disadvantages
  – Integration and interoperation are much more difficult
  – Sharing tools is complicated
  – Too many transformations required \([n(n–1)]\)

Option 2: Definition of a common metamodel

• Assumptions
  – There exists a common metamodel
  – An agreement is reached w.r.t. such a common metamodel
  – The common metamodel is expressive enough
  – Transformations are possible to/from all individual metamodels

• Benefits/advantages
  – Integration and interoperation are easier
  – Sharing tools is possible
  – Core competencies (presentation/information/tools/…)
  – Less transformations between metamodels \([2n]\)

• Disadvantages
  – Individuality is somehow lost
  – Too many assumptions
  – Interoperability conflicts between different proposals

Of course, none of these options is free from problems. For example, should the common metamodel be (a) just the basics of MDWE; (b) the intersection of the metamodels of all MDWE proposals; or (c) the union of all metamodels?

Regarding the notation to express the metamodels, should we use MOF, eMOF, Ecore, KM3, or other metamodeling languages?

This leads to a more delicate question, regarding the MDE approach to use. Should we go try to be compatible with the OMG approach (which means using MOF, UML, QVT, etc), the Microsoft approach, or other (e.g., use AMMA and the ATLAS way)?

This has also to do with the choice of the modeling tools, since they do not interoperate at present. This is another important decision, since the only way to be able to seamlessly exchange models and artefacts is by sharing a common modeling tool (such as Enterprise Architect, MagicDraw, etc.). And the same is true for the model transformation language and tool to use: QVT (Together), graph-based (AGG, VIATRA, ATOM3), or other (e.g., ATL).
4.2 Decisions

As aforementioned, we decided to explore the two options above: (1) to define and use individual metamodels and transformations between them; and (2) to define a common metamodel and transformations to/from the metamodels of the different proposals. The common metamodel will be defined as the union of all metamodels. The metamodeling language will be Ecore, and the MDE approach will be based on the ATLAS group initiative, i.e., using the ATL as model transformation language. For drawing models we agreed to use MagicDraw as modeling tool.

4.3 Plan of actions

Based on these decisions, a concrete plan of actions was set up. It was organized into two phases, the first one running for 6 months. The actions to be developed during the first phase focus on the definition of a common metamodel, on the specification of the metamodels of the three initial proposals (UWE, OO-H and WebML) and on the transformations between these metamodels.

In addition, the actions should achieve the preparation of a survey of existing MDWE approaches and a “map” of communities that work on topics closely related to Model-Driven Web Engineering. A second phase would build on the result of the first one, and would consist of the definition of a Web Engineering modeling ontology, the evaluation of existing Web Engineering modeling tool environments and their capabilities for integration. Another goal is to cooperate in teaching and research, e.g., sharing teaching material and the acquisition of funding for joint projects.

4.4 Results so far

Although there is still a long way to go, we already count on a set of results, which could be of interest to the MDWE community.

The first one is a Wiki web, used by the group as a collaborative platform. The Wiki allows the exchange of information, documents, models, and tools, as well as the development of joint work on the material. It fulfills also the role of a repository of all kind of interesting information on model-driven Web Engineering topics.

The Wiki also contains the results that the actions have produced. In particular, it includes a collection of information on funding opportunities, the specification of the common metamodel, core metamodels of OO-H, UWE and WebML and a set of example model problems.

5 Future plans

The current activities are limited to the proof of concept of a first approach of interoperability of three methods OO-H, UWE and WebML. We also restricted the number of issues the different methods should manage to a small set of basic features of Web applications.
We plan to extend the current state with further modeling elements of the selected methods in order to cover all static and dynamic model-driven aspects of Web applications. Further methods and experimental material for volunteers to conduct experiments on external quality of Web applications developed with Web Engineering methods will be provided as well. At a long term plan the vision of a fully integrated environment where modeling and generation of Web applications using any method would be possible.

References


On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges

Manuel Wimmer¹,‡, Andrea Schauerhuber²*, Wieland Schwinger³,‡, Horst Kargl¹,‡
¹Business Informatics Group
Vienna University of Technology
{wimmer, kargl}@big.tuwien.ac.at
²Women’s Postgraduate College for Internet Technologies
Vienna University of Technology
schauerhuber@wit.tuwien.ac.at
³Department of Telecooperation
Johannes Kepler University Linz
wieland.schwinger@jku.ac.at

Abstract. The Unified Modeling Language (UML) is considered as the lingua franca in software engineering. Despite various web modeling languages having emerged in the past decade, in the field of web engineering a pendant to UML cannot be found yet. In the light of this “method war” the question arises if a unification of the existing web modeling languages can be successfully applied in the style of UML’s development. In such a unification effort we defer the task of designing a “Unified Web Modeling Language”. Instead, we first aim at integrating three prominent representatives of the web modeling field, namely WebML, UWE, and OO-H, in order to gain a detailed understanding of their commonalities and differences as well as to identify the common concepts used in web modeling. This integration is based on specifying transformation rules allowing the transformation of WebML, UWE, and OO-H models into any other of the three languages, respectively. To this end, a major contribution of this work is the languages’ definitions made explicit in terms of metamodels, a prerequisite for model-driven web engineering for each approach. Furthermore, the transformation rules defined between these metamodels - besides representing a step towards unification - also enable interoperability through model exchange.

Keywords: Web Modeling, Model Integration, Common Metamodel for Web Modeling, Model-Driven Web Engineering

‡ This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) and FFG under grant FIT-IT-810806.
* This work has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.
1 Introduction

In the past decade various modeling approaches have emerged in the research field of web engineering including WebML [7], UWE [13], W2000 [1], OOHDM [26], OO-H [10], WSDM [8], and OOWS [25]. Each of those model-based approaches follows the similar goal of counteracting a technology-driven and ad hoc development of web applications. Beyond this, we notice similar and simultaneous extensions to the individual web modeling approaches, e.g., for supporting context-aware web applications [2, 6, 9], business process modeling [4, 14], and for exploiting all benefits of model-driven web engineering (MDWE) [15, 18]. The current situation somewhat resembles the object-oriented modeling “method war” of the 90ies. A situation from which after a unification process the UML [24] eventually has become the *lingua franca* in software engineering. In the light of the current “method war” in the research field of web engineering (cf. Figure 1) the question arises if a unification of the existing web modeling approaches can be successfully applied as it was achieved for the UML.

![Web Modeling Languages History](image)

**Figure 1:** Web Modeling Languages History, based on [28]

In the MDWeNet initiative [16], which has recently started by a small group of researchers working on MDWE, this and further questions are tackled. More specifically, the initiative’s goal is to improve interoperability between current web modeling approaches as well as their tools for the model-driven development of web applications.

As a prerequisite for unification a common agreement on the most important web modeling concepts is essential. This agreement can only be gained when investigating the concepts used in existing web modeling languages and fully understanding the languages’ commonalities and differences. In the MDWeNet initiative, we therefore defer the task of designing a “Unified Web Modeling Language”. Instead, we first aim at integrating three prominent representatives of the web modeling field, namely WebML, UWE, and OO-H, since they are well elaborated and documented as well as
supported by modeling tools. This integration is based on specifying and implementing transformation rules allowing the transformation of WebML, UWE, and OO-H models into any other of the three languages, respectively. This way a detailed understanding of the common concepts used in web modeling can be obtained as well as their different realizations in the three selected languages. On the basis of this integration task the definition of a common metamodel for web modeling can be achieved in the future.

Consequently, the major contribution of this work is a step towards identifying the common concepts in web modeling by first defining transformations between different modeling languages. We present the general integration approach as well as first results on the integration of WebML and OO-H.

Besides representing an important step towards unification, the transformation rules also enable model exchange between the three different languages. For defining the transformation rules, the languages’ definitions had to be made explicit in terms of metamodels, which in turn represent a prerequisite for enabling model-driven web engineering for each individual approach. On the basis of tool adapters the models’ representation within the approaches’ tools could be translated into instances of these metamodels and vice versa thus also insuring interoperability. Furthermore, it will be possible to exploit the different strengths of each web modeling approach, e.g., code generation facilities for different platforms such as J2EE in WebML’s WebRatio1 tool and PHP in OO-H’s tool VisualWade2.

In the remainder of the paper, we discuss our methodology for integrating existing web modeling languages in Section 2. We elaborate on preliminary results of the integration task with respect to WebML and OO-H in Section 3 and provide our lessons learned in Section 4. Finally the paper is concluded with a discussion on future challenges in integrating as well as unifying web modeling languages.

2 Integration Methodology used in MDWEnet

In this section we discuss the general methodology used for the integration of WebML, OO-H, and UWE. We first explain why integration on the basis of already existing language artifacts is not possible. Second, we outline a model-based integration framework, and third, we discuss how to obtain the most important prerequisite for integration – the metamodels for the three web modeling languages.

Why is the integration on the basis of existing language artifacts not possible?

In Table 1 we present an overview of the formalisms used for defining WebML, OO-H, and UWE as well as the approaches’ model storage formats. When looking at the languages’ definitions, one can easily identify that each language is specified in a different formalism, even in different technological spaces [17], which a-priori prevents the comparability of the languages as well as model exchange.

For the integration of modeling languages in general and for web modeling languages in particular, the first requirement is that the languages are defined with the

1 www.webratio.com
2 www.visualwade.com
same meta-language. This enables to overcome syntactical heterogeneities and to compare the language concepts in the same formalism. Furthermore, defining languages with the same formalism also allows expressing their model instances in the same formalism which further fosters comparability of the languages’ concepts and beyond allows the uniform processing of the models, e.g., their visualization or transformation.

Table 1. Differences concerning Language Definition and Model Storage.

<table>
<thead>
<tr>
<th>Language Definition</th>
<th>Model Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebML</td>
<td>WebRatio, DTD(^3)</td>
</tr>
<tr>
<td></td>
<td>XML documents</td>
</tr>
<tr>
<td>OO-H</td>
<td>VisualWade, Rational Rose Model</td>
</tr>
<tr>
<td></td>
<td>Proprietary format</td>
</tr>
<tr>
<td>UWE</td>
<td>ArgoUWE, UML Profile</td>
</tr>
<tr>
<td></td>
<td>XMI</td>
</tr>
</tbody>
</table>

Consequently, it seems necessary to split up the integration process in order to tackle two distinct integration concerns, namely syntactical and semantical integration: In the first step, i.e. the syntactic integration, the different formats used by WebML, UWE and OO-H are aligned towards one common integration format. For example, a WebML model represented in terms of an XML document has to be translated into this common integration format. The second step, i.e. the semantical integration step, covers the transformation of a model from one language into another one, e.g. from WebML to OO-H, while preserving the semantics of the input model within the output model. This transformation is based on transformation rules which require the input models to be available in the common integration format.

How to use model-based techniques for integration purposes?

We decided to apply a model-based approach and use techniques and technologies which have emerged with the rise of Model Driven Engineering (MDE) [3]. MDE mainly propagates two techniques which are relevant for integration purposes: (1) metamodels for defining the concepts of modeling languages, and (2) model transformations. Model transformations in the context of MDE can be divided into vertical model transformations and horizontal model transformations [19]. While the first kind concerns transformations between different abstraction levels, e.g., continuously introducing more details that finally are necessary for code generation purposes, e.g., for model refactoring. Consequently, in this work we rely on horizontal model transformations.

In Figure 2, we present our model-based integration framework, which is based on the tool integration pattern of Karsai et al. [12]. The framework is built upon open-source technologies for MDE, which have been developed under the hood of the Eclipse project. In particular, we are using the Eclipse Modeling Framework (EMF) [5], as a model repository for a common syntactic integration format and EMF’s Ecore, i.e. an implementation of the Meta Object Facility (MOF) standard [21], as the meta-language for defining the metamodels for WebML, OO-H, and UWE.

---

\(^3\) Recently, two different proposals for a WebML metamodel have been published in parallel [20, 27].
Furthermore, we employ ATL [11] as model transformation language to implement the transformation rules and finally, the ATL engine for actually executing the transformation. In Figure 2, we also sketch the model-based integration process of WebML (WebRatio), OO-H (VisualWade), and UWE (ArgoUWE), which is described more detailed in the following:

1) **Syntactic Integration.** On the basis of tool adapters for bridging the native model storage formats of the approaches’ tools towards the EMF models can be integrated syntactically. Thus, realizing import functionality the tool adapters have to parse the models in their native format and generate an XMI [22] version for the EMF. In addition, the tool adapters also must be capable of exporting the models by transforming them into the tools’ native format.

2) **Semantic Integration.** After the syntactic integration, the user can focus on the correspondences between modeling concepts of different languages. This is done by relating the metamodel elements and implementing the integration knowledge in terms of ATL model transformation rules.

![Figure 2: Model-based Integration Framework](image)

3) **Definition of a Common Metamodel for Web Modeling.** The top of Figure 2, illustrates the goal of MDWEnet, i.e., a unification of existing web modeling languages in terms of a common metamodel for web modeling. By defining the metamodels for WebML, OO-H, and UWE, as well as working out the integration knowledge in a first step, we hope that the creation of such a common metamodel is easier to achieve afterwards. For the future, the common metamodel for web modeling can serve as a pivot model and thus lowering the integration effort drastically.

4) **Execution of the Transformations.** The model transformation rules then can be executed in a model transformation engine. More specifically, the ATL
engine reads an input model, e.g. a WebML model, and generates an output model, e.g. an OO-H model, according to the transformation rules. Subsequently, the generated models can be exported via the specific tool adapter to the proprietary tool.

What’s missing for a model-based integration and how to close the gap?

As a key-prerequisite for a model-based integration, the metamodels for WebML, OO-H, and UWE must be available, which currently, however, is not the case. Within the MDWEnet initiative, we have decided to use a top-down approach for building the individual metamodels by starting with a focused set of requirements which are specific to the web modeling domain [30]. This approach has the advantage that we can concentrate on the core modeling constructs of the web modeling domain supported by the addressed approaches instead of focusing on a huge amount of concepts available in the individual approaches and implemented in their tools. Following this top-down approach, a set of modeling requirements for the core of web modeling were defined each focusing on a specific modeling problem. In the following, these requirements are briefly explained and categorized into requirements for content modeling, hypertext modeling, and content management modeling.

Layer 0 – Content Modeling. This layer is required to express domain objects and their properties on which the web application is built upon.
Example: Class Student with attribute name, age, and a relationship to the class Professor.

Layer 1 – Hypertext Modeling. This layer covers the requirements for web applications that allow navigation among the hypertext nodes and publish within a node the content extracted from domain objects (cf. Layer 0), possibly based on input provided by the user. The following four cases are subsumed by Layer 1:

- Global Navigation: This case requires a starting point in the web application, i.e. a home page, and subsequently, a navigation mechanism for moving to another page of the hypertext.
- Content Publication: This case requires a page, which publishes a list of domain objects and displays for each object a set of attribute values.
- Parametric Content Publication: This case requires a page, which publishes a list of domain objects each having attached a navigation mechanism, e.g., a button, an anchor. This mechanism shall allow the user to navigate to the details of the object.
- Parametric Content Publication with Explicit Parameter Mapping: This case requires one page, which contains an input form with various input fields. The user inputs are used for computing a set of domain objects. Thereby, the attribute values of the objects need to satisfy a logical condition including as terms the input provided by the user.

Layer 2 – Content Management Modeling. This layer covers the requirements for web applications that allow the user to trigger operations for updating the domain objects and their relationships (cf. Layer 0).
Example: Create a new instance of type Student. Update the age value of the instance of type Student where name='Michael Smith'.

The definition of metamodels is of course an art on its own and can be approached in different ways. For the purpose of this work it was decided to employ an example-
based approach by a process of obtaining a metamodel from the aforementioned requirements as follows [16]: One or more concrete modeling examples were derived from the requirements specification and modeled in the respective modeling language within each approach’s accompanying tool. The code generation facilities of each tool were then used to find out if the examples modeled were semantically identical, i.e., the generated applications should work in the same way. From these models the language concepts which have been used were identified, as well as how these concepts were related to each other. Consequently, this information is then defined in a corresponding metamodel. These metamodels should allow expressing the same models as within the approaches’ tools, meaning the same information must be expressible in the models.

3 Preliminary Results

In this section we present our preliminary results. First, we briefly discuss the modeling examples realizing the MDWEnet’s modeling requirements for web modeling and provide the resulting metamodels in Section 3.1. In order to illustrate how, on basis of those metamodels, the integration is realized with ATL in Section 3.2 we then present excerpts of the set of ATL transformation rules that have been defined for the metamodels.

3.1 Derived Metamodels

Our first task after the MDWEnet’s modeling requirements for web modeling have been agreed on has been the derivation of concrete modeling examples realizing these requirements specifications. Inspired by previous examples in the web modeling domain, we are using excerpts of the often referred to album store running example [6], which covers all the aforementioned requirements. After defining the modeling examples, each of them was modeled within the approaches’ tools, i.e., WebRatio, VisualWade, and ArgoUWE, respectively. Furthermore, we used the code generation facilities to compare the behavior of the models by executing the generated web applications.

On the basis of the modeling examples, each expressed in WebML, OO-H, and UWE, we identified the language concepts used in the individual examples and obtained first versions of the metamodels for WebML as well as for OO-H. The metamodel for UWE is currently under preparation. Beyond, we have grouped the metamodels’ elements into packages which directly correspond to the layers of the modeling requirements presented in Section 2. In the following, the class structures of the metamodels for WebML and OO-H are presented and briefly explained. For more detailed versions of the metamodels the reader is referred to [30].

WebML Metamodel. In Figure 3, we present the resulting WebML metamodel, i.e., its packages, classes and their interrelationships. While the Structure package and ContentManagement package correspond to the Layer 0 and Layer 2 of the modeling requirements, respectively, for Layer 1 two packages have been defined, namely Hypertext and HypertextOrganization.
The Content package contains modeling concepts that allow modeling the content layer of a web application. Since WebML’s content model is based on the ER-model, it supports ER modeling concepts: An Entity represents a description of common features, i.e., Attributes, of a set of objects. Entities that are associated with each other are connected by Relationships. Unlike UML class diagrams, ER diagrams model structural features, only.

The ContentManagement package contains modeling concepts that allow the modification of data from the content layer. The specific ContentManagementUnits are able to create, modify, and delete Entities (cf. EntityManagementUnit) as well as establish or delete Relationships between Entities from the content layer (cf. RelationshipManagementUnit).

In contrast, the hypertext layer represents a view on the content layer of a web application, only. The Hypertext package summarizes ContentUnits, used, for example, to display information from the content layer which may be connected by Links in a certain way.

The HypertextOrganization package defines the Page modeling concept which is used to organize and structure information from the content layer, e.g., ContentUnits from the Hypertext package, SiteViews group Pages as well as operations on data from the content layer, e.g., OperationUnits from the ContentManagement package. More specifically, SiteViews represent groups of pages devoted to fulfilling the requirements of one or more user groups.

OO-H Metamodel. The class structure of the resulting OO-H metamodel is presented in Figure 4. Similar to the WebML metamodel, the Layer 0 and Layer 2
The OO-H metamodel, i.e., the Content package and Service package, respectively. Concerning Layer 1, two packages have been defined, however, namely the Navigation and Presentation packages.

In the Content package, OO-H’s content model is based on the UML class diagram: A Class represents a description of common structural and behavioral features, e.g., Attributes and Operations, respectively. Classes can be connected with each other via Associations.

The Service package contains the modeling concept ServiceNode that allows the execution of arbitrary operations defined at the content layer. The modeling concept ServiceLink is needed to connect NavigationalNodes with ServiceNodes, and in addition, to transport information in terms of arguments from NavigationalNodes to Operations.

The Navigation package represents a view on the content layer of a web application. In the Navigation package two types of NavigationalNodes can be distinguished, namely ClassNodes displaying information from the content layer, and Collections providing additional information such as navigation menus. Both types have in common that they may be connected by Links. OCLExpressions attached to Links either filter certain objects which should be displayed at the target NavigationalNode or are used as preconditions that must be assured to access the target NavigationalNode.

The Presentation package defines the Page modeling concept which is used to organize and structure the NavigationalNodes of the navigation layer.
3.2 Model Transformations Using ATL

Following we discuss one representative example for the commonalities between WebML and OO-H, in order to exemplify how integration is achieved on the basis of metamodels and model transformation.

As already mentioned, ATL was used as model transformation language, which is a uni-directional, rule-based transformation language. For a full integration, consequently, transformation rules have to be specified for both directions, e.g. from WebML to OO-H and vice versa. An ATL rule consists of a query part (from keyword), which collects the relevant source model elements, and a generation part (to keyword) creating the target model elements.

In Figure 5 (a) we illustrate the semantic correspondences between WebML and OO-H metamodel elements and present two ATL rules implementing the transformation from WebML to OO-H in Figure 5 (b).

1. Rule Entity_2_Class is responsible for transforming each Entity of the WebML model into a Class in OO-H.
2. Rule DisplayUnit_2_ClassNode is responsible for transforming each instance of the concrete subclasses of DisplayUnit into ClassNodes.
3. This minimal example already shows some advantages of using ATL in contrast to using a general-purpose programming language. When executing ATL rules, a “trace model” is created transparently, which saves how instances are transformed. In our example the ATL engine traces which Class instance is generated for an Entity instance. Therefore, it is possible to retrieve the Class instance for the referenced Entity, which allows for the simple statement cN.displayedClass <- dU.displayedEntity.

4 Lessons Learned

Following, we summarize our lessons learned concerning the integration of WebML and OO-H. In general, the integration of WebML and OO-H has turned out to be straight-forward for the most part. At least for the core concepts of web modeling, which have been the focus of the MDWEnet initiative, there exist many commonalities between the two languages. Since the chosen modeling examples
could be realized in each language the languages can be considered to have “equal” expressivity with respect to the defined core requirements. Nevertheless, we also faced differences between the languages, which aggravated the integration. When integrating languages based on their metamodels, further information, which often are not covered by the metamodels, must be incorporated into the transformation rules. This kind of information is on the one hand incorporated into the code generator and on the other hand defined by the frameworks for which code is generated. Some of these differences and the complexity they introduced during integration are explained following the structure of the modeling requirements layers. Nevertheless, from our current experiences we are able to conclude that the differences can be eliminated within the transformations rules. Due to space restrictions and readability reasons we explain the transformation rules textually and refer the reader to [30] for detailed information on the ATL code.

4.1 Content Modeling (Layer 0)

As can be seen in Figure 1, WebML and OO-H have different origins. WebML is based on the ER-model, which is typically used in the context of modeling database schemas. In contrast, OO-H has emerged from an object-oriented background. Consequently, in WebML each Entity has a set of operations which are “implicitly” available and need not be defined by the modeler, i.e., WebML’s ContentManagementUnits actually represent a data manipulation language (DML). These operations include typical create, update, and delete operations as well as operations for linking Entities (cf. Table 2). In contrast, in OO-H there are some predefined operation types available, which have to be explicitly defined for each Class by the modeler (cf. Table 2). Thus, when transforming WebML Entities in OO-H Classes, the default operations must be created for each corresponding Class, in order to ensure that OO-H’s ServiceNodes can execute them.

Table 2: Comparison of Object Operations between WebML and OO-H.

<table>
<thead>
<tr>
<th>WebML Content Management Units</th>
<th>OO-H Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateUnit</td>
<td>Constructor()</td>
</tr>
<tr>
<td>ModifyUnit</td>
<td>Modifier()</td>
</tr>
<tr>
<td>DeleteUnit</td>
<td>Destructor()</td>
</tr>
<tr>
<td>ConnectUnit</td>
<td>Relationer()</td>
</tr>
<tr>
<td>DisconnectUnit</td>
<td>Unrelationer()</td>
</tr>
</tbody>
</table>

Example: Figure 6 (a) shows an excerpt of the content model of the album store example. In Figure 6 (b) we depict the corresponding OO-H content model that needs to be generated by the transformation rules. For each Entity in the content model of WebML an OO-H Class is generated. Besides transforming the Entities’ Attributes, in OO-H the Constructor(), Destructor(), and Modifier() operations must be defined for the Class as well. Likewise for each Relationship of an Entity the Relationer() and Unrelationer() operations have to be generated for the corresponding Class in the OO-H content model.
4.2 Hypertext Modeling (Layer 1)

Generally speaking, one could say that WebML is the more explicit language compared to OO-H, i.e., in the way that there are much more language concepts used. In particular, this is the case for hypertext modeling where OO-H uses a minimal set of concepts, which are refined with OCLExpressions, i.e., preconditions and filters, for Links. In contrast to WebML, where various types of ContentUnits are available, OO-H uses the concepts ClassNode and Collection, only. The actual content and behavior of ClassNodes is defined by their incoming Links. Furthermore, for parameter passing WebML offers LinkParameters with explicit source and target parameter bindings, while in OO-H this is again expressed by OCLExpressions.

Besides the difference in the number of explicit concepts, WebML and OO-H both use their own selector language for computing the content to be displayed. While in OO-H the Object Constraint Language (OCL) [23] has been reused and extended, WebML’s selector language is defined within the metamodel as well as based on the concepts of Selector and SelectorCondition [30]. However, in the current version of the OO-H metamodel, the modified grammar for the OCL is not yet covered as it is done for the WebML selector language in the WebML metamodel. Thus, currently the OCL statements are hard-coded in the transformations rules as ordinary Strings. Incorporating the OCL grammar into the OO-H metamodel and the refinement of the model transformations in order to define the OCL statements as model elements is subject to future work. In the following, an example illustrating these differences between WebML and OO-H is given.

**Example:** A search scenario is given, where in the first page the user provides input, i.e., a certain year, for searching the set of albums. Figure 7 (a) shows the example modeled with WebML⁴, where the EntryUnit AlbumSearch with a Field named ‘from’ represents the input form. The Link to the IndexUnit AlbumResults carries the user input in terms of a parameter. Therefore, a LinkParameter is assigned to the Link, which has as LinkParameterSource the input Field and as LinkParameterTarget the SelectorCondition of the AlbumResults IndexUnit. This SelectorCondition computes the subset of all albums where the input value of the user equals the value of the year attribute. In Figure 7 (b), the same information is modeled with OO-H, where a separate concept for the information that is transported via Links is not available. More specifically, the search scenario can be modeled with a Collection AlbumSearch and a Link to the ClassNode AlbumResults. The Link

---

⁴ Please note that ellipse-shaped legends are not part of WebML’s notation.
contains a filter OCLExpression `dst.year = ?`, with the question mark standing for the user’s input value and `dst.year` meaning the ‘`year`’ Attribute of the `Album` Class.

![WebML model](image)

![OO-H model](image)

**Figure 7: WebML Unit Types vs. OO-H Filter Conditions**

This example illustrates the need to integrate the various WebML ContentUnits with OO-H Collections and ClassNodes as well as WebML LinkParameter and SelectorConditions with OO-H filter OCLExpressions.

### 4.3 Content Management Modeling (Layer 2)

Due to the differences at the content modeling layer, the modeling concepts for content management modeling are also differently defined in WebML and OO-H. For each operation on Entities of the content modeling layer WebML offers an explicit modeling concept, e.g., CreateUnit, DeleteUnit, and ConnectUnit. In contrast, OO-H’s `Service` package encompasses two concepts only, namely `ServiceNode` and `ServiceLink`. This means that OO-H does not differentiate between the typical create, update, and delete operations by defining sub-concepts of `ServiceNode`. Instead a `ServiceNode` has a reference to the Operation which should be executed when the `ServiceNode` is entered.

**Example:** The given scenario describes the deletion of a specific album by an authorized user. In Figure 6 (a) a DeleteUnit `DeleteAlbum` is shown which might be accessed, e.g., through an IndexUnit `AlbumSearch` (cf. Figure 7 (a)). Likewise, concerning OO-H a `ServiceNode` `DeleteAlbum` might be accessed, e.g., through a `ClassNode` (cf. Figure 7 (b)). For the given scenario we assume that the Selectors and SelectorConditions are translated according to the transformation rules defined for the hypertext modeling layer. Beyond, each OperationUnit from the WebML model needs to be translated into a `ServiceNode` in the OO-H model. Thereby, the reference identifying the corresponding operation type (cf. Table 2) must be set for the `ServiceNode`.

### 5 Conclusions and Future Challenges

In this paper we have presented our methodology of integrating three of the most prominent web modeling approaches, namely WebML, OO-H, and UWE, on the basis of a set of core web modeling requirements. As a proof of concept, we have defined the core languages in Ecore-based metamodels and subsequently, have implemented the integration in ATL model transformations rules. From our preliminary results and lessons learned from the integration of WebML and OO-H sofar, we conclude that the
core of the three languages can be integrated without loosing information. Nevertheless, the presented results are only a first step in the direction of a full integration of the languages and to the definition of a common metamodel for web modeling.

Future challenges concerning the integration of WebML and OO-H include the finalization of the integration for their core modeling concepts which requires the OCL version used in OO-H to be incorporated in the metamodel. Therefore, we plan to employ the EBNF_2_Ecore transformer [29], which is capable of generating the corresponding metamodel elements from a textual EBNF grammar. On the basis of this we intend to finalize the transformation rules from OO-H to WebML.

The UWE metamodel is currently under preparation. As soon as a first stable version is available, we plan to integrate UWE with the two other modeling languages as well. We expect that a third language would bring further insights for building the common metamodel for web modeling and on these results a first unification of the modeling concepts can be proposed for the core requirements.

Beyond the core requirements, the modeling requirements and modeling examples need to be extended to other web modeling concerns such as presentation, context-awareness, and business processes in the future to broaden the view on the unification of the modeling concepts. Furthermore, a refinement of possible variants of modeling requirements, in order to find further sub-concepts and alternative modeling styles would be of interest.

Acknowledgments. We would like to thank the members of the MDWEnet initiative that have contributed to this paper in terms of preliminary work, including Pierro Fraternali (Politecnico di Milano) for setting up the set of modeling requirements and Cristina Cachero, Jaime Gomez, Santiago Melià, Irene Garrigós (Universidad de Alicante) as well as Nora Koch (LMU München) for their work on the UWE metamodel.

References


Bridging the Gap between BPMN and WS-BPEL. M2M Transformations in Practice

Pau Giner, Victoria Torres, Vicente Pelechano

Department of Information Systems and Computation
Technical University of Valencia
46022 Valencia, Spain
{pginer, vtorres, pele}@dsic.upv.es

Abstract. The Web is being consolidating as the main platform for the development of applications. Moreover, these applications are not conceived just as isolated systems. This fact implies that the requirements that Web applications should satisfy are now more challenging than before. One important requirement for these systems is to provide support for the execution of business goals expressed by means of Business Process definitions. From the Web Engineering area, several methods have provided a solution to cope with this requirement. In this work we present, within the context of the OOWS Web Engineering method, how business process definitions are transformed into executable process definitions by the application of model-to-model transformations. To accomplish this goal, this work has been developed in the context of the Eclipse environment jointly with the BABEL project.

1 Introduction

Web applications are no longer conceived just as systems to perform CRUD functions over the persistence layer. In fact, the possibilities that bring the environment in which these applications live widen the kind of systems being build as well as introduce new challenges such as security, reliability, integration, etc.

One of the main advantages introduced by the Internet is that “services” are available 24x7. This fact allows service providers to reach a larger community of customers. Moreover, when these services are offered using a standard technology the potential number of customers can grow easily. In this direction, Web services were built as the standard technology to provide functionality over the Web.

However, the great potential of services does not limit to the use of services as units. In contrast, it is the service composition what brings value to them. Service composition usually involves the interaction between different partners, some of them behaving as clients and others as providers. Then, if we go one step forward, we can see service compositions as business processes, where different services provided by different partners are put together to accomplish certain agreed goals.

1 This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

270
In a previous work [11] we presented an extension to the OOWS [9] Web Engineering method to provide support for the generation of Business Process Driven Web applications. This extension embraced mainly the Navigational model. The main goal of this work was to obtain from a business process definition the Navigational model necessary to provide support to the original processes. Moreover, as these processes can extend in time, we decided to introduce into the architecture of the generated applications a process engine that was in charge of driving processes during their life. Therefore, we need to transform these business process definitions into a format that could be executed by the engine.

Moreover, following a Model Driven Approach for the construction of these applications allows us to define them in a technological independent way (in terms of the service composition) as well as to perform separation of concerns. In this case, as we bet on Web services the independence is relegated to service composition. From service compositions defined in the BPMN [5] notation we could then transform it into different process executable languages. In this work we focus on the generation of WS-BPEL [6].

The main contribution of this work is to present the application of the MDA approach within a Web Engineering method for the construction of Business Process driven Web Applications. In particular, this work focuses on the task of translating business processes defined graphically in the BPMN notation (defined at the PIM) into a specific language such as WS-BPEL (placed at the PSM level). Moreover, this work has been developed within the Eclipse and BABEL [2] projects.

The remainder of the paper is structured as follows. Section 2 provides a revision over the related work developed within in the Web Engineering area. Section 3 puts into context the work developed and presents the tools used to accomplish it. Section 4 provides a brief overview over the BPMN language (the language used in this work for service composition). Section 5 presents step by step the process followed to extend an existing to tool to provide full transformation from BPMN to WS-BPEL. Section 6 presents some conclusions about the experience of this work. Finally, two appendixes are included to show both, the schema generated for the extended tool and the ATL [1] transformations implemented for this purpose.

2 Related Work

Web Engineering methods provide modeling mechanisms (supported in some cases by tools) to overcome the development process of web solutions. Due to the inherent dynamism of the Web, most of these methods have evolved to provide support to the new arising requirements. As a result of these requirements a broader range of systems are considered by these methods. Within this range we find process-driven web applications.

In this context, some of the existing proposals developed within the Web Engineering area have coped with the issue of integrating business process with navigation. The solutions provided by these methods can be divided into two groups. In the one hand, some of them propose introducing business process primitives into the naviga-
tional model. Within this group we find proposals such as OOHDM [10], UWE [8] or WSDM [7]. On the other hand, others propose simply modeling the navigation that occurs during the process execution as if it were pure navigation. In this group we find the OO-H [8] proposal. All these proposals make use of a process notation such as UML Activity Diagrams or Concur Task Tree to define process requirements. However, it is important to note that the navigation allowed within a business process execution is completely driven by the process and not by the user. This means that it is not necessary to include in the Navigational model the process flow. In fact its inclusion can complicated considerably the understanding and modeling of the navigational model. Nevertheless, this do not implies that no navigation has to be defined for business process execution. In fact, it can be desirable to improve or complete the content of the navigational nodes (web pages) in order to generate better user interfaces.

Regarding workflows or long-running processes, WebML [4] and Hera [3] provide modeling mechanisms to support this kind of processes. The modeling of this kind of processes involves that multiple process and activity instances can be handled by users. Moreover, different users behaving with different roles are in charge of performing certain process tasks. In the one hand, the webML approach introduces a process reference model into its conceptual model. This process reference model is interconnected with the application data model and the user model and it is used to control the state of cases and processes. This proposal also introduces primitives to model the navigation that occurs during process execution. However, these are not separated from the ones that refer to pure navigation. As a result, navigational models can get complicated not only when the size of processes is considerable or the process grows but also if the control flow includes too many forks.

Similarly, Barna et al. in [3] propose a specification method for the automatic generation of web models from a workflow model. This proposal takes into account workflow processes providing a solution at the modeling level by introducing a task and workflow modeling phases. This proposal also considers the asynchrony that appears in workflow processes. To overcome it they present a mechanism based on message queues to handle multiple workflow instances. Again, the navigation of the workflow is moved to the navigation structure what complicates the definition of the navigation.

Our proposal tries to introduce modeling mechanisms for developing both short and long-running processes. Moreover, we have tried to minimize the impact that this new mechanisms can have over the remainder models. For this reason, we introduced the Business Process Model (BPM) that allows us defining the set of business processes that govern the organization. The Navigational Model is only used to specify the navigational contents that are going to be included in the implemented GUIs. No navigation is again defined in this model. The navigation that occurs during process execution is fixed and the user only has to follow it.
3 Work Context

This work has been developed as a part of a bigger project aimed at the development of Business Process driven Web applications. This bigger project involves the development of Web applications based on the MDA approach. Next two subsections are dedicated (1) to present an overview of the whole project pointing out the part covered in this work and (2) to reference the tools that have been used in its development.

3.1 Project Overview

As we have mentioned previously, the project has been conceived taking into account the MDA approach. Following this philosophy, the whole system is specified in a technological independent fashion by means of different models which represent the different aspects that characterize this kind of Web applications. Fig. 1 shows both the models included in the proposal (including the dependences between them) and the transformations required to evolve this specification into new models or even final code (depending on the case).

![Fig. 1 Project Overview](image)

This figure shows that the Business Process Model (BPM) is defined using functionality that is defined both in the Structural Model and in the Services Model. This allows the composition at the modeling level of internal functionality and functionality that is “imported” from external partners. The following paragraphs provide a rough explanation of these models.

The OO-Method (Object Oriented Method for Software Development) models specify the structural and functional requirements of dynamic applications. These models are:
- the Structural Model, which defines the system structure (its classes, operations and attributes) and relationships between classes by means of a Class Diagram.
- the Dynamic Model, which describes (1) the different valid object-life sequence for each class of the system using State Transitions Diagrams and (2) the communication between objects by means of Sequence Diagrams.
- the Functional Model, which captures the semantics of the state changes to define service effects using a textual formal specification.

The Services and Business Process models were introduced to specify the interaction with external partners.

- the Services Model, which brings up external services (functionality provided by external partners) into the modelling level in order to manage them more easily [12].
- the Business Process Model, which specify by means of BPMN diagrams a set of processes in which can intervene both functionality provided by the local system as well as functionality provided by external partners. The activities that made up these processes represent functionality defined both in the Structural Model and in the Services Model.

The OOWS (Object Oriented Web Solutions) models were introduced in order to extend conceptual modeling to Web environments. These models are:

- the User Model, which defines the kind of users that are going to interact with the web application. Moreover, it defines inheritance relationships between them.
- the Navigational Model allows us defining appropriate system views (in terms of data and functionality) for each kind of user defined in the User Model. This model was extended in a previous work [11] with a set of new primitives in order to integrate business process execution within the Navigation.
- the Presentation Model allows us to model presentation requirements (information paging, layout and ordering criteria) of the elements defined in the Navigational Model.

The whole project relies both on transformations between models (to move knowledge between different aspects) and between model(s) and text (to generate an equivalent/compliance representation in terms of an implementation language). In particular, the part of the project that has been implemented and presented in this work refers to the Model-to-Text transformation that moves process definitions represented in the BPM into the WS-BPEL executable language.

### 3.2 Technological Context

For the development of this work we have made use of a set of tools most of them included in the Eclipse project and that are commented in the following paragraphs.
The Eclipse Modeling Framework\(^2\) (EMF) is the basis of several modeling projects developed by the Eclipse community. EMF includes tools for the generation, edition and serialization of models conforming to Ecore metamodels (an implementation of the OMG’s Essential MOF to represent metamodels).

The necessity of operations to work with models comes from the fact that the Model Driven Development approach considers models as first-class citizens. The Atlas Transformation Language\(^3\) (ATL) was defined to cope with operations referred to model transformation. It allows the definition of transformation rules for the creation of one or more output models from several input models. Moreover, the scope of model transformation provided by this language is wide being quality improvement, model refinement or model merging examples of some applications. The concept of cartridge, a metamodel representing a technology and the corresponding code generation in conjunction, permits the usage of model transformation to generate final artifacts and helps to maintain the rationale of the generation in a model-to-model transformation. With this approach, in opposition to a direct model-to-code transformation, the artifacts involved in the development are maintained at the modeling level.

XML Schemas are used to define XML-based formats, deriving an Ecore metamodel from them allow the definition of Platform Specific Models (PSM). EMF permits the generation of Ecore metamodels from XML Schemas. Models conforming the generated metamodel are, when serialized, valid according to the schema. Web Tools Platform\(^4\) (WTP) project offers several editors for different web-related formats to ease Web applications and Web Services development. The XML tools have been used to define XML Schemas and test them.

The Babel BPMN2BPEL\(^5\) tool is a java application to transform BPMN diagrams into WS-BPEL definitions. BPMN is a graphical notation and has no defined textual format, so the tool input format is a concrete textual representation with no formal description. The lack of a model behind the input format prevents integration of the tool in a model driven environment. As the format is XML based, the definition of its supported XML Schema and the equivalent Ecore metamodel enables us (1) the creation of models and (2) the automatic code generation for the underlying format. Moreover, it constitutes a cartridge usable at the modeling level.

The SOA Tools Platform (STP) project aim is to offer a framework and a set of tools to enable the development of SOA based applications. One of its sub-projects\(^6\) consists on the definition of a graphical editor to create BPMN diagrams. The editor is based on Graphical Modeling Framework (GMF) and the BPMN metamodel has been defined in Ecore, that enables its usage with EMF-based tools. Business Processes are modeled using the BPMN graphical editor included in the STP project and a mapping targeting the platform specific model of Babel tool will be defined.

---

\(^2\) [http://www.eclipse.org/modeling/emf/](http://www.eclipse.org/modeling/emf/)
\(^3\) [http://www.eclipse.org/m2m/atl/](http://www.eclipse.org/m2m/atl/)
\(^6\) [http://www.eclipse.org/stp/bpmn](http://www.eclipse.org/stp/bpmn)
4 Using BPMN for Service Composition

As we have mentioned previously, service composition provide an added value to the companies that provide them. Moreover, the execution of compositions that include services that involve human participation can take a long time for being completed and then it requires keeping the state of the composed service. To accomplish this task we rely on the use of a process engine.

In the following subsections we present the languages used to define and execute service compositions.

4.1 BPMN to define Service Compositions

There are available several notations (such as UML Activity Diagrams, UML EDOC Business Processes, IDEF, ebXML BPSS or BPMN among others) that can be used to model BPs. In particular, we are going to use the BPMN notation because it provides a mapping between the graphics of the notation to the underlying constructs of an execution language, in particular to the WS-BPEL language, what makes this notation a good candidate to be used. This notation is designed to cover a wide range of type of diagrams (such as “high-level private process activities”, “detailed private business processes”, “detailed private business processes with interactions to one or more external entities” or “two or more detailed private business process interacting” among others). However, as our goal is to obtain those software components that implement these BP definitions, we are going to use the notation for the design of “detailed private business processes with interactions to one or more external entities”. It is important to make this clear in order to obtain, after the application of the transformation rules, a running Web Application solution.

4.2 WS-BPEL to Execute Service Compositions

The growth in the adoption of the Web service technology made us to consider services composition languages that were based on this technology. For this purpose, the OASIS consortium is been developing Web services standards to cover for instance issues such as security or e-business. One of these standards is the WS-BPEL, which allows describing the behavior of a business process based on interactions between the process and its partners through Web Service interfaces.

5 Bridging the Gap between BPMN to WS-BPEL

To achieve the goal established in this work (the translation between business processes definitions represented in the BPMN notation into WS-BPEL executable definitions) we have made use of the BPMN2BPEL tool. In fact, we have extended this tool to provide support to the whole transformation.
This tool is part of the process transformation tools developed in the BABEL project (project developed by the Business Process Management group at QUT). However, although this tool translates process models into process definitions represented in WS-BPEL we wanted to provide the transformation directly from the BPMN notation to WS-BPEL. Moreover, as these transformations represent just a part of a bigger project, which is being developed within the Eclipse environment, we wanted to integrate them as well in the same environment. Fig. 2 depicts graphically the process followed to accomplish these goals.

5.1 XML Schema Definition for the BPMN2BPEL tool

BPMN is a graphical notation that lacks of a standard textual representation. Then, the format used by the BPMN2BPEL tool is a particular XML application with no definition of its grammar. In order to make the tool more interoperable it is desirable to have, an XML Schema that represents the model behind the data. Although there are several options to define XML applications (DTD, XML Schema, Relax NG, Schematron, RDF and the like), we decided to use XML Schema because of its tight integration with the EMF tools. XML Schema can be converted to an Ecore metamodel and the instances of this model can be converted back again into an XML conformant with the schema.

First of all we needed to know the schema used by the BPMN2BPEL tool to represent processes. Based on the suit of examples attached to the tool we could extract and produce the process schema. The complete generated schema is included in the Appendix A. This schema defines three elements which are nodes, arcs and code. The first two elements refer to activities and flows respectively. The latter element allows the definition of code that is directly copied into the generated WS-BPEL file.

5.2 Transform the XML schema into the Ecore format

Then, to define transformations between the original BPMN model and the process model used by the BPMN2BPEL tool we have to transform the latter into the Ecore Metamodel (which is used by EMF-based tools and allows us to manipulate it properly).
5.3 Define model-to-model transformations

Once we have both, source (BPMN) and target (process definition used by the BPMN2BPEL tool) metamodels represented in the Ecore Metamodel we could proceed to define the model-to-model transformations. To perform this transformation we have used ATL as the transformation language. Although this language is not fully compliance with the QVT standard it is enough mature and reliable to accomplish the model-to-model transformations required in this project.
Fig. 4 depicts graphically a screenshot of the implemented model-to-model transformations to obtain the model accepted by the BPMN2BPEL tool. Appendix B includes all the ATL transformations defined in this work.

![Business Process Model](image)

**Fig. 5 Business Process Model defined using the BPMN editor provided within the STP project**

Finally, the application of these transformations allows us to use the BPMN2BPEL tool to obtain a process definition initially modelled following the BPMN notation (Fig. 5) into an executable definition in WS-BPEL.

### 6 Conclusions

The application of a Model Driven Approach has proven satisfactory in several aspects. In the one hand reduces the risk of manual errors (in this case the definition of BPMN models was correct since there is a metamodel behind the scenes that enforces a consistent definition them). On the other hand, the use of a graphical tool allows a more comfortable definition of BPMN models.

Moreover, the integration of tools at the modelling level allows the abstraction of technological details and decouples the concepts from its physical representation. Defining the bridge between models with a model-to-model transformation permits the creation of simple mapping rules only dependent of both metamodels. The transformation rules are explicit statements of the equivalence of the metamodels in a technological agnostic way, thus the maintainability of the rules is better than ad-hoc export solutions.

The paper has presented a real application of the MDA approach in the scope of the Web engineering area. In this case, model-to-model transformations have been implemented in the ATL language, which has been proved adequate to achieve the proposed goals. Moreover, the use of some of the projects developed under the Eclipse community such as STP, WTP, EMF and M2M, have made possible its successful development.
Similar to these transformations, the rest of the project involves both model-to-model transformations defined in ATL (to obtain navigational models from business process descriptions) and model-to-text transformations to obtain the executable code that satisfies the system defined at the modeling level.

References


Appendix A

This appendix provides a complete listing of the XML Schema used by the BPMN2BABEL tool as input format for business process definition.

```xml
<schema
  elementFormDefault="qualified"
  xmlns=http://www.w3.org/2001/XMLSchema
```
Appendix B

This appendix provides a complete listing of the ATL Transformations to move from the BPMN to BABEL input format.
module bpmn2babel;
create OUT : babel from IN : bpmn;

helper context bpmn!Activity def: isStartEvent:Boolean=
    let types: Sequence(bpmn!ActivityType)=
        Sequence{ #EventStartEmpty,
        #EventStartMessage,
        #EventStartRule,
        #EventStartTimer,
        #EventStartLink,
        #EventStartMultiple}
in
        types->includes(self.activityType);

helper context bpmn!Activity def: isMessageEvent:Boolean=
    let types: Sequence(bpmn!ActivityType)=
        Sequence{#EventIntermediateMessage}in
        types->includes(self.activityType);

helper context bpmn!Activity def: isTimerEvent:Boolean=
    let types: Sequence(bpmn!ActivityType)=
        Sequence{#EventIntermediateTimer}in
        types->includes(self.activityType);

helper context bpmn!Activity def: isXorJoin:Boolean=
    self.activityType=#GatewayDataBasedExclusive and --XOR
    self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isXorSplit:Boolean=
    self.activityType=#GatewayDataBasedExclusive and --XOR
    self.incomingEdges.size()<self.outgoingEdges.size();--input <
output

helper context bpmn!Activity def: isEbXorJoin:Boolean=
    self.activityType=#GatewayEventBasedExclusive and --EBXOR
    self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isEbXorSplit:Boolean=
    self.activityType=#GatewayEventBasedExclusive and --EBXOR
    self.incomingEdges.size()<self.outgoingEdges.size();--input <
output

helper context bpmn!Activity def: isAndJoin:Boolean=
    self.activityType=#GatewayParallel and --AND
    self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isAndSplit:Boolean=
    self.activityType=#GatewayParallel and --AND
    self.incomingEdges.size()<self.outgoingEdges.size();--input <
output

helper context bpmn!Activity def: isEndEvent:Boolean=
    let types: Sequence(bpmn!ActivityType)=
        Sequence{ #EventEndEmpty,
        #EventEndMessage,
        #EventEndError,
        #EventEndCompensation,
        #EventEndTerminate,
        #EventEndLink,
        #EventEndMultiple,
helper context bpmn!Activity def: isTask:Boolean=
  self.activityType = #Task;
helper context bpmn!Activity def: nodeType:babel!NodeType=
  let nodeTypes: Map(Boolean,babel!NodeType)=
    Map{
      (self.isStartEvent, #StartEvent),
      (self.isMessageEvent, #MessageEvent),
      (self.isTimerEvent, #TimerEvent),
      (self.isXorJoin, #XORJoin),
      (self.isXorSplit, #XORSplit),
      (self.isEbXorJoin, #EBXORJoin),
      (self.isEbXorSplit, #EBXORSplit),
      (self.isAndJoin, #ANDJoin),
      (self.isAndSplit, #ANDSplit),
      (self.isEndEvent, #EndEvent)
    } in
    nodeTypes.get(true);
rule Main {
  from
d: bpmn!BpmnDiagram
to
  r: babel!DocumentRoot(
    bpmn<- bp ),
  bp: babel!Bpmn(
    process<- p ),
  p: babel!Process(
    id<-d.name, nodes<-nodes, arcs<-arcs
  ),
  nodes: babel!Nodes( node<-bpmn!Activity.allInstances() --
    >union(bpmn!Activity.allInstances()--
      >collect(x|thisModule.resolveTemp(x,'gate'))) ),
  arcs: babel!Arcs( arc<-bpmn!SequenceEdge.allInstances()--
    >union(bpmn!Activity.allInstances()--
      >collect(x|thisModule.resolveTemp(x,'arc'))) )
}
rule Activity2Node{
  from a: bpmn!Activity to
  node: babel!Node(
    id<-a.iD, name<- a.name, type <- a.nodeType)
}
rule Sequence2Arc{
  from s: bpmn!SequenceEdge to
  arc: babel!Arc(
    id<-s.iD, source<- s.source.iD, target<- s.target.iD, guard<-s.name)
Model Transformation for Service-Oriented Web Applications Development

Valeria de Castro, Juan Manuel Vara, Esperanza Marcos

Kybele Research Group
Rey Juan Carlos University
Tulipán S/N, 28933, Móstoles, Madrid, Spain
{valeria.decastro, juanmanuel.vara, esperanza.marcos}@urjc.es

Abstract. In recent years, innovation in technologies such as web services, business process automation, etc., have motivated a new paradigm in the application development field to appear, known as Service-Oriented Computing. This new paradigm, which utilizes services as fundamental elements for developing applications, has encouraged the evolution of web applications and the way they are developed. Attending to this evolution we have already presented a model driven method for service-oriented web applications development. The method defines new Platform Independent Models (PIMs) and mappings between them. The PIMs proposed have been grouped in a UML profile based on the behavioral modeling elements of UML 2.0. In this work, we focus on the mapping between those PIMs and we define the model to model transformations needed for service-oriented web applications development. We first specify the transformation rules with natural language to later formalize them with graph transformation rules.

Keywords. Service-Oriented Web Applications, MDA, UML, Model Transformations, Graph Transformation Rules.

1 Introduction

A new paradigm in the field of application development, known as Service-Oriented Computing (SOC) [12] has encouraged the evolution of web applications and the way they are developed. Thus, while first web applications were created as a way to make available information to users, and they were built basically by linking static and dynamic pages; currently, most of the web applications are understood as networks of applications owned and managed by many business partners providing several services satisfying the needs of consumers that pay for them. Services usually range from quite simple ones, like buying a book or renting a car to the ones which involve complex processes such as obtaining sales ratings or participating in a public auction. For that reason, in the Web Engineering field, there is a need for methodologies for development based on current technologies such as web services, business process execution, etc.
Although the design and implementation of web services can be apparently easy, the implementation of business processes using web services is not so effortless. Languages for the implementation of business processes have many limitations when they are used in the early stages of the development process [19]. This occurs mainly because the transformation from high-level business models generally carried out by business analysts; to a composition language that implements those business processes with web services is not a trivial issue.

Model Driven Architecture (MDA) [11] provides a conceptual structure where the diagrams used by business managers and analysts, as well as the various diagrams used by software developers can be fit. Moreover MDA allows organizing them in such a way that the requirements specified in one diagram can be traced through the more detailed diagrams derived from the former. Hence, MDA is a useful tool to anyone interested in aligning business processes with IT systems [8].

This paper deals with the MDA approach for the development of service-oriented web applications. In a previous work we proposed a model-driven method which starts from a high level business model and allows obtaining a service composition model that makes easy the mapping to a specific web service technology [5]. To obtain this service composition model, which is represented through a UML activity model, the method defines: a Computational Independent Model (CIM) for business modeling, called value model [7]; four Platform Independent Models (PIMs) for the behavioral modeling of service-oriented web application; and mappings rules between them.

In this work we present the metamodels of the PIMs defined by the method, which includes new elements for service-oriented web applications modeling that extend the behavioral modeling elements of UML 2.0 [10]; and we focus on the mapping rules between these PIMs, which allows obtaining a service composition model that makes easy the mapping to a specific web service technology, starting from a high level UML use cases model in which the services required by the web consumers are represented.

Given that the method is based on a continuous development process in which, according to the MDA principles [9], the models act as the prime actors, mappings between models play a very important role. Each step of this process consists basically in the generation of an output model starting from one or more input models on which the mapping rules are applied. In this work, we follow a graph transformation approach to effectively realize the mappings between the PIMs proposed by the method. The term Graph Transformation is used to refer to a special kind of rule-based transformations that are typically represented diagrammatically [14]. So, given that the mappings were defined in a rule-based manner, it seems appropriate to use a graph transformation approach to later formalize them. A similar approach for object-relational database development was presented in a previous work [18].

The rest of the paper is structured as follows: section 2 presents the UML profile that includes the new elements for service-oriented web applications modeling at PIM level; section 3 describes the model to model transformations between the proposed

---

1 This research is partially granted by the GOLD projects financed by the Ministry of Science and Technology of Spain (Ref. TIN2005-00010).
PIMs; finally, section 4 concludes the paper by underlying the main contributions and the future works.

2 UML profile for service-oriented web applications modeling

As mentioned before, the method proposed for service-oriented web applications development defines four new PIMs for modeling the behavioral aspect of web applications: the Business Services model, the Extended Use Cases model, the Services Delivery Process model and the Services Composition model. Each one is defined through a metamodel that extends the UML metamodel [10]. Figure 1 shows the dependences of the new models proposed (shadowed in the figure) with respect to the UML packages for behavioral modeling. As shown in the figure, the models proposed in our method are represented through UML behavioral diagrams: while the business services model and the extended use cases model are represented through use cases diagrams, the services delivery process model and services composition model are represented through activity diagrams.

![Fig. 1. Dependencies of new models regarding the UML packages for behavioral modeling](image)

These new PIMs defined by the method include new modeling elements which have been grouped in a UML profile called MIDAS/BM (MIDAS Behavior Modeling). According to UML 2.0, a UML profile is a package that contains modeling elements that have been customized for a specific purpose or domain, using extension mechanisms, such as stereotypes, tagged definitions and constraints [10]. Our profile is defined over the behavioral modeling elements of UML 2.0 and it describes new elements for modeling the behavioral aspect of service-oriented web applications. Figure 2 shows the profile, including the newly proposed stereotypes that are applied over the existing metaclasses of the UML metamodel. The new stereotypes defined are described in Appendix A at the end of this document.

Next, we are going to present the metamodel of the new PIMs in which these elements are represented, to later describe the mapping rules between them. For the sake of space, we explain the metamodels by describing only the new elements defined,
the associations between them and the specification of the respective restrictions over these metamodels defined using the OCL standard. A complete example of how these models should be used can be found in [5].

**Fig. 2.** The MIDAS/BM profile

**Business Services Metamodel.** The business service model is an extension to the UML use cases model in which only the actors and the business services that the system will provide them are represented. We define a business service as a complex functionality offered by the system, which satisfies a specific need of the consumer. The consumers of the system are represented in this model as actors. The business services are represented in this model as use cases stereotyped with <<BusService>> (see stereotype BusinessService in Appendix A).

Figure 3 shows the business services metamodel in which the new modeling element is shadowed. In the business services model each business service is associated to the actor needing the business service.

**Fig. 3.** Business services metamodel
Extended Use Cases Metamodel. This metamodel also extends the elements of the UML package for use cases modeling. In the extended use cases model we propose to represent the basic or composite use services. We define a use service as a functionality required by the system to carry out a business service. Thus, it represents a portion of the functionality of a business service. A basic use service is a basic unit of behavior of the web application, for instance 'registering as a costumer'. A composite use service is an aggregation of either basic or composite use services. The composite and basic use services are represented in this model as a special kind of UseCase stereotyped with <<CS>> and <<BS>> (see stereotypes CompositeUseService and BasicUseService in Appendix A).

Figure 4 shows the extended use cases metamodel in which the new modeling elements are shadowed. Note that UseService is an abstract class therefore it is not represented in the extended use cases model.

Services Delivery Process Metamodel. This metamodel extends the elements of the UML activity package. In the service delivery process model we propose to represent the activities that must be carried out for delivering a business service. The activities of this model are called service activities. The service activities are obtained transforming the basic use services identified in the previous model into activities of a
process. So, the services activities represent a behavior that is part of the execution flow of a business service. A service activity is represented as an ActivityNode stereotyped with $<$<SAc$>$> (see stereotype ServiceActivity in Appendix A).

The ServiceActivity element is shadowed in Figure 5 which shows the services delivery process metamodel.

![Services Delivery Process Metamodel Diagram](image)

**Fig. 5.** Service delivery process metamodel

**Services Composition Metamodel.** This metamodel also extends the elements of the UML activity package. In this model we represent the execution flow of a business service too, but in a more detailed way by including the concepts: activity operation and business collaborator.

We define an *activity operation* as an action that is supported by the service activity. It is represented in this model as a special kind of ExecutableNodes stereotyped with $<$<AOp$>$> (see ActivityOperation in Appendix A). Additionally, the service composition model proposes to identify those activity operations that can be implemented as Web services, using a special kind of ExecutableNode stereotyped with $<$<WS$>$> (see stereotype WebService in Appendix A).

A *business collaborator* is defined as an organizational unit that carries out some activity operation which is involved in the services offered by the web application (i.e.: as a Web service). The business collaborators are represented in this model as ActivityPartitions, which can be indicated as a swim-lane in the activity diagram. The ActivityOperations and WebServices are distributed in ActivityPartitions according to the business collaborator that carries out the operation. A business collaborator can
be external to the system, in which case the ActivityPartition is labelled with the keyword «external».

Figure 6 shows the service composition metamodel, in which the new modeling elements are shadowed.

```
context Services Composition inv Contents_Model:
  self.classes->forAll(c |
    (c.oclIsKindOf(ExecutableNode) and
     c.stereotype.name -> includes("ActivityOperation")) or
    (c.oclIsKindOf(ExecutableNode) and
     c.stereotype.name -> includes("WebService")))
```

Fig. 6. Service composition metamodel

3 Model Transformation for service-oriented web applications development

As mentioned before, the proposed method for service-oriented web applications development is based on the definition of models at different abstraction levels, the basis of the model-driven development paradigm [2], [13]. In the previous section we have defined the metamodels (consequently the models) that must be considered in our method, thus, according to MDA principles, the only issue that must be faced in order to complete the proposal is the definition of the mapping between these models. This process stands for model transformation [11], [14].
3.1 Mapping Rules

Figure 7 shows the modeling process proposed for service-oriented web applications development that includes the models defined in the previous subsections. As stated earlier, in this work we focus on the mapping rules between PIMs, remarked in Figure 7. At PIM level, the process starts by building the business services model and includes two intermediate models to finally obtain the services composition model.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig7.png}
\caption{Modeling process for service-oriented web applications development}
\end{figure}

In relation to the way mappings should be defined in [11] it is stated that “the mapping description may be in natural language, an algorithm in an action language, or a model in a mapping language”. In this case, and as a first approach, we have decided to describe the transformation rules between models in natural language for later expressing them as graph transformation rules. These transformations rules are collected in Table 1. According to [11], as some of the mapping rules of the transformation process require design decisions, it is not possible to automate them completely. As a result, we have made the distinction between the mapping rules that can be Completely (C) or Partially (P) automated.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|p{10cm}|c|}
\hline
From & To & Mapping Rules & Grade of Autom. \\
\hline
Business Services Model & Extended Use Cases Model & 1. Every Service found in the business service model will be split into one or more CompositeUseService (CS) and/or BasicUseServices (BS). & P \\
\hline
Extended Use Cases Model & Service Delivery Process Model & 2. Every CS generated will be split into one or more BS. & P \\
\hline
Extended Use Cases Model & Service Delivery Process Model & 3. For every BS corresponding to a same BusinessService, there will be a ServiceActivity (SAct) in the service delivery process model that describe this BusinessService. & C \\
\hline
Extended Use Cases Model & Service Delivery Process Model & 4. Every extend association identified in the extended use cases model will be represented in the service delivery process model by a ForkNode. The SAct corresponding to the source BS of the extend association must be previ-
\hline
\end{tabular}
\caption{Mapping rules between PIMs in the method for service-oriented web applications development}
\end{table}
<table>
<thead>
<tr>
<th>Services Delivery Process Model</th>
<th>Service Composition Model</th>
<th>6. Every SAct found in the service delivery process model will be split into one or more ActivityOperation (ActOp).</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. The control flow between ActOps is the same as the flow between their relative SActs.</td>
<td></td>
<td>7.1 In the case of a SAct containing two or more ActOps, the designer has to choose the particular control flow between the ActOps.</td>
<td>P</td>
</tr>
</tbody>
</table>

### 3.2 Graph Transformation

To observe the MDA principles, the model to model transformation of our method for service composition modeling development must be automated, at least in some extent. To achieve this objective we have decided to use a graph transformation approach [1], [4], [16]. Using a graph transformation approach results in two main advantages: on the one hand, graph grammars are based on a solid mathematical theory and therefore they present a number of attractive theoretical properties that allows formalizing model transformations; on the other hand, the use of graph grammars for mappings definition could be shown as a direct step towards to implementation since projects like Attributed Graph Grammar System (AGG)[15], VIATRA[3] or ATOM3[6] will provide us with the facilities to automate model to model transformations defined as graph transformations. Moreover, as previously mentioned, the term Graph Transformation is used to refer to a particular category of rule-based transformations that are typically represented diagrammatically. So, given that we have already formally defined the mappings in a set of rules, it seems appropriate to translate these rules to graph transformations rules. Finally, from a pure mathematical point of view, we can think on UML-like models as graphs. A graph has nodes and arcs, while an UML model have classes and associations between those classes; this way the fact that models are well represented as graphs is particularly appealing to shorten the
distance between modelers and model transformation developers, a big problem around model transformation. Rule-based transformations with a visual notation may close the semantic gap between the user’s perspective of the UML and the implementation of transformations.

To express model transformations by graph grammars, a set of graph rules must be defined. These rules follow the structure LHS:= RHS (Left Hand Side:= Right Hand Side). Both, the LHS and the RHS are graphs: the LHS is the graph to match while the RHS is the replacement graph. If a match is found on the source model, then it is replaced by the RHS in the target model. In this work we have used the approach already applied in previous works like [18] to define the graph rules that collects the transformation rules proposed in Table 1.

According to these guidelines, we have defined the graph rules for the model transformations needed in our proposal for service-oriented web applications development that were susceptible of being expressed by graph grammars.

From now on we present these graph rules next to the respective definition rules in natural language. Figure 8 describes the mapping rules corresponding to transformations from the business services model to the extended use cases model. Figure 9 to 12 describe the mapping rules corresponding to transformations from the extended use cases model to the service delivery process model. Finally, Figure 13 describes the mapping rules corresponding to transformations from the service delivery process model to the service composition model. For the sake of space we have had to reduce the size of these pictures, in some cases they could result difficult to read. In order to improve their clarity, they can be acceded in http://kybele.es/models/MTsowa.htm.

**Fig. 8.** BusinessServices and Actors in the business services model mapped to CompositeUse-Services, BasicUseServices and actors in the extended use cases model
Fig. 9. Extend associations in the extended use cases model mapped to the service delivery process model.

Fig. 10. Extend associations (with several sources BasicUseServices) in the extended use cases model mapped to the services delivery process model.
5. Whenever an include association is found in the extended use cases model, a ServiceActivity (SAct) corresponding to the source BS of the include association must be subsequent to the SAct corresponding to the target BS of the include association.

**Fig. 11.** Include associations in the extended use cases model mapped to the services delivery process model.

---

5.1 If the include association has several targets, the designer must decide the appropriate sequence for the different SAct corresponding to the target BS (that will be obviously previous to the SAct corresponding to the source BS).

**Fig. 12.** Include associations (with several target BasicUseServices) in the extended use cases model mapped to the services delivery process model.
4 Conclusions and Future Works

In this work we have presented the model to model transformations needed to complete an MDA approach for service-oriented web applications development. This way, we have firstly described the metamodels for the PIMs considered by the method. They provide with new elements for service-oriented web applications modeling and extend the behavioral modeling elements of UML 2.0. Next we have defined the mapping rules between these PIMs following a graph transformation approach. As a first approach to model transformations from the proposal for service-oriented web application development, we have firstly defined the transformation rules in a declarative manner for later formalize them with graph rules in order to automate them using some of the existing facilities to automate graph transformations. The mapping rules defined in this work allows obtaining a service composition model that can be easily translate to a specific web service technology, starting form a high level use cases model in which the services required by the web consumers were represented.

This work serves as a clear example of the value of model transformations in Software development: the model to model transformations presented in this work complete the definition of our process for service-oriented web applications development, a contrasted and published method that founds in model transformations the piece that remained to become a completely feasible methodology.

At the present time we are working in the integration of the method described in this work in a CASE tool which is now under development in our research group and
which its early functionalities have already been presented in previous works [17]. Besides, the open issue of making automatic the graph transformations by using existing technologies like ATOM 3 has been tackled.

References

Appendix A: Stereotypes of MIDAS/BM profile

This appendix includes all the stereotypes defined in the MIDAS/BM profile. It defines the new modeling elements which extend the existing metaclasses of the UML metamodel. For each modeling element we describe UML metaclass extended, semantics and notation.

<table>
<thead>
<tr>
<th>Business Services Model</th>
<th>&quot;BusinessService&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
<td>UML metaclass “useCase”</td>
</tr>
<tr>
<td>Semantics</td>
<td>Represent a complex functionality, offered by the system, which satisfies a specific need of a consumer.</td>
</tr>
<tr>
<td>Notation</td>
<td>&lt;&lt;BusService&gt;&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extended Use Cases Model</th>
<th>&quot;CompositeUseService&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
<td>UML metaclass “useCase”</td>
</tr>
<tr>
<td>Semantics</td>
<td>Represent a functionality that is required to carry out a business service, which is composed of other basic or composite use services.</td>
</tr>
<tr>
<td>Notation</td>
<td>&lt;&lt;CS&gt;&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;BasicUseService&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
</tr>
<tr>
<td>Semantics</td>
</tr>
<tr>
<td>Notation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services Delivery Process Model</th>
<th>&quot;ServiceActivity&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
<td>UML metaclass “ActivityNode”</td>
</tr>
<tr>
<td>Semantics</td>
<td>Represent a behavior that is part of the execution flow of a business service.</td>
</tr>
<tr>
<td>Notation</td>
<td>&lt;&lt;SAc&gt;&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services Composition Model</th>
<th>&quot;ActivityOperation&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
<td>UML metaclass “ExecutableNode”</td>
</tr>
<tr>
<td>Semantics</td>
<td>Represent an action that is supported by a service activity.</td>
</tr>
<tr>
<td>Notation</td>
<td>&lt;&lt;AOp&gt;&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;WebService&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend</td>
</tr>
<tr>
<td>Semantics</td>
</tr>
<tr>
<td>Notation</td>
</tr>
</tbody>
</table>
Modeling data-intensive Rich Internet Applications with server push support *

Giovanni Toffetti Carughii
Politecnico di Milano,
Dipartimento di Elettronica e Informazione,
Via Giuseppe Ponzio, 34/5 - 20133 Milano - Italy
giovanni.toffetti@polimi.it

Abstract. Rich Internet applications (RIAs) enable novel usage scenarios by overcoming the traditional paradigms of Web interaction. Conventional Web applications can be seen as reactive systems in which events are 1) produced by the user acting upon the browser HTML interface, and 2) processed by the server. In RIAs, distribution of data and computation across the client and the server broadens the classes and features of the produced events as they can originate, be detected, notified, and processed in a variety of ways. Server push technologies allow to get over the Web “pull” paradigm, providing the base for a wide spectrum of new browser-accessible collaborative on-line applications. In this work, we investigate how events can be explicitly described and coupled to the other concepts of a Web modeling notation in order to specify server push-enabled Web applications.

1 Introduction

Rich Internet Applications (RIAs) are fostering the growth of a new generation of usable, performant, reactive Web applications. Whilst RIAs add complexity to the already challenging task of Web development, among the most relevant reasons for their increasing adoption we can consider: 1) their powerful, desktop-like interfaces; 2) their accessibility from everywhere (along with the fact that final users tend to avoid installing software if a Web-accessible version exists); 3) the novel support they offer for on-line collaborative work. This last aspect is based on getting over the limits of Internet standards to provide server push techniques, enabling applications such as instant messaging, monitoring, collaborative editing, and dashboards to be run in a Web browser.

In this paper we focus on push-enabled Rich Internet Applications and the lack of existing modeling methodologies catering for their specific features. The most significant contributions of this work are:

1. the extension of a Web engineering language to consider collaborative Web applications using push technologies (Section 3) through the individuation

* We wish to thank Alessandro Bozzon, Sara Comai, and Piero Fraternali for the precious help and insightful discussions on this work.
of a set of primitives and patterns (Section 4) catering for the different aspects of distributed communication such as (a)synchronicity, persistence, and message filtering. We stress that the extensions we propose are general and can be applied to the most relevant Web engineering approaches;

2. a validation by implementation of the proposed concepts and notations (Section 5).

1.1 Running example

To ease the exposition we will use an example: as a case study, we consider a collaborative on-line application for project management. The aim of the application is to serve as an internal platform to let users communicate and organize projects and tasks. The purpose is to have a simple but consistent example we can use throughout the different sections: we will keep it as naive as possible so as to avoid cluttering diagrams with unnecessary detail, but the concepts we will introduce are general and can be applied to complex industrial scenarios.

Application users impersonate different roles: project managers and project participants. A project manager is responsible of creating projects and connecting them to their participants. For each project, tasks are created, precedence relationships among them are defined, and they are assigned to project participants for execution. Project participants can exchange messages with their contacts, while performing assigned tasks and signaling their completion.

![Data Model of the project management application](image)

Figure 1. Data Model of the project management application

Figure 1 shows the data model for the collaborative on-line application: the user entity represents all application users, a self-relationship connects each user with his contact list. A single user can participate in one or more projects, while each project can be directed by a unique manager. Users are assigned tasks: each task belongs to a project and can have precedence constraints w.r.t. other tasks. Messages can be exchanged between users, they have a sender and a set of recipients.
2 Background

RIAs extend the traditional Web architecture by moving part of the application data and computation logic from the server to the client. The aim is to provide more reactive user interfaces by bringing the application controller closer to the final user, minimizing server round-trips and full page refreshes.

The general architecture of a RIA is shown in Figure 2: the system is composed of a (possibly replicated) Web server\(^1\) and a set of user applications (implemented as JavaScript, Flash animations, or applets) running in browsers. The latter are downloaded from the server upon the first request and executed following the code on demand paradigm of code mobility [7].

\[\text{UI updates} \rightarrow \text{UI Events} \rightarrow \text{Web Server} \]

\[\text{Web Server} \rightarrow \text{XML messages} \rightarrow \text{Client Data} \rightarrow \text{Client Logic} \rightarrow \text{User Interface} \]

\[\text{User1} \rightarrow \text{UserN} \]

\[\text{Fig. 2. A Rich Internet Application architecture} \]

Each user interacts with his own application instance: as part of the computation is performed on the client, communication with the server is kept down to the minimum, either to send or receive some data, generally in XML.

Server-push and communication The communication between client and server is bidirectional in the sense that, after the first client request, messages can also be pushed from the server to the client. This is technically achievable because of the novel system architecture including client-side executed logic.

In a ”traditional” Web application the HTML interface is computed by the server at each user’s request. When the user interacts with a link or a button on the page the server is invoked and a new page is computed and sent back to the client. The role of the browser is simply to intercept the user’s actions, deliver the request to the server, and display a whole new interface even if the change is minimal.

\(^1\) We abstract from the server-side internal components as they are not relevant for the discussion.
In a RIA instead, the client-side logic handles each user interaction updating interface sub-elements only. Furthermore, when client-server communication is needed to transfer some data, it can be performed in the background, asynchronously, allowing continuous user interaction with the interface. This, together with HTTP/1.1 persistent connections, are the key ingredients of a technique called HTTP trickling, one of the solutions enabling servers to initiate server-to-client communication (server-push) once a first HTTP request is performed. The programming technique for server-push is also known as “Comet”, relevant implementations are Cometd\(^2\), Glassfish\(^3\), and Pushlets\(^4\), but most Rich Internet Application development frameworks provide their own.

A user can therefore be notified of events occurring outside of his application instance, either other users actions or in general occurring on the server. Direct communication between client applications is in general not possible\(^5\), but the server can be used as an intermediary (i.e., a broker).

Most on-line applications (especially collaborative) can benefit from this approach: think for example of work-flow-driven applications, shared calendars or whiteboards, stock trading platforms, plant monitoring applications, and so on. Interaction of other users, server internal or temporal events, Web service invocations can all be occurrences that can trigger a reaction on a user client application.

Considering the project management case study, server push can be used for instant messaging, or to signal task assignments and task completions in order to immediately start the execution of new tasks.

### 2.1 Problem statement

As Rich Internet Application adoption is experiencing constant growth, a multitude of programming frameworks have been proposed to ease their development. While frameworks increase developer productivity, they fail in providing instruments to abstract from implementation details and provide an high level representation of the final software; this becomes necessary when tackling the complexity of large, data-intensive applications.

While Web engineering methodologies offer sound solutions to model and generate code for traditional Web applications, they generally lack the concepts to address Rich Internet Application development. In a previous work [4] we suggested an approach to tackle these shortcomings concerning data and computation distribution among components; here, we focus on another essential

\(^2\) http://cometd.com/
\(^3\) http://glassfish.dev.java.net/
\(^4\) http://www.pushlets.com/
\(^5\) Web clients are in general not addressable from outside their local area network; in addition most browser plug-ins run in *sandboxes* preventing them to open connections to other machines than the code originating server. For this reason RIAs only allow direct communication between clients and the server; no direct communication can take place between client instances.
aspect of RIAs: server push and the new interaction paradigms and application functionalities it enables.

3 Approach overview

In a traditional Web application, data and computation reside solely on the server. Thus, the whole application state, and all actions upon it, are concentrated in a single component.

In a RIA, distribution of data and computation across the server and different clients causes:

- user actions to be performed asynchronously w.r.t. the server and other user applications;
- client-side data for each user and server-side data to evolve independently from each other.

In order to achieve better reactivity client-server communication in RIAs is reduced to the minimum: therefore a mechanism is needed to signal relevant events, either to reduce the misalignment between replicated and distributed data, or simply to signal that, at a specific instant, an action is being performed in the distributed system. The classes of actions that are relevant for a system are application-specific.

**Example** For instance, considering our running example, each action upon a task instance can be considered a specific *event type*, we have event types:

**Task assigned:** when a project manager performs the action of assigning a task to a specific project participant;

**Task completed:** when a project participant marks one of her assigned tasks “completed”.

Both actions can be signaled to application users to begin working on the associated or next tasks.

3.1 Notification communication

Considering that in a RIA all communication must go through the server (as described in Section 2), only two scenarios apply:

1. An event occurring on a client application has to be notified. In this case the notification starts from a client application, goes to the server, and is eventually delivered to other users;
2. An event occurring on the server has to be notified. In this case only server to client communication takes place.

Different aspects can influence the process of communicating event notifications across the system components: for instance how to identify notification recipients, or whether the communication happens synchronously or not. We
call these aspects semantic dimensions, they are listed in Table 1. Semantic dimension identification is necessary in order to correctly draw the primitives and devise the appropriate patterns covering all their possible combinations. In the following paragraphs we give a brief definition of each aspect.

**Event filtering: location and logic.** Not all users need to be notified of all event occurrences; in distributed event systems the process of defining the set of recipients is generally indicated as *event filtering* [16] and two dimensions can influence it: *where* it takes place and *how*.

The former dimension considers the most general architecture for publish / subscribe systems [12] that involves three kinds of actors: a set of publishers, a broker, and a set of subscribers. Events occur at publishers that alert the broker who, in its turn, notifies the subscribers. Thus, the decision of which subscribers to notify can be taken at the publisher, at the broker, or all subscribers can be notified leaving to them the decision of whether to discard or not an already transmitted notification.

The latter dimension considers the logic that is used to determine notification recipients: the spectrum of possibilities ranges from statically defined recipients, to conditions on event attributes, to interpreted run-time defined rules (e.g., using a rule engine to detect composite event patterns).

**Communication persistence.** In distributed systems message communication can be [20]:

- *Persistent*: a message that has been submitted for transmission is stored by the communication system as long as it takes to deliver it to the recipient. It is therefore not necessary for the sending application to continue execution after submitting the message. Likewise, the receiving application need not be executing while the message is submitted;
- *Transient*: a message is stored by the communication system only as long as the sending and receiving applications are executing. Therefore the recipient application receives a message only if it is executing, otherwise the message is lost.

Depending on the application, some events may need to be communicated in a persistent way to prevent their loss, others only need transient communication. For example, email transmission uses persistent communication: the message is accepted by the delivery system and stored until it is deleted by the recipient;
the sender completes the communication as soon as the message is accepted by the delivery system.

Transient communication, instead, does not store the message and requires both sender and recipient to be running and on-line: it is often used when the loss of some event notification is not critical. For instance, in a content management system the event that two users are trying to edit the same resource is important for the colliding users, but if one of them disconnects before receiving the notification it’s no use storing it persistently.

4 Proposed Extensions

In this section we present the extensions we propose to cover all the combinations of the previously introduced communication aspects. We will illustrate them in WebML [9], although we stress that they apply in general to most Web engineering languages. First we will consider the data model, then the navigation model.

4.1 Extension to the data model

Application-specific event types are represented by adding new entities to the data model organised in a specialisation hierarchy. All event types extend a predefined Event entity and can have relationships with application domain entities.

We chose to extend the existing data model instead of adding a new event model so that we could leverage existing CRUD WebML operations leaving full control to the application designer to:

- enable persistent communication by directly storing event occurrences in the database (or client-side storage);
- represent and instantiate relations between event occurrences and domain model entities;
- instantiate an event base and explicitly draw ECA rules with provision for specific dimensions such as granularity, composite event detection, and event consumption [13].

**Example** Considering the project management example: the event types “Task assigned”, and ”Task completed” apply. They are represented in the event hierarchy in Figure 3. The WebML data model of Figure 1 is extended with entities representing the needed event types that are connected by means of relations to the application domain entities. Thus, the events ”Task assigned” and ”Task completed” have a relation with the task to which they refer. In addition to being related to a task, the event ”Task assigned” also has a relationship to the users to which the assignment was made.

---

6 Create, Read, Update, Delete
4.2 Extension to the hypertext model

In order to support event notifications, we added to the WebML hypertext model two operations: send event and receive event. The former allows one to send an event notification to a (set of) recipient(s); the latter is used to receive the notification and trigger a reaction. Send and receive event operations allow (indirect) communication among users without the need to use data on the server. Each operation is associated with an event type as defined in the extended data model. The event type provides both mapping between send and receive event operations (i.e., operations on the same event type trigger each other) as well as their specific parameter set. Operations for conditional logic (e.g., switch-operation, test-operation, as introduced in [5]) or queries can be used in conjunction with event-related operations in order to specify event filtering: retrieving notification recipients from a database, or discarding notifications upon application-specific conditions. Different patterns apply, catering for the possible combinations of filtering and communication needs (see Section 3.1).

**Fig. 3.** The data model of Figure 1 extended with event types

**Fig. 4.** On the left, Send Event operation usage. On the right a Receive Event operation

**Send event operation** A send event operation (Figure 4) triggers the notification of an event. It needs to be explicitly activated by a navigation link or
by an OK or KO-link. It is associated with an event type (see Section 4.1), and consumes the following predefined input parameters:

1. sender [optional]: the unique identifier of the sender of the event (e.g., a user ID, or the server)
2. recipient: the (set of) identifier(s) of the recipient(s) (e.g., user IDs, the server). The ‘*’ wild-card is used to indicate that the event is to be notified to all possible recipients.

Additional input parameters stem from the associated event type. Send event operations have no output parameters.

**Example** Considering the project management case-study, the send event operation can be used to signal the assignment of a task to a project participant as in Figure 5. The data-unit *Current Task* is used to show the selected task to be assigned, *Current Project* provides the current project identifier to show project participants in the *Project Members* index-unit. Selecting one, the *Assign Task* connect-operation is invoked to instantiate a relationship between the current task (with ID *TaskID*) and the selected user (with ID *UserID*): the same IDs are provided to the *NotifyAssignment* send event operation to define the notification recipient and to set the notification parameter identifying the assigned task.

**Receive event operation** A receive event operation (Figure 4) is an *event notification listener*: it is triggered when a notification concerning the associated event type is received. It is therefore associated with an event type, it has no input parameters, and the following predefined output parameters:

1. sender: the unique identifier of the sender of the event
2. recipient: the id set of recipients of the notification
3. timestamp: the timestamp at server when the event was signaled

In addition to the predefined output parameters, a receive event operation also exposes the parameters of the associated event type to be used by other units (e.g., for condition evaluation). Receive event operations only have exiting transport links or OK-links and cannot have incoming links.
Example Figure 6 shows the hypertext model of a page receiving a task assignment notification. Page *My Tasks* shows a list of task assigned to the current user with index-unit *Current Tasks*. It is a RIA page, marked with 'C' in the upper left corner, to specify that the page contains client-side executed logic. Thus, the page can establish a persistent connection with the server and receive notifications: when a *Task Assigned* notification is received, the *RecAssignment* receive event operation is triggered. Upon activation, the unit passes the *TaskID* parameter to the *New Task* data-unit that will retrieve the task details from the server to be shown in a pop-up window; the *refresh()* signal on the transport link connecting *RecAssignment* with *Current Tasks* causes the latter to refresh its content to show the updated task assignments list.

A receive event operation can be placed both in a siteview\(^7\) (starting an operation chain ending in a RIA page), or in a new diagram: the *event view*. Respectively the former solution specifies the reaction that will be performed by the client application if the notification recipient is on-line and viewing a determined page, the latter specifies what condition evaluation and actions will be performed when the server (which is supposed to be always on-line) receives the notification.

Event view The event view models the *server* reaction upon receipt of event notifications. Reactions to events are modeled by means of operation chains\(^8\) starting with a receive-event-operation. They can trigger any server-side operation such as invoking Web services, sending emails, performing CRUD operations on the database, or signaling new event occurrences.

The event view:

- provides a mechanism for asynchronous or persistent communication, by letting a designer specify the server reaction to an event notification. This can include making the notification persistent using the database, to asynchronously signal the event when the intended recipient is back on-line;

---

\(^7\) The WebML diagram representing the hypertext structure for an application user

\(^8\) i.e. Sequences of WebML operations
can be used together with conditional and query operations to specify *broker filtering* (Section 3.1) using server-side resources (e.g., databases, rule-engines, subscription conditions);

- allows reuse by factoring out operation chains triggered from different sources (e.g., different site views, areas, pages).

**Example** Figure 7 depicts an event-view operation chain for the running example. The application requires that a notification be sent to a user when a task is assigned to her. When the user is off-line, she cannot receive notifications with server push: on the server, the *User Online?* switch-operation triggers a send-email-operation if the recipient is not immediately reachable.

![Diagram](image)

**Fig. 7.** If the assignment user is offline, send her an email

5 Considerations and experience

The primitives and models we propose provide a simple notation for the specification of server push-enabled Rich Internet Applications. They have been designed so that the combination with WebML primitives enables the complete coverage of the semantic dimensions space individuated in Section 3.

- Receive-event-operations used in conjunction with switch-operations provide a mechanism to draw Event-Condition-Action rules distributed both on the server (event view) and client (siteview) applications. This caters for concerns such as filtering at recipient and broker, as well as detecting composite events\(^9\).

- Send-event operations in concert with query-based units allow the design of different communication paradigms including publish-subscribe, uni-, multi-, and broadcast. WebML units such as the selector unit can be used to cater for both static and query-based filtering to select event notification recipients.

- The event view lets a designer specify the behaviour on the server upon event notifications in order to provide support for asynchronous communication and event persistence.

\(^9\) e.g. by storing event occurrences in an event base and considering conditions querying it upon event occurrences
The same primitives can be used to represent other classes of system events such as, for instance, temporal events, external database updates, or Web service invocations. We have implemented the integration of such events in our prototype, but, due to space reasons, we do not discuss them here\(^\text{10}\). Concerning database updates, our solution was inspired by the work in [22].

### 5.1 Implementation

The implementation of the presented concepts builds on the runtime architecture for WebRatio we developed for our previous work presented in [4]. We developed a working prototype of a code generator from the extended WebML notation to Rich Internet applications implemented using the OpenLaszlo [1] technology. To validate our proposal, we extended it with the needed components for server-push: the resulting architecture is shown in Figure 8.

![Figure 8](image)

**Fig. 8.** The runtime architecture of our prototype implementation

OpenLaszlo provides natively the concept of a persistent connection\(^\text{11}\) to enable server to client communication. Thus, the implementation of the receive event and send event operations on the client-side are quite straightforward. An event handler (Event Manager) is triggered whenever the Dataset\(^\text{12}\) associated with the persistent connection receives a message (an event notification) from the server. The message is an XML snippet, its structure reflects that of the associated event type, each message carries its type information. The Event Manager checks the type information and triggers the appropriate receive event operation instance passing the pointer to the received message. The receive event operation extracts relevant parameters from the message, sets the values for its outgoing parameters and calls the next operation in chain.

A send event operation on the client builds, when triggered, an XML snippet reflecting the associated event type structure and whose attribute and text values

\(^{10}\) A complete exposition can be found in [21]

\(^{11}\) [http://www.openlaszlo.org/lps/docs/guide/persistent_connection.html](http://www.openlaszlo.org/lps/docs/guide/persistent_connection.html)

\(^{12}\) a data store of XML in OpenLaszlo
stem from the operation input parameters values upon invocation; then it invokes the sendXML() method of the persistent connection.

The server-side stub of the persistent connection is a servlet and thus accessible through a regular HTTP request. We extended it so as to be able to intercept event reception on the server and trigger the appropriate server-side operations in the event view (using the server-side Event Manager).

The code-generator was used to produce the running code of a personal information management application with features such as a shared calendar, and instant messaging.

6 Related work

Although RIA interfaces aim at resembling traditional desktop applications, RIAs are generally composed of task-specific client-run pages deeply integrated with the architecture of a traditional Web application. Web Engineering approaches build on Web architectural assumptions to provide simple but expressive notations enabling complete specifications for code generation. Several methodologies have been proposed in literature like, for example, Hera [23], OOHDM [19], WAE [10], UWE [14], and W2000 [3], but, to our knowledge, none of them addresses the design and development of RIAs with server push support.

This work, instead, considers system reaction to a collection of different events depending on any user interaction, Web service calls, temporal events, and data updates. Apart from defining the generic concept of event in a Rich Internet Application, our approach also individuates the primitives and patterns that can be used to notify and trigger reactions upon external events. This is something that, to our knowledge, all Web engineering approaches lack as reactions are only considered in the context of the typical HTTP request-response paradigm. Also, the approach we suggest provides the interfaces and notations necessary to integrate external rule engines (e.g., to detect composite events) with the Web application enabling the specification of reactions to events in terms of hypertext primitives (pages, content units, and operations).

Both a survey and a taxonomy of distributed events systems are given in [16]: most of the approaches bear the concepts individuated in [18] concerning event detection and notification, or in [12] w.r.t. the publish-subscribe paradigm. To cite but a few relevant works we can consider [2], [8], and [11]. Reactivity on the Web is considered for example in [6] where a set of desirable features for a language on reactive rules are indicated (the actual language is proposed in [17]).

With respect to these works, ours addresses events and notifications in a single Web application where on-line application users are the actors to be notified. In contrast, the above proposals aim at representing Internet-scale event phenomena with the traditional problems of wide distributed systems such as, for instance, clock synchronization and event ordering [15]. The system we are considering, instead, is both smaller in terms of nodes, and simpler in terms
of topology: the server acts as a centralized broker where all events are ordered according to occurrence or notification time. Nevertheless, it enables the implementation of complex collaborative on-line applications accessible with a browser.

7 Conclusions

Server push in RIAs provides the means to implement a new generation of on-line collaborative applications accessible from a Web browser. Although notification-based communication of events (e.g., publish / subscribe) and server-push technologies (e.g., based on AJAX) are well-known and established technologies, the explicit definition of the primitives supporting the modeling of Web applications employing said style of communication, and their introduction to modeling languages are new contributions. In this work we have presented the extension we propose to a Web Engineering language to represent the novel interaction paradigms enabled by Rich Internet application technologies. We considered the possible ways in which events occurring across different system components can be detected, notified, and processed in order to identify a set of simple, yet expressive, primitives and patterns.

References

International Conference on Web Engineering 2007

Workshop on Web Quality, Verification and Validation

Organisers
Tevfik Bultan, University of California Santa Barbara, USA
Coral Calero, University of Castilla-La Mancha, Spain
Angélica Caro, University of Bio Bio, Chile
Alessandro Marchetto, Fondazione Bruno Kessler – IRST, Italy
Mª Ángeles Moraga, University of Castilla-La Mancha, Spain
Andrea Trentini, Università degli Studi di Milano, Italy

Workshop program committee members
Silvia Abrahão, Universidad Politécnica de Valencia, Spain
Carlo Bellettini, Università degli Studi di Milano, Italy
Cornelia Boldyreff, University of Lincoln, UK
Cristina Cachero, Universidad de Alicante, Spain
Oscar Díaz, Universidad del País Vasco, Spain
Howard Foster, Imperial College London
Marcela Genero, Universidad de Castilla-La Mancha, Spain
Tiziana Margaria, University of Potsdam
Emilia Mendes, University of New Zealand, New Zealand
Sandro Morasca, Universita dell’Insubria, Italy
Mario Piattini, Universidad de Castilla-La Mancha, Spain
Marco Pistore, Universita di Trento, Italy
Lori Pollock, University of Delaware, USA
Antonio Vallecillo, Universidad de Málaga, Spain
Tao Xie, North Carolina State University, USA
Andrea Zisman, City University, London, UK
Foreword

This volume contains the proceedings of the First Workshop on Web Quality, Verification and Validation (WQVV2007) that took part in conjunction with the 7th International Conference on Web Engineering (ICWE2007).

The main topics of the workshop were related to quality, verification and validation of web, fundamental factors nowadays when advances in technology and the spread of Internet have favoured the appearance of a great variety of Web software systems (i.e., applications, portals and Web services). The success of these technologies in several fields such as the electronic commerce and their increasing use in safety critical applications makes quality, validation and verification of Web-based software important and critical factors/problems. Developing “good” Web applications will require effective approaches, methods and tools to design, model, develop, evolve and maintain these software systems.

Considering all this, the objective of this workshop was to bring together members of the academic, research, and industrial community interested in quality, testing, analysis and verification of Web applications. We wanted to promote all these areas by giving to the researchers the opportunity of sharing their works and a place for discussing. Moreover, we hope to identify during the workshop possible future lines related to the topics of the workshop, establishing the foundations for the creation of a community of researchers with interest in these areas. Additionally, two invited speakers were held. One of these speakers was Silvia Abrahao from the Politechnical University of Valencia in Spain, who talked about “Bridging the Gap between Model-driven Development and Web Project Estimation”. The invited speakers are internationally recognized specialists in different areas and their talks have definitely contributed to increase the overall quality of the workshop.
The program of this workshop required the dedicated effort of many people. Firstly, we must thank the authors, whose research and development efforts are recorded here. Secondly, we thank the members of the program committee for their diligence and expert reviewing. Last but not least, we thank the invited speakers for their invaluable contribution and for taking the time to synthesize and prepare their talks.

July 2007

María Ángeles Moraga
Andrea Trentini
Program Chairs
WQVV2007

SPONSORED BY
This workshop has been partially supported by Escuela Superior de Informática of University of Castilla-La Mancha and the CALIPSO project (TIN20005-24055-E) supported by the Ministerio de Educación y Ciencia (Spain).
Table of Contents

Towards Self-Managing Services (invited talk by Mario Pezzè) ....... 319

Bridging the Gap between Model-driven Development and Web Project Estimation (invited talk by Silvia Abrahão) ................................. 321

Analyzing Trackback Usage as a Inspection of Weblog Data Quality.323
  Shinsuke Nakajima

Improving the Quality of Website Interaction with Lightweight Activity Analysis ................................................................. 334
  David Nutter, Cornelia Boldyreff and Stephen Rank

Establishing a quality-model based evaluation process for websites . 344
  Isabella Biscoglio, Mario Fusani, Giuseppe Lami and Gianluca Trentanni

Subjectivity in Web site quality evaluation: the contribution of Soft Computing ................................................................. 352
  Luisa Mich

Testing Techniques applied to AJAX Web Applications .......... 363
  Alessandro Marchetto, Paolo Tonella and Filippo Ricca

Automated Verification of XACML Policies Using a SAT Solver ... 378
  Graham Hughes and Tevfik Bultan
Towards Self-Managed Services

invited talk

Mauro Pezzè

University of Lugano (Switzerland)
and
Università degli Studi di Milano Bicocca (Italy)
www.inf.unisi.ch/faculty/pezze/

Abstract. Services and service-based applications enable new design praxis. Many service-based applications are intrinsically multi-vendor, multi-platform, quickly evolving and changing to adapt to modifications in requirements, environmental conditions and user needs. Users and developers have little or no control on the whole code: different vendors can independently update services embedded in applications maintained by other parties, and applications can dynamically select alternative services to better meet the required quality of service [1].

Classic maintenance cycles that require expensive test and debugging activities, and that interrupt system operation to identify and remove faults, and check for the validity of updates and new functionality, may not work for service-based applications due to cost and time constraints. If for example an application fails in completing a transaction, suspending the application to diagnose and remove the fault can avoid future failures, but will not satisfy current users.

In this new scenario, classic test and debugging techniques may not suffice anymore, since many of them do not adequately cope with evolving requirements, lack of access to source code, dynamically reconfiguring applications and environment dependent behaviors.

Emerging results in autonomic and self-managed software systems may address many of the problems that characterize service-based applications. Although the terminology is not standardized yet, and some terms are overloaded, autonomic or self-managed refer to software systems that can identify problems, diagnose, and fix faults without human intervention. [4]. A self-managed software system can for example dynamically detect a module incompatibility that escaped testing, diagnose and fix the fault, for instance, by substituting the incompatible component with a compatible one that offers equivalent services. Depending on the problems, self-managed systems are indicated with different terms: Self-adaptive systems can adapt to environmental changes, self-configuring systems can modify the system architecture to solve configuration problems, self-optimizing systems can address performance problems by automatically optimizing resource allocation and use, self-organizing system can deal with automatic installation and deployment of new components, self-protecting system can defend themselves from malicious attacks, self-healing systems can automatically diagnose and heal different classes of faults.
Self-managed technology work without human intervention, and thus does not require expensive classic maintenance activities. Moreover, it work at run time, without requiring access to the source code, thus overcoming practical limitations of classic test and debugging techniques.

In this talk, we will analyze the new problems of service-oriented applications, we will see the limits of classic test and debugging approaches, and we will discuss the possibility offered by self-managed technologies. We will appreciate the applicability of self-healing mechanisms through some early results of an ongoing project that aims to define a design methodology to produce self-healing service oriented applications [3, 2].

References

Bridging the Gap between Model-driven for the Development and Web Project Estimation

Silvia Abrahão
Department of Computer Science and Computation
Valencia University of Technology
Camino de Vera, s/n, 46022, Valencia, Spain
sabrahao@dsic.upv.es

Developing Web applications is significantly different from traditional software development. The nature of Web development forces project managers to focus primarily on the time variable in order to achieve the required short cycle times. In this context, Model-Driven Architecture (MDA) approaches seem to be very promising since Web development can be viewed as a process of transforming a model into another model until it can be executed in a development environment.

Over the last few years, several Web development methods that provide some support to MDA have been proposed (e.g., WebML, OO-HDM, W2000, OO-H). Adopting such methods, however, poses new challenges to the Web project manager, in particular with respect to resource estimation and project planning.

A fundamental problem in this context is the size estimation of the future Web application based on its conceptual model. The functional size measurement (FSM) methods used in industry date from a pre-Web era. None of the ISO-standard FSM methods (e.g., IFPUG FPA, NESMA FPA, COSMIC) were designed taking the particular features of Web applications into account. Therefore, existing FSM methods need to be adapted or extended to cope with Web projects. Some approaches for sizing Web projects have been proposed in the last few years. The main limitation of these approaches is that they cannot be used early in the Web development life cycle as they rely on implementation decisions. Furthermore, for project estimation purposes, measurements of this type come too late.
In this talk, I will discuss the benefits and challenges of using size estimates obtained at the conceptual model level of a Web application. In particular, I will show how measurement procedures for Web size estimation can be defined as a mapping between two metamodels: the metamodel of an ISO-standard FSM method and the metamodel of a Web development method. The definition and automation of a model-driven measurement procedure for Web applications using this approach will be presented in detail, as well as its use for Web effort and productivity estimation. Several aspects of the validation of FSM methods will also be analyzed. The talk will finish by showing the research directions that can have a significant impact on the estimation of Web projects.
Analyzing Trackback Usage as a Inspection of Weblog Data Quality for Weblog Mining

Shinsuke Nakajima¹, Katsumi Tanaka² and Shunsuke Uemura³

¹ Graduate School of Information Science, Nara Institute of Science and Technology. 8916-5 Takayama-cho Ikoma Nara 630-0101, Japan
shin@is.naist.jp

² Dept. of Social Informatics, Kyoto University. Yoshida Honmachi Sakyō-ku Kyoto 606-8501, JAPAN, ktanaka@i.kyoto-u.ac.jp

³ Faculty of Informatics, Nara Sangyo University. 3-12-1 Tateno-Kita, Sango, Ikoma, Nara 636-8503, Japan, UemuraShunsuke@nara-su.ac.jp

Abstract. Recently, blogs have become widely used as tools for putting out information quickly and easily. Thus, the Web has become not only a place for getting information, but also a place for communication. It can be said that blogs have changed the way people use the Internet and become the mirror of public opinion. Some researchers do blog analysis such as blog community analysis and reputation analysis using blog data. It is known that the link structure among blog entries considerably influences the formation of blog communities on blogspace. Thus, it is very important to investigate hyperlinks and trackback links in order to understand characteristics of blog communities and blogger behavior. However, most researchers do not focus on trackback links, despite their importance in understanding the relations between blog entries. Therefore, we analyze trackback usage in order to inspect weblog data quality for weblog mining and investigate their importance in understanding blogspace for blogger behavior. According to our analysis, we have realized that most existing trackbacks are blank-trackbacks that differ from the definition of weblog trackback. We will also discuss relationship between blog entries connected via trackback link.

1 Introduction

Recently, blogs have become widely used by general users as tools for putting out the information quickly and easily. According to a report[1] of Japanese Ministry of Internal Affairs and Communications (MIC) in May 2005, The cumulative number of bloggers (Internet users who maintain their blogs) in Japan is about 3.35 million (when considering bloggers who maintain two or more blogs, the net number of bloggers is about 1.65 million.) as of the end of March 2005. The MIC Study Group forecasts that by the end of March 2007, those numbers will increase to about 7.82 million and about 2.96 million, respectively. Thus, Web space becomes not only a place for getting information but also a place for
communication. It can be said that blogs have changed the way people use the Internet.

In Blogspace, general users can be not only contents consumers but also contents providers. We may say that blog contents are the mirror of public opinion. Consequently, it is reasonable to suppose that the importance of blog information is getting bigger.

In fact, some researchers do blog analysis such as blog community analysis and reputation analysis using blog data. It is known that the link structure among blog entries considerably influences the formation of blog communities on blogspace. Thus, it is very important to investigate hyperlinks and trackback links in order to understand characteristics of blog communities and blogger behavior. However, most researchers do not focus on trackback links, despite their importance in understanding the relations between blog entries. The reason why they do not focus on trackback is likely that it is not clear the significance and the meaning of trackback in Blogspace.

The “The Motive Internet Glossary [2]” says that

Trackback is a standard that can be used to automatically create a link between webpages (reciprocal link), usually between webpages on different websites.

Namely, A trackback is a function that can create a link from another blog page to user’s own blog page independent of an intention of the another blog author. According to the above definition, trackbacks should exist with the opposite hyperlink in pairs. However, trackback links are automatically created by sending trackback ping, even if there are not the opposite hyperlinks. Actually, there exist such “blank-trackbacks” whose opposite hyperlinks are blank. Therefore, the purpose of this study is analyzing trackback usages to inspect weblog data quality for weblog mining and investigating its importance in understanding blogspace.

We first describe blogs and trackbacks, and related work. This is followed by a analyzing and considering of how trackbacks are used. We end with a summary and outline our plans for future work.

2 Blogs and Trackbacks

A blog entry, a primitive entity of blog content, typically has links to web pages or other blog entries, creating a conversational web through multiple blog sites.

Figure 1 shows a schematic of a typical blog site. A blog site is usually created by a single owner/blogger and consists of his or her blog entries, each of which usually has a permalink (URL: uniform resource locator) to enable direct access to the entry. Blog readers can discover bloggers’ characteristics (e.g., their interests, role in the community, etc.) by browsing their past blog entries. If readers know the characteristics of a particular blog, they can expect similar characteristics to appear in future entries in that blog.
Figure 1 shows concept diagram of a typical trackback link.

Figure 2 illustrates a typical trackback link with hyperlink.

Originally, A trackback is a function that automatically create a link from another blog page to the user’s own blog page when referring to another blog page. A typical procedure of creating a trackback is shown below:

1. A user refers to a blog entry.
2. The user send trackback ping to the referred blog entry.
3. The referred blog system create a trackback link from the referred page to the referring page automatically.

Next, Figure 3 indicate concept diagram of a blank-trackback. As Figure 3 shows, trackback function automatically create a trackback link to when receiving trackback ping even if there is no the opposite hyperlink in pairs.
3 Related work

In related work on analyzing blogspace, Kumar et al. studied the burstiness of blogspace\cite{3}. They examined 25,000 blog sites and 750,000 links to the sites. They focused on clusters of blogs connected via hyperlinks named blogspaces and investigated the extraction of blog communities and the evolution of the communities.

Gruhl et al. studied the diffusion of information through blogspace\cite{4}. They examined 11,000 blog sites and 400,000 links in the sites, and tried to characterize macro topic-diffusion patterns in blogspaces and micro topic-diffusion patterns between blog entries. They also tried to model topic diffusion by means of criteria called Chatter and Spikes.

Adar et al. studied the implicit structure and dynamics of blogspace\cite{5}. They also examined both the macro and micro behavior of blogspace. In particular, they focused on not only the explicit link structure but also the implicit routes of transmission for finding blogs that are sources of information.

Nakajima et al. studied how to discover important bloggers by analyzing blog threads\cite{6}. They proposed a method of discovering bloggers who take important roles in conversations and characterized bloggers based on their roles in blog threads (a set of blog entries connected via usual hyperlinks). They considered that these bloggers are likely to be useful in identifying hot conversations.

However, their purposes were not to analyze trackback usages and to investigate its importance in understanding blogspace.

4 Analysis of Trackback Usages in Blogspace

4.1 Crawling through blog entries and extracting trackback links

The system crawls through RSS feeds registered on the RSS list and registers permalink of blog entries. Our RSS list have been created based on PING.BLOGGERS.JP\cite{7} that open RSS feeds of JP domain to the public.

We need to extract the trackback links from html files of blog entries that have been crawled already. Therefore, we have to be able to recognize the scope
described the trackback data, based on an analysis of the HTML tag. However, each blog site server has its own tag structure so we need to set up rules for analyzing the tag structure of each blog site server that we want to analyze. By using the rules, we extract data of trackback links that are starting URL, destination URL and time stamp of the trackback link. Our target blog sites are limited to famous blog-hosting sites because naturally we are unable to set up rules for every blog site. We therefore set up rules for analyzing the tag structure of about 17 famous hosting sites of JP domain. We call them “blog sites for the analysis.” We use 10,683,678 blog entries in “blog sites for the analysis” published from October 2005 to January 2006.

4.2 Relationship between entries connected via trackback links

Link-base relationship According to original definition of trackback, trackbacks should exist with the opposite hyperlink in pairs. We have investigated actual condition of link-base relationship between entries.

![Fig. 4. Representation of a hyperlink and a trackback link](image)

First, We examine the link-base relationship between entry(x) and entry(y) when existing trackback link (TB(x-y)) from entry(x) to entry(y). As indicated in Figure 4, TB(x-y) corresponds to a trackback link existing in entry(x), and it is created when receiving trackback ping from entry(y). link(y-x) corresponds to a hyperlink from entry(y) to entry(x).

Table 1 shows link-base relationship of hyperlinks and trackback links. In this result, all patterns of existence of TB(y-x), link(x-y) and link(y-x) are investigated when a TB(x-y) exists. “link(x-y)O” denotes that link(x-y) exists. “link(x-y)X” denotes that link(x-y) does not exist.

In this case, blank-trackback means a situation that link(y-x) does not exist. It exists 99.08% of all patterns. In fact, almost all patterns in blog sites for the analysis are blank-trackback. Thus, the latest situations of trackback usages are different from the original definition of trackback.
Moreover, another result that we focus on is the mutual blank-trackback relationship which is a situation that both TB(x-y) and TB(y-x) exist. This situation totally exists 11.88% of all patterns.

We may suppose that blank-trackbacks are kinds of spams, because a purpose of creating a blank-trackback may be to get more inlinks without making outlinks in order to get higher PageRank and more web users visiting own web page.

However, it is quite likely that bloggers give recognition each other in the mutual blank-trackback relationship. In this case, blank-trackback may not be spam.

Therefore, we are examining content-base relationships between entries that have blank-trackback relationship in the next chapter.

**Content-base relationship** In this section, we investigate content-base relationship between entries that have a blank-trackback link. The target data are 100 pairs of entries that have a blank-trackback. They are picked up at random from entries in the term from October 2005 to January 2006. The investigation is based on human judgment. The tester browse both contents of starting URL and destination URL of blank-trackback. According to trackback definition, a blogger who send a trackback ping should mentions in his/her blog entry about the content of blog entry that receive the trackback ping. Thus, the tester investigates which a blogger has mentioned about target blog entry of blank-trackback, or not.

Table 2 shows content-base relationship between entries that have blank-trackback. In all of the cases in Table 2, there exists TB(x-y).

<table>
<thead>
<tr>
<th></th>
<th>TB(y-x)O</th>
<th>TB(y-x)X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>link(y-x) O</td>
<td>link(y-x) X</td>
</tr>
<tr>
<td>link(x-y)O</td>
<td>0.03%</td>
<td>0.29%</td>
</tr>
<tr>
<td>Related O</td>
<td>0%</td>
<td>7%</td>
</tr>
<tr>
<td>Related X</td>
<td>-</td>
<td>4%</td>
</tr>
</tbody>
</table>

“Related O” or “Related X” denotes yes or no for existences of relation of the contents (Topics) between entries.
“Mention(y-x) O” or “Mention(y-x) X” denotes yes or no for existences of mentioning entry(x) in entry(y) that send trackback ping to entry(x).

As Table.2 indicates, there exist 33% of trackback links that point to unrelated entry. Actually, all of them are adult sites and spams indeed. Moreover, 4% in this 33% cases have mutual trackback relationship. In this case, all pairs of entries are mutual trackback-spams indeed.

We have recognized existences of trackback-spam. However, the percentage in number of blank-trackbacks is only 33%. Thus, we cannot say that blank-trackbacks are always spams.

In addition, there exists 57% of trackbacks which have no-mention but related to the opposite entry in pairs that is the target of trackback ping. Actually, they have the same topics. For example, review of movies and books, forecasting of horse racing, and so on. However, they do not mention contents of other entries at all. This kind of entries often has not only one way trackback but also mutual trackback. It seems that they form a “soft” blog community based on those trackback connection. Their connections are not strong and explicit.

4.3 Difference between usual hyperlinks and trackback links

Table. 3 shows a number of hyperlinks and a number of trackback links against total number of entries (10,683,678 entries) in “blog sites for the analysis.” Now, we consider hyperlinks only appearing in the blog-description written by the blogger, except automatically-created links to access the past entries and commercial links, and of course except trackback links.

Table 3. Comparison between number of links and number of trackbacks

<table>
<thead>
<tr>
<th></th>
<th>Num. of Links</th>
<th>Num. of Links / Num. of Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usual Hyperlink</td>
<td>4,613,581</td>
<td>0.432</td>
</tr>
<tr>
<td>Trackback link</td>
<td>1,696,177</td>
<td>0.159</td>
</tr>
</tbody>
</table>

As shown Table.3, an average number of hyperlinks per number of entries is 0.432 from October 2005 to January 2006, and an average number of trackbacks per number of entries is 0.159 in the same term. We can say that trackbacks become important to connect blog entries each other though the hyperlinks are still mainstream of connecting blog entries.

Figure.5 shows number of entries vs. the number of links contained in each entry. Figure.6 shows percentage of entries vs. the number of links contained in each entry.

As shown in Figure.5, most of entries containing hyperlinks (or trackback links) have less than 5-10 hyperlinks (or trackback links). The trends in the distribution between hyperlink and trackback link are similar though the number
of hyperlinks is more than the number of trackback links. As shown in Figure 6, percentages of entries that have one or two trackback links are higher than the case of usual hyperlinks. It seems that connections via trackback links between blog entries are stronger than connections via usual hyperlinks because most of trackback links are often created by several particular bloggers.

Next, let’s discuss difference between hyperlink-base blog communities and trackback-base blog communities. At beginning, we explain blog thread regarded as temporal blog community.

An example of a blog thread is shown in Fig. 7. We define a blog thread as follows. A blog thread is composed of entries connected via links to a discussion among bloggers. Namely, a blog thread is a directed connected graph and is defined as follows.

$$\text{thread} := (V, L)$$

$V$ is a set of blog entries.
$L \subseteq \{(e, e') | e \in V, e' \in V\}$
$L$ corresponds to a set of links

Ideally, the entries in a blog thread should share common topics. The blog threads seem to be blog communities formed of blog entries that have similar topics via links.

Table 4 indicate numbers of hyperlink-base threads and trackback-base threads. In this investigation, we use blog entries published in October 2005 and November 2005 and their hyperlinks and trackback links. We can see the result in the cases that threads have more than 50 entries, more than 30 entries and more than 10 entries in Table 4.

As indicated in Table 4, the number of trackback-base threads is more than the number of hyperlink-base threads in all of three cases. As shown in Table 3, we may say that trackback links have about 3 times abilities to form blog threads.
Fig. 6. Percentage of entries vs. the number of links contained in each entry

Fig. 7. Example of blog thread

as strong as hyperlinks because the number of trackback links is one third of the number of hyperlinks. Therefore, the trackback links are very important to investigate relationship between blog entries by analyzing blog communities.

4.4 Considerations

– Differences between original definition and actual usage of trackback
  As mentioned above, blank-trackback, whose opposite hyperlinks are blank, exists about 99% of all patterns. Moreover, blank-trackbacks are not always spams and they can form a “soft” blog community. It is different from original definition of trackback. It seems that we should re-define what trackback is, because such undefined usage becomes majority of trackback usages, at least for JP domain.
– Existences of trackback spams
Table 4. Comparison of number of threads between Hyperlinks and Trackbacks(Oct. 2005 and Nov. 2005)

<table>
<thead>
<tr>
<th></th>
<th>more than 50 entries</th>
<th>more than 30 entries</th>
<th>more than 10 entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of threads (via hyperlink)</td>
<td>51</td>
<td>145</td>
<td>1,032</td>
</tr>
<tr>
<td>Num. of threads (via trackback)</td>
<td>74</td>
<td>163</td>
<td>1,243</td>
</tr>
</tbody>
</table>

There exists 33% of trackback links that are spams indeed. It is quite likely that the trackback spams may become cause of great error when analyzing blog data. However, it is a problem of not only trackbacks but also Web itself. As Table 3 indicates, the trackback links are very important to form blog communities. Therefore, we had better consider trackbacks with developing spam filtering in order to analyze relationship between blog entries.

- **Blog communities based on trackbacks**

  As mentioned above, trackback links have about 3 times abilities to form blog threads as strong as hyperlinks. The blog threads are regarded as temporal blog communities. Actually, characteristics of trackback-base communities are “soft” communities and a little different from characteristics of hyperlink-base communities. Accordingly, the trackback links are very important to discover and analyze blog communities.

5 Conclusions

In this study, we have analyzed trackback usages and investigating its importance in understanding blogspace, for understanding blogger behavior.

The results of this study can be summarized as follows:

- We have set up rules for analyzing the tag structure of famous hosting sites of JP domain and have analyzed trackback links
- We have investigated blank-trackback, whose opposite hyperlinks are blank, exists about 99% of all patterns.
- We have investigated that existences of “soft” blog communities based on blank-trackback connection.
- We have investigated difference between a number of hyperlinks and trackback links, and investigated ability to form blog threads. As a result, we have examined an importance of considering trackback when analyzing blog communities.

In addition, in future work we plan to investigate trackback data of blog entries in the other domain for understanding blogger behavior.

6 Acknowledgments

This research is partly supported by MEXT (Grant-in-Aid for Scientific Research on Priority Areas #19024058).
References


2. The Motive Glossary - trackback
http://www.motive.co.nz/glossary/trackback.php


Improving the Quality of Website Interaction
with Lightweight Activity Analysis

David Nutter, Cornelia Boldyreff, and Stephen Rank
{dnutter,cboldyreff,srank}@lincoln.ac.uk
University of Lincoln

Abstract. Understanding user interaction with websites is at present a
black art requiring either significant investment in initial development or
later re-engineering of websites to track individual users reliably or time-
consuming surveying approaches. A lightweight system for identifying
common interactions with websites such as uploading a file or logging in
using only webserver logs is proposed; and an associated tool which can
automatically extract such information from webserver logs and visualise
such interactions as graphs is discussed. Using this tool, the web-based
CALIBRE Work Environment (CWE) has been evaluated and several
improvements based on the findings have been made.
Such an approach has advantages over existing website evaluation meth-
ods, primarily, ease of deployment against unmodified sites and web-
servers or archives of historical data. For thorough website evaluation,
the activity analysis approach must be a part of a larger strategy; how-
ever, it can represent a first step on the path towards improved quality
for the users interacting with the website.

1 Introduction

Evaluation is an important activity throughout—and not simply at the end of—
the software lifecycle [1, 2, 3]. Such evaluation is a key activity to assure software
quality in the widest sense of fitness for purpose. Our earlier work focussed on the
development of software applications, from scratch or from discrete components.
This paper applies the same principles to the development, maintenance and
operation of a web-based collaborative environment built using Zope and Plone.
Three key factors in the evaluation of website quality are performance, security,
and usability [4]. This study’s primary focus is website usability.

The CALIBRE project was a 2-year coordination action funded under EU
Framework 6. It aimed to promote best practice in the use of libre software in
the secondary software sector (e.g., telecoms, automotive etc) and to conduct
systematic research into libre software topics including process, business models,
efficacy and dependability issues. The project had 12 partners in 9 countries. To
support collaboration between the partners, the CALIBRE Work Environment
(CWE) [5] was created. The CWE’s main aim was to support collaborative
research. The key collaboration required was in the production and dissemination
of research outputs (such as research papers and formal project deliverables), augmented with web-based discussion forums for each work package.

The CWE is a web-based system which supports simple content management, wiki editing, access control based on groups of users (Plone WorkGroups), file upload, event management and news management. The site was built using Zope, Plone, and a collection of third-party plugins for the latter. Very little development effort beyond integration was required to assemble the initial system.

The development and operation of the system was part of the contractual obligations of the project (under “Dissemination”). Therefore several requirements for its evaluation existed. Firstly, the impact of the system on the target demographic must be assessed and reported back to the European Commission. Secondly, the usability of the system must be monitored and any problems found addressed to ensure that the CWE is meeting the users’ needs. In the case of the CWE, usability was seen as a key quality factor.

Web servers already gather detailed logs of resource accesses and error messages, and the potential of mining these logs to support site evaluation has been noted [6, 7] in the past. As an initial attempt at addressing the first requirement, two simple monitoring tools—analog and reportmagic—were deployed to statically analyse the webserver log files. These tools produced simple reports such as the number of requests from particular organisations, the most popular files and directories, the user agents employed by clients and so forth. While useful for studying the user base en masse, these reports do not allow developers to study the activities of a particular subset of users, nor identify problematic areas of the site and therefore the second goal, of improving usability, could not be achieved using such reports. Therefore, a smarter analysis method was deemed to be required, one which allows the analysis of the activities of individual users.

There is an extensive base of literature addressing web usage mining for a variety of purposes: website adaptation [8, 9], target group identification [10], personalisation [11], system improvement, site modification, business intelligence, and usage characterisation [12]. Sophisticated data mining and clustering techniques have been applied to discover web usage patterns; usually some form of sessionalization of web server data or session tracking is a necessary step in the mining processing.

2 Activity-Based Session Tracking

Session tracking allows the identification of individual usage sessions and thus supports the analysis of the activities of individual users. By default, webserver logs are request-based and do not track “sessions” associated with individual users. Several approaches to adding session information to logs have been discussed in the literature, including:

User cookie Each unique user is issued a cookie to be presented to the website on return visits. For those users that accept and retain the cookie, this method provides a very accurate method of tracking their activities.
The Apache webserver transparently provides such functionality using the `mod_usertrack` module, which annotates the log file with unique user identifiers.

**Special URL** Instead of explicitly issuing a cookie, the site’s URL is modified to contain a unique identifier. While this method can track all users, it cannot maintain tracking between different web browser session as the cookie can. Such functionality requires modification of the web site itself and in our case modifying Zope and Plone to provide such capabilities was deemed too difficult in the time available.

Both these methods require intervention in the application at an early stage of development to enable the necessary tracking technologies. This is of little use when one wishes to “mine” [13] existing web server log files that do not contain the necessary information. However, another method exists, whereby requests are heuristically grouped by time and other factors into distinct sessions [14, 15, 16]. This has the additional advantage of working on unmodified web-servers and without compromising user privacy in the manner of introduced unique identifiers. Unfortunately, such a method is unreliable as it only identifies short bursts of activity. Fortunately, these are sufficient for this project’s purposes.

Since there was no requirement for this evaluation to identify unique users, merely to study interaction with the site, identify problematic features and thereafter improve them, the shortcomings of activity analysis do not affect the work described here. The tendency of this method to identify short bursts of activity rather than a full session is also useful, as it tends to pick out particular sequences of closely-coupled actions making up a particular activities such as logging in, uploading a file etc. This allows common activities to be identified and, by tracking error conditions as well as successful requests, problematic activities can also be identified and is sufficient for our purposes.

To this end, two tools were developed. The first was a simple prototype that processed an Apache log-file, split it into logfile fragments corresponding to particular sessions, then visualised those sessions using the GraphViz tool. To adjust the visualisation, or to annotate the graph program modifications were required, making this solution tricky to use.

Consequently a new tool was required, a tool that was interactive; web-based; could filter and clean data; supported visualisation and backed with a persistent store so once a raw log file was uploaded the original file could be discarded. The intention was to provide a simple solution to exploring activity visualisations. Other more complex tools [17] do exist for visualising the structure and navigational paths users pursue through websites; however, that is not the main goal here.

### 2.1 Tool Architecture

Given the previous requirements, this new version of the activity analysis tool was designed to satisfy them. For maximum flexibility and extensibility, a pipeline
architecture was decided upon. This permits the use of simple, composable operations on a sequence of logfile entries to express complex analysis requirements.

A basic pipeline which implements a transformation between Apache format logfiles to W3C Common Logfile format is shown in Figure 1. Figure 2 shows a second pipeline configured to produce an activity graph from log records extracted from the persistent store.

At present, the following pipeline components exist. All are derivatives of the PComponent class.

LogProcessor Turns a raw logfile into a sequence of LogEntry objects using a LogFormat entry

DBAgent Interacts with a database to insert and remove log entries from persistent store.

Discarder Terminates the pipeline and discards any entries

SessionMaker Accepts LogEntry objects and attempts to group them into sessions. Consequently this object has a connection with the backing store via DBAgent allowing it to look for other entries in the same session.

LogWriter Accepts LogEntry objects and prints them out into a file using a specified LogFormat. Using a pipeline composed of a LogProcessor and a LogWriter, one can convert between two logfiles in different formats e.g. W3C to Apache style and vice versa, as may be seen in figure 1.

LogFormat Describes the format of a logfile entry in the manner of the Apache LogFormat directive.
Once sessions have been created by the SessionMaker pipeline component, they may be retrieved from the backing store and visualised as a directed graph using GraphViz [18]. A session is visualised as follows:

1. Each node represents a page within the site, or some error condition (like a 404)
2. Each edge represents a transition between pages
3. The edge weight represents the number of times the transition took place within the session

Figure 3 is one such visualisation showing the activity associated with browsing workpackages in the CWE. A and B represent the top of the tree of workpackages: the front page and the “workpackages” sub-page respectively. Users then explore deeper into the tree, entering workpackage 4 (C) using the simple view. Then it appears that certain users are content with default the simple view, whilst others require the detailed view (D) and consequently make a further transition.

In order to reduce clutter on the activity graph some information is discarded. Firstly, edges with weights less than a threshold (5 by default) are treated as noise. Secondly, GET parameters in the page URL are also discarded, leading to self-referential transitions in certain session graphs. Finally, requests for non-page objects (e.g., images, CSS files), or from known crawlers are also discarded.

At this point it should be made clear that the activity graphs emerge spontaneously after this cleaning process. The developer is only required to interpret the resulting graphs.

3 Identifying Usability Issues

The analysis requirements for the CWE are of two main types: usability improvement and impact assessment. The latter requirement is largely addressed
by the existing static monitoring system. Hitherto, the former requirement was addressed only in an ad-hoc manner when users reported problems. Therefore we analysed the collected log data for the year 2005 to identify usability issues.

As an initial step we analysed the months of March and April in 2005 to capture the typical activities. These were two fairly active months as a deliverable was due in April and therefore suitable for capturing all basic activities without analysing excessive amounts of data. Each month was analysed separately, using the default edge discard threshold of 5. The results were visualised and subgraphs corresponding to distinct activities were selected.

Prior to examining the activity graph, we first listed the activities we thought the site should support, with the intention of matching them to activity graph(s) later on. Analysis of the activity graphs is of course subjective and not directly supported by the tool. The operator must explore the activity graphs using the tool and decide what they mean. In performing our analysis we considered the following:

- Any error conditions encountered. Many transitions to a 404 error indicate a broken link, for example. 403 errors, on the other hand may be intentional—a user is attempting to access something which they are deliberately prevented from accessing.
- The prevalence of cycles in the activity graph, and whether this was due to a multi-function page controlled by form variables or user confusion.
- The expected flow through a particular feature of the site (uploading, logging in etc) and whether this matches the activity graph. Excessive deviation may indicate a problem.
- Whether activity graphs are present for all the “features” of the site. Lack of an activity graph indicates an under-utilised feature. We found that adjusting the threshold for discarding edges was helpful here as the under-used feature graphs would appear when the threshold was reduced.
- Similarly, if an activity graph is present that does not correspond to one of the previously identified features, this indicates that the site is being used in an unanticipated way and that further requirements capture from the users may be required to support and develop this new feature.

In dealing with problematic activities, we followed several strategies. Firstly, we improved documentation of common activities that confused users. Secondly, we re-engineered any features that lead to errors or could not be easily documented and finally revised the requirements of the site to reflect user needs more accurately. The next section discusses the activities we identified both before and after analysis, the problems and finally solutions we identified after analysis.

### 3.1 Results, Problems Identified and Solved

Table 1 shows some common activities identified, and whether they were known before from user requirements, after the activity analysis or both. Many other activities were identified during the course of this research but these have been omitted for reasons of space and clarity.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Before</th>
<th>After</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login/out</td>
<td>x</td>
<td></td>
<td>Users may sign in to gain access to restricted content.</td>
</tr>
<tr>
<td>Search resources</td>
<td>x</td>
<td></td>
<td>Users may search the CWE contents on keywords and full text.</td>
</tr>
<tr>
<td>Manipulate workgroups</td>
<td>x</td>
<td></td>
<td>Users may be assigned to workgroups which delegate privileges to their members making access control easier.</td>
</tr>
<tr>
<td>Post news or event</td>
<td>x</td>
<td></td>
<td>Users may place events and stories that appear on the calendar.</td>
</tr>
<tr>
<td>Upload file</td>
<td>x</td>
<td></td>
<td>Users may upload files to locations within the site and by this action make them available to others.</td>
</tr>
<tr>
<td>Single-page session</td>
<td>x</td>
<td></td>
<td>Results from users following a link in an email directly to a resource. These may be discarded.</td>
</tr>
</tbody>
</table>

Table 1. Activities

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users “returning” from logout page to other pages in the site</td>
<td>This is partially an artefact arising from the use of the “back” button in web browsers. However, the site’s user interface does not indicate which resources are restricted to logged-in users. Documentation to this effect was written.</td>
</tr>
<tr>
<td>Errors in file upload</td>
<td>Due to web server configuration restrictions, files greater than a certain size cannot be uploaded. By lifting the restriction for the CWE site, this problem was alleviated.</td>
</tr>
<tr>
<td>Users seeking the “detailed” view of folders</td>
<td>Since users were constantly switching from the simple (default) view of the folder to the more detailed view, it was clear that providing an option for users to change their default view was necessary. This was done.</td>
</tr>
<tr>
<td>Confusing workgroup functionality</td>
<td>The activity graphs related to changing or using workgroups were very confused. User feedback confirmed our suspicions that the workgroup tool was difficult to use. Documentation was improved accordingly.</td>
</tr>
<tr>
<td>Internationalisation: site does not show which translations are (un)available. The activity graphs showed users entering and then immediately leaving the empty translation pages</td>
<td>To resolve this problem, we merely changed the style of the links for translations which really were available. The “dead” links could still be followed (indeed this was necessary to allow users to edit them) but the confusion occasioned by their presence was reduced.</td>
</tr>
</tbody>
</table>

Table 2. Problems and Solutions
Table 2 shows some of the more critical problems found through this analysis and the solutions proposed and implemented to deal with them. From these tables, it is clear that the main use of this type of analysis tool is improving the usability of the site rather than spotting other problems such as security issues. Therefore activity analysis can only form a part of a larger web site evaluation and evolution strategy.

4 Limitations, Outstanding Issues and Further Work

Though the tool supports data collection, cleaning, visualisation and some graph exploration activities, it does not directly support the analysis of those graphs. Thus the time to analyse a website using these tools will increase in proportion to the number of distinct activities present. The subjective analysis of activities is of course prone to investigator bias and other errors; however, since website usability is often evaluated using subjective aesthetic notions and rules-of-thumb [19, 20, 21], this is not fatal.

Since some request data (such as GET parameters) is discarded or is never recorded in the web server log (as in the case of POST data), this tool is clearly limited to analysing transitions between distinct pages. Consequently, this tool is probably best applied to simple web applications (each “page” encapsulating a particular feature) rather than more complex applications where each page is involved in several features. Due to the paucity of information available in the web server logs this tool can never provide the analysis potential of a fully-instrumented web site.

Nevertheless, the advantages of this method are its wide applicability, flexibility and ease of use with unmodified websites and servers. In the case of the CWE, most features are sufficiently separated to allow the tool to work satisfactorily, but in the case of logs originating from another web application (Horde [22]) the lack of separation and heavy use of POST data means no meaningful activity graphs can be drawn.

Due to the grouping approach used by the SessionMaker module, it is possible for sessions to become confused where NAT gateways or HTTP proxies are used which cause requests from distinct hosts to apparently originate from the same host. Furthermore the graph visualisation does not make any attempt to mark the beginning or end of an activity, which can make interpreting cyclic activity graphs difficult.

Aside from incremental improvements to the tool interface, addition of graph analysis support to the tool may be helpful. It seems plausible that well documented and intuitive website features will generate activity graphs with certain key properties (few reversible transitions, few error states and few cycles for example). Assuming this holds, a future version of the tool could look for activity graphs which do not express these features and present them to the operator for further inspection.

Before such functionality is developed it will be necessary to investigate whether activity graphs do in fact display such “marker” properties. Two strate-
gies for doing this suggest themselves, firstly by analysing a number of existing websites and web-based applications, identifying their salient features and then selecting a sample of users to interview about those features. Using the interview results to determine which website features are difficult to use, the investigators may then determine whether the activity graphs display any marker properties that can be used to detect problematic features.

Secondly, as a follow-on to the above study, a set of simulated “good” (usable) and “bad” (unusable) website features could be created and used to test a sample of users in controlled conditions to confirm the assumption that the chosen activity graph marker properties are indicative of quality problems with the corresponding website feature.

5 Conclusions

The everyday co-ordination, collaboration and dissemination work of the CALIBRE consortium has been well supported by the evolving CWE throughout the project. The CWE web site now forms a permanent repository of the project’s work throughout its lifetime. Monitoring of the CWE using existing web site monitoring tools has given the big picture of its usage over time and more specialised monitoring of user activities has been accomplished using the tools described here.

The finer grained monitoring enabled us to identify specific usability problems and take remedial action, thus improving the CWE during the project.

The research prototype version of the activity analysis tool and glue code to run Report Magic 2.21 and Analog 5.32 and some test data are available at: http://cross.lincoln.ac.uk/projects/calibre/CWELogAnalysis.html Further development of this tool is on-going research.

References


Establishing a quality-model based evaluation process for websites

Isabella Biscoglio¹, Mario Fusani¹, Giuseppe Lami¹, Gianluca Trentanni¹

¹ ISTI (Institute of Science and Technologies in Informatics) - CNR (National Research Council), Via Moruzzi 1, 56124 Pisa, Italy
{Isabella Biscoglio, Mario Fusani, Giuseppe Lami, Gianluca Trentanni}@isti.cnr.it

Abstract. This paper presents the main aspects of an ongoing project, aimed at defining a website independent evaluation process as a part of the mission of a service-providing organization. The process uses as reference a quality model that is defined starting from existing proposals and general requirements for quality models. The problem of integrating human judgment and automation in the evaluation process is also introduced, and technical solutions, involving the use of experimental work, are discussed.

Keywords: Website quality evaluation processes, quality models.

1 Introduction

Quality Models (QMs), broadly intended as collections of expected properties of human activities (processes) and their results (products, services), have quite often been introduced in literature. The concept is no further discussed here, but is adopted as one starting point towards the derivation of basic practices (including technology and management) of an independent evaluation process for websites (intended as products with their lifecycle processes).

With this initiative our organization, the Software and System Evaluation Centre (SSEC) of the National Research Council at Pisa, that has been working for a couple of decades in 3rd party software product and process assessment/improvement, is planning to extend its activity into the domain where most applicative effort is nowadays being devoted by both mature and less mature developers. Besides the applicative and business-oriented opportunity offered, it seems that some research problems, now traditional in the software lifecycle domain, are confirming themselves in web engineering (WE), where better applicability of empirical methods stimulates spending some investigative effort.

The approach, on which our organization is investing some time and resource, is as follows.

First, an analysis of explicit/implicit QMs proposed in literature (including QMs for QMs, see [12,14]) is performed (Section 2). Then, the classic problem of expressing QM properties at various levels of abstraction, also referred as attributes or
characteristics, in meaningfully quantitative ways [7] is addressed and an experimental activity is presented to cope with this problem (Section 3).

This covers only part of the preparation work for establishing the evaluation process practices, but regards its most difficult (and interesting) step.

2 Quality Models

2.1 QMs for software products vs QMs for websites

The study of the quality characteristics of software products and their relationships has been absorbing an impressive amount of effort that can be dated back to the 1970’s [1], [15]. In spite of the huge research work spent over decades, that actually led to a better comprehension of the problems involved, no practically (industrially) satisfying solutions have been reported up to our days [24].

Some credits can be granted to one popular standard for software product quality, ISO/IEC 9126 [10] and its derivates (we recall that the six main abstract characteristics of quality are: Functionality, Reliability, Usability, Efficiency, Maintainability, Portability; plus four characteristics representing the point of view of software users: Effectiveness, Productivity, Safety and Satisfaction). The principal merit of ISO/IEC 9126 can be found in its attempt to reduce the product quality predicate to a limited number of independent characteristics, and to have developed the notion of various levels of qualities (“internal”, “external” and “in-use”). Nevertheless, such a standard was not successful in providing meaningful, quantitatively expressed (or measurable) indicators associated to quality characteristics [24].

If we want to adopt a QM for websites, how much can we import from this experience? And, are there any chances that we come out, in the more restrict WE environment, with a somewhat “more measurable” framework than in the broader Software Engineering (SE) environment?

First, we must be aware of differences and similarities between software products and websites, in the perspective of their qualities:

- In case of technical flaws in project or implementation, a website can tolerate consequent sensible loss of quality and still be operative and available. The same is not generally true for a software product: even minor defects can put it out of operation.
- Maintaining a software product is a recommendable practice while maintaining a website is just necessary to keep it alive.
- Whereas an experimental environment for analysing software products can be technically hard and expensive, it is easier and cheaper to experience the availability of websites belonging to homogeneous classes.
- In most cases, we can easily get availability of both external (behavioural) and internal (code) aspects of a website, whilst a comparable range of availability for a software product can hardly be obtained.
- For both software products and websites we can use the notions of internal, external and in-use quality levels.
- Considering development process, some typical practices or subprocesses of software development (such as, for example, configuration management) might not be equally adoptable in website development.
- Website aspects (and quality characteristics) may change during the evaluation phases [19].

The above considerations, mainly the one about availability, encourage us to design an experimental environment (section 3) to study, using statistical methods, the relationship between internal (easier to collect automatically and measure), external and quality-in-use (user perceivable and subjective) characteristics. The results of such a study are expected to give a valuable input for defining the practices of the evaluation process.

### 2.2 Adopting a Quality Model for website evaluation

Any attempt to evaluate, under any perspective, the quality of a website implies, implicitly or explicitly, a QM (implicit QMs typically exist behind evaluation methods and tools). Although our purpose is not to introduce yet another QM but to define an evaluation process, we must adopt a working QM to go on. This we do by synthesizing from existing ones.

We are not going to undertake any extensive survey of QMs proposed in the literature, but are noticing that, among the wide plethora of proposals [4], [13], [16], [17], [18], [19], [21], [23], [25], some general and systematic work do emerge, whose value is to define concepts, relationships, terminology and methods as common references [2]. This is a good basis for us to establish some entity definition criteria for our independent evaluation process. Yet this work, along with other outstanding ones for completeness of modeling [19], [20], still takes too much from ISO/IEC 9126, whose evaluation module metrics (based on elements counts and ratios) has not been proved much successful when applied to industrial environments. Also, no surveyed literature addresses the differences between SE and WE as being important for investigation (we will be possibly agreeing with this after our experiments). Most of the proposals (excepting some cautious adoption in [2]) seem to express good confidence that inter-level, quantitative relationships among characteristics can be known and used, in a way similar (although somewhat evolved) to the metrics reported in the so-called “evaluation modules” associated to the ISO/IEC 9126 [20].

In the following, a sample of just seven QMs, proposed in the last few years, that cover various points of view in observing, gauging and evaluating a website are summarized (Table I). If we try to abstract the high level concepts which the characteristics of the presented QMs refer to, it seems possible to identify a few of them, namely: Usability, Content, Navigability, Management and Relationality.

These concepts encompass characteristics which probably are not totally mutually independent; it is possible in fact that several characteristics, though presented with different denominations, have similar meaning or recall the same concept; rarely the different QMs use the same terms for semantically equivalent characteristics: perhaps only the Content characteristic is a sort of agreed one, probably because its meaning is
less controversial. An extensive application of the ontology proposed in [2] could solve all the related ambiguities. So we have to recall the definitions of the characteristics reported in Table I.

Usability is “The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments” [8]; this concept is recurrent when the authors make implicit or explicit reference to an efficient, effective and satisfactory use of the web site.

Content is considered a component that identifies what is contained in the site, and has its further characterisations (as “sound”, “original”, …).

Navigability is used to underline the ability to exploit the relationships among the elements (pages, images, ...) which compose a site.

The concept of Management recalls the set of the activities that allow full operability of the site and that include the maintenance finalized to stability and evolution, good operation of the site, including protection of privacy and security.

Relationality is related to the process through which two or more entities act to reciprocally modifying their state, and is used as Identification and as Interactivity.

Table I. Example of Quality Models and Related High-Level Characteristics.

<table>
<thead>
<tr>
<th>Model ID and Ref.</th>
<th>Usability</th>
<th>Content</th>
<th>Navigability</th>
<th>Management</th>
<th>Relationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>2QCVOQ (7 Loci) [17]</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Comprehensive [23]</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Exciting [25]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minerva [18]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QEM [19]</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>QWEB [21]</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

The semantics associated to the above characteristics, and to others proposed in literature, depends on the category of the websites and on the actors involved (site owners, site developers and site users, where each type of actors is conceivable at various levels of involvement). As providers of an external service, we suppose that our best category target is among commercial sites. Then, site owners and site users play the role of suppliers and customers, respectively, and their mutual relation is a commercial one: Ideally, the supplier wants the site being able to perform the transfer of maximum perception of the value of the goods or services offered, possibly enhancing this transferred value perception. This may change the semantics of the same characteristics for another category of site. Postponing further abstraction level adjustments, we initially adopt as characteristics the above ones plus the explicitation of the Correctness of the source code (that impacts in various, difficult to quantify ways, into other characteristics) and Accessibility (that is requested by compliance to public guidelines). As mentioned in next Section where we introduce our experiment, we may note that the completeness (not even the composition) of this set of characteristics is not an issue for our purposes: we can complete the set while or after analysing intra and cross correlations of internal and external characteristics (Sections 3.2 and 4).
3 Preliminary work for a website evaluation service

To establish a quality-model based evaluation, a set of criteria and actions aimed at finding, in the object under examination, evidence of the desired quality characteristics must be defined. Such actions include procedure execution that in turn may include objective measurements that can be automated and some intervention of human, subjective judgments that can not. Management practices and procedures are equally important to achieve the goal, but we are not dealing with these in this paper.

3.1 Problems found in establishing an evaluation process

The basic requirement for an evaluation process is to be able to quantitatively determine the degree of presence of each quality characteristic of the model in the product under analysis. Other requirements (such as objectivity, cost effectiveness, maintainability, repeatability) are related to the means for satisfying the main requirement and to the results of the evaluations. We just report here a challenging aspect of the problem.

Table II. Example of Lower-Level Characteristics.

<table>
<thead>
<tr>
<th>Lower Level Characteristics</th>
<th>CAR MAKER 1</th>
<th>CAR MAKER 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Links Mapped</td>
<td>2646</td>
<td>1341</td>
</tr>
<tr>
<td>Time Elapsed (DD:HH:MM:SS)</td>
<td>0:00:20:34</td>
<td>0:01:25:32</td>
</tr>
<tr>
<td>Total DL Time (DD:HH:MM:SS.ms)</td>
<td>0:00:13:31.947</td>
<td>0:00:53:32.581</td>
</tr>
<tr>
<td>Total Bytes Downloaded</td>
<td>15,325,809</td>
<td>10,646,007</td>
</tr>
<tr>
<td>Average Download Rate (bytes/sec)</td>
<td>18875.4</td>
<td>3313.8</td>
</tr>
<tr>
<td>Depth Reached</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Total Unique URLs on Site</td>
<td>684</td>
<td>555</td>
</tr>
<tr>
<td>Broken Links &amp;/or Unavailable Pages</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Excluded URLs</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Pages Loading Slower than 3s</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td>Pages Larger than 1024 bytes</td>
<td>453</td>
<td>513</td>
</tr>
<tr>
<td>Pages Older than 24 Hours</td>
<td>219</td>
<td>41</td>
</tr>
<tr>
<td>(Unique) Off-site Links</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>Metrics for Pages larger than 1KB</td>
<td>453</td>
<td>513</td>
</tr>
<tr>
<td>Average Links per Page</td>
<td>3.87</td>
<td>2.42</td>
</tr>
<tr>
<td>Average Bytes per Page</td>
<td>22,406.15</td>
<td>19,181.99</td>
</tr>
<tr>
<td>Average DL Time (ms/page)</td>
<td>1187.06</td>
<td>5788.43</td>
</tr>
<tr>
<td>Broken Links per Page</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Slow Pages Visited</td>
<td>5.85%</td>
<td>7.93%</td>
</tr>
<tr>
<td>Large Pages Visited</td>
<td>66.23%</td>
<td>92.43%</td>
</tr>
<tr>
<td>Old Pages Visited</td>
<td>32.02%</td>
<td>7.39%</td>
</tr>
</tbody>
</table>

As pursuing objectivity is a goal for any evaluation process, one might think that a set of extensive measures, covering all the scope of the qualities, would make the job. Regrettably, what is more easily measurable is a number of lower-level characteristics whose quantitative relationships with the external characteristics can hardly be known, even if hypotheses about have been made in [15] and in successive works. An
example of such lower-level characteristics is represented in Table II, where the values are obtained using a commercial tool [5].

Another problem is typical of services that must be self-sustaining, and is represented by the cost of the evaluation process. Directly analysing higher-level (external and quality-in-use) characteristics is mostly thorough, checklist-assisted judgment work, and measuring usually is to map a sort of degree of presence of the characteristic to ordinal scales. Automation here intervenes in checklist managing and result reporting, and not in the very measuring act. This makes the job rather expensive.

In software products there has been a nice deal of confidence on the causal relationships between lower-level and higher-level characteristics, and, as we have already observed, this attitude has been preserved in websites as well [10], [19], [20], [2]. We want to approach the investigation from another point of view.

3.2 Some features of the approach

The approach is partly based on conducting experiments that exploit the practically unlimited availability of websites and the accessibility to their internal technicalities. Tool-aided, extensive measurements are being executed on homogeneous website categories, to collect a set of lower-level characteristics such as those shown in Table II. Another data collection is going to be started on the same sample, this one manual and checklist aided, oriented to collect higher-level characteristic ratings according to the QMs shown in Table I. A database is in construction, to be populated with all these data. Each record of the database has a field subset corresponding to lower-level characteristics, and another subset corresponding to higher-level ones. Once the database has been populated, statistical analysis will be performed to find whether or not non-casual relationships exist between lower-level indicators and higher-level ones.

Any significant relationship found can be used to lower the cost of the evaluation, as part of the manual analysis would be corroborated or even substituted by the tool-based, automated analysis.

4 Conclusions and planned work

As said in Section 3, we decided to use a browser-based commercial tool, able to collect and report a huge amount of metrics [5]. Data collection on public and commercial sites is now in progress (Table II shows an example). Checklists are being generated from the QM characteristics shown in Section 2, some of them split in (one-level) sub-characteristics. Checklist construction for software products and processes analysis has been an intensive activity of the SSEC for two decades, and we are confident that a working version can be ready in a few months. The method for statistical analysis has not been established yet, but we think of using Factor Analysis.

If no significant relationship can be found, checklists will be used anyway, and the results from the tool will be interpreted by using experience and common-sense
reasoning. Also, we think that we could use count-based metrics as proposed in the Annexes of ISO/IEC 9126 and shown as an example of usage in a well-defined measurement framework in [2], [20], being aware of their un-meaningfulness risks. Such metrics could as well be validated with the experimentation results.

We want to point out again that for the experiment we may choose an extended, possibly quasi-redundant, set of higher-level characteristics, much taking from what has been proposed in literature (Section 2). Our final QM will be adjusted according to the experimental results.

Another feature to be added to our evaluation process is concerned with the lifecycle processes for websites. In fact, our relationships with the site owners must be complemented with other stakeholders (typically, requirements analysts, designers, developers). The experience of SSEC with software lifecycle process definition, started in 1993 with the SPICE project to support the ISO/IEC 15504 standard development [11] and continued with tens of process assessments [6] can be used in the WE domain. We think that the process set should be changed, possibly reduced and adapted to WE. (the SPICE framework proved to be well adaptable to other, even quite different, application domains [22], [3]).

Then, in terms of reference and supporting standards, our evaluation process would take from both ISO/IEC 14598 [9] for assessing WE products and from ISO/IEC 15504 for assessing WE processes. Which is an ambitious but workable program, also allowing for service scalability.

References

5. eValid - http://www.soft.com/eValid/
http://eprints.biblio.unin.it
http://www.hicss.hawaii.edu/
Subjectivity in Web site quality evaluation: the contribution of Soft Computing

Luisa Mich

Department of Computer and Management Science, University of Trento
Via Inama 5, 38100 Trento, Italy
luisa.mich@unitn.it

Abstract. In this paper we investigate the problem of the subjective nature of some features of a Web site and of the decisions related to an evaluation of its quality. The goal is to analyze in which steps of the quality evaluation process Soft Computing (SC) could help to address them. The viewpoint is that of a Web engineer who wants to answer that question raised by a SC expert. Referring to a general Web quality model, some preliminary consideration are given as a first step toward the applications of SC techniques for specific evaluation tasks.

1 Introduction

One of the most critical issues regarding quality has to do with the need to define models, norms and standards [1]. The aim is to have a series of conceptual and operative tools - and therefore methodologies - that facilitate the realization of high-quality products and services (see for example, [2]). The complexity of Web sites means that it is not an easy task to define a methodology for the design and evaluation of Web site quality. Indeed, the challenge lies not only in the inherently systemic nature of Web sites but also in the variety of possible target viewers and users. One of the major difficulties in the identification of the characteristics to consider when defining quality in Web sites is found in the presence of components that are intrinsically subjective. In fact, alongside technological components to evaluate - hardware, software, networks - there are other elements such as graphic design, page layout, the effectiveness of communication, etc.; for these elements the definition of evaluation criteria must take into account subjective and qualitative aspects. To take into account the subjectivity related to both the quality evaluation models and their application it is necessary to adopt an approach that makes it possible to manage qualitative aspects - and as such imprecision, uncertainty, partial truth and approximation. In a methodological sense, responding to these requirements implies a change in logic and in computational models, changing from a crisp approach to one that is more fuzzy [3], [4], [5]. In more general terms, we can refer to the methods of Soft Computing (SC), where, apart from Fuzzy Logic, the principal constituents are Neural Computing, evolutionary Computation, Machine Learning and Probabilistic Reasoning [6]. The goal of this paper is to identify the aspects and the tasks of the
Web site quality evaluation process that could benefit from the application of SC techniques. Specifically, we start from the viewpoint of a Web engineer who is considering which activities related to design and evaluation of Web site quality would gain the most from cooperation with an SC expert. The ultimate aim is to have an approach to Web site quality evaluation that is more flexible and robust and that makes it possible to manage one of the most important trade-offs in Web site quality: the need to define standards of reference in the presence of factors that are by nature subjective and difficult to measure.

The paper is structured as follows. The next section provides a definition of the concept of quality and the specific aspects of quality in Web sites, so as to expose the relativity of the concept. The third section introduces a general methodology of Web site evaluation as a conceptual framework to identify the activities in which SC can be applied to consider also that information which is subjective and incomplete, thereby improving the efficacy of the evaluation projects. The concluding section summarises the findings that emerged and which are relevant for the application of SC to the evaluation of the quality of Web sites.

2 Quality and Web sites

2.1 The concept of quality

When speaking of standards and a definition of quality it is a good idea to refer to ISO (International Standard Organization) norms [7], where we find the following definition: “Quality is the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.” [8]; and the most recent description of quality as the “Degree to which a set of inherent (existing) characteristics fulfils requirements.”[9]. Both of these definitions link the concept of quality with the satisfaction of needs or requirements [10] and, implicitly, presuppose the existence of subjects that have such needs. Moreover, both definitions emphasize that quality is related to the characteristics of an entity. The basic difference between the two definitions is the notion that not all needs are explicit (first definition) and the focus on inherent or existing characteristics (second definition). This preliminary analysis of the concept of quality allows us to put forth the following general considerations:

- The principal attribute of quality is its subjectivity, that is, quality is not an absolute concept nor an intrinsic attribute of a given entity; it depends instead on the context and on the aims and needs to be met, and can be defined and evaluated only in relation to these.
- The needs of users and the characteristics of the entity change over time, with a rate of variability that increases continuously [11]. This second aspect implies the need to provide for a periodic quality evaluation process.
2.2 Quality in Web sites

Three essential elements emerge from the definition of quality; necessary for quality evaluation, they require:

- a characterization of the entity that will undergo a quality evaluation (models);
- identification of needs that the entity must satisfy (stakeholder and requirements);
- verification of the degree to which the entity meets these requirements (methodology of evaluation).

These three points are covered in greater depth in this section, where we aim to build a framework in which it is possible to identify areas where SC could be used to improve the process of design and evaluation of Web sites. Web sites differ from conventional software systems [12], [13] and the aspects that generally characterize Web sites can be traced to four fundamental facts:

- **Presence of several diverse components in a Web site** Web sites contain elements that go beyond the traditional components of software systems. They require both a multidimensional, systemic approach and a multidisciplinary development team.
- **Variety of stakeholders** The design and use of a Web site involves a wide spectrum of actors or “stakeholders” both internal and external to the company. Each stakeholder has his own viewpoint on and expectations of the site.
- **Strategic role of Web sites** Given the high level of competition existing on the Web, simply being present on-line does not guarantee that a site’s sponsors will reach their objectives for and through the site.
- **Market and technological evolution/changes** The pressures of time and continuous changes in the market and technological environment call for innovative solutions to maintain competitiveness, thereby imposing ever tighter demands on time and resources.

More specifically, the presence of several diverse components in a Web site implies that an adequate model for the sites (point a) must take into account all of them, involving not only people skilled in ICT but also expertise in business, marketing, creative design, and of the field or domain itself. In general this implies having fairly complex models to evaluate quality, and at a conceptual level this is the reason why there is such a large number of models (a workable classification is given in [14]). In fact, when evaluating Web site quality, one of the most critical decisions lies in the choice of a model [15]. For our purposes - to identify the critical points where Soft Computing can be useful - we introduce a meta-model which serves as a common conceptual foundation for this “feasibility” study.

As regards the second point, identifying the needs the Web site must respond to (point b), all of the already mentioned aspects that characterize Web sites - the presence of several diverse components, variety of stakeholders, strategic role, market
and technological evolution/changes – must be considered in order to have an adequate definition of requirements, which we refer to as all the needs and aims described by all stakeholders. Thus the initial need to identify the stakeholders. There are three principal roles which are involved in the development and use of a site: the owner of the site, the user, and the developer. Each of these have different expectations of the site and attribute different degrees of importance to different elements:

- The owner (one or more) of the Web site focuses principally on the aims to be reached by means of the site.
- The users: Web sites have a potentially wide and differentiated target consumer/user base.
- Technical as well as non-technical developers contribute to the site’s development.

The fundamental step in any quality evaluation project is to verify whether and how much the entity – in our case the Web site – is found to satisfy the requirements of stakeholders. Herein lies the need to deal with the qualitative and subjective aspects of evaluation. As mentioned, the complexity of Web sites means that in order to have an adequate model of their characteristics it is often necessary to consider qualitative aspects, which are impossible to measure with precision. Statistical metrics could also be introduced for Web site quality (see for example[16]). Nonetheless these approaches have the same limitations that in other areas lead to the introduction of fuzzy techniques. In terms of methodology, contrary to what might be expected, the characterization of Web sites with the conventional approach of system analysis can result in greater subjectivity. In fact, forcing an association between metrics and intrinsically subjective elements could increase the level of arbitrariness. For example, a metric that is based on the number of broken links to evaluate the maintenance of a site means first deciding a) which links to analyze: all, only those internal links which the site owner has full control of, also external links that give information related to the core business, those related to the first hierarchical levels of the site, those that can be analyzed automatically (keeping in mind that automatic tools do not analyze pages of a site where the address is no longer linked to the home page); each of these possibilities could be reasonably defended, but the decision must take into account the type and size of the site, the domain, and the evaluation objectives; b) what is an acceptable number of broken links: the literature refers to a 3% threshold, yet even if this is applicable for each evaluation project it must still be integrated with information that considers the position and semantics of links (a non-functioning link on the home page or on the first pages creates greater problems, but even internal links can be critical if connected to important transactions or information). These problems can be exacerbated by features of the site such as graphics, where it is nearly impossible to conduct an objective evaluation; as an example, let’s suppose that we were confronted with the task of comparing two sites to determine which has the better graphics. Since there are no available means to measure in some detail the quality of graphics, we can introduce some indicators that are based on assumptions such as follows: “the page with more images has a higher value than one with fewer images,” or the opposite if it is the desired characteristic. In
the end this technique is essentially an attempt to impose qualitative standards on the artistic value of a creative work. It is necessary to adopt subjective criteria, also calling upon experts in the field for their evaluation. Moreover, even when the parameters for a specific metric have been set, the process of obtaining the necessary data to quantify them could be excessively costly in terms of time and financial outlay. Another problem with classifications and crisp thresholds is the loss of information, given that they are not able to incorporate imprecise or incomplete information, often qualitative in nature, into the evaluation; this leaves out information that could be highly valuable because of its link to specialized knowledge provided by experts in the domain. The field of sports provides a general example of how an evaluation expressed with a number can be even more subjective or arbitrary than one expressed in qualitative terms. There are sports that can be defined as “measured” – cycling, football, auto racing, athletics, etc. - where the results are expressed in a precise measurement, usually time or length. Other sports instead are “judged” – gymnastics, diving, skating, etc. (One can also argue that “measured” sports have elements of subjectivity: an example being the numerous post-match debates about whether players were off-sides or a referee call was correct, etc.). In judged sports the problem of evaluation of performance is handled by experts, each of whom gives a score (which in a strict sense does not express a measurement) and applies general criteria, judging style and the presence of mistakes, etc. Rules also attempt to provide some objectivity to judged sports (throwing out the highest and lowest scores, for example). This all serves to provide greater transparency in the judging process and to reduce the effect of inevitable opportunistic judgements (for example, favouring the team or athlete from the home nation, etc.). In short, given that we are talking about the design and systemic as well as systematic assessment of Web sites, it is more appropriate to refer to it as an evaluation rather than a measurement of quality and to apply techniques that take into account subjective aspects of the evaluation.

An analysis of the general elements contributing to the complexity of Web site evaluation and thus of the need to use techniques based on SC is incomplete without adding the fundamental point that quality comes in different forms [17]. Descriptions of at least ten different types of quality can be found in the literature, some linked to the quality of the product (in turn described as internal or external quality) or of the service (some authors further classify this: technique, relational, environmental, image, economic, organizational) or to its fruition: quality in use, perceived quality, expected quality, latent (unexpected) quality, requested quality; others refer to the management of quality within an organization: planned quality, quality of the resources, of the process, and quality offered or delivered. For each of these types of quality there are different corresponding metrics (see for example the quality metrics for products of ISO 9126 of 2001 [18]). For our purposes it is important to note how the different types of quality are interrelated, thereby producing gaps that create problems for a thorough and accurate evaluation as to whether an entity satisfies stated and implied needs; these gaps fall into four general categories:

1) understanding and identification of needs and requirements of customers, meaning a gap between expected quality (needs) and planned quality (requirements);
2) nonconformity, referring to the gap between planned and delivered quality;
3) communication, arising from the gap between quality actually delivered and quality perceived;
4) satisfaction, stemming from the gap between perceived and expected quality, thus providing an indication of customer satisfaction.

The fourth gap is a function of all the others and determines the success or otherwise of the product or service [19]. This is likely the reason why most Web site quality evaluation models focus on the user (see for example, [20]), besides the fact that most evaluations are based mainly on the client-side information (Figure 2).

### 3 Subjectivity in the evaluation of quality in Web sites

#### 3.1 A general methodology to evaluate Web site quality

In light of the previously discussed aspects characterizing Web site quality, in this section we introduce a meta-model and a process model to arrive at a general quality evaluation methodology. The goal is to obtain a conceptual framework that will then make it possible to decide – as in a feasibility study – what steps and activities in a quality evaluation can benefit from the use of SC techniques.

**A meta-model for the quality of Web sites** All models for Web site quality evaluation are based on a series of characteristics, ranging from just a few to several hundred [21]. Moreover, nearly all models have a hierarchical structure of two or three levels; some of the principal characteristics are specialized into sub-characteristics or attributes, the latter are then defined through their sub-attributes. Structures of this type are found, for example, for ISO software quality models [18].

To provide a general framework we will refer to a meta-model called 7Loci (Figure 1), which foresees seven dimensions that correspond to the loci used in the rhetoric of Cicero [14]. The 7Loci can be seen as a meta-model for classification of diverse criteria for quality, given that existing models can be obtained by “instantiating” it [22].

<table>
<thead>
<tr>
<th><strong>QVIS?</strong> (Who)</th>
<th><strong>QVID?</strong> (What)</th>
<th><strong>CVR?</strong> (Why)</th>
<th><strong>VBI?</strong> (Where)</th>
<th><strong>QVANDO?</strong> (When)</th>
<th><strong>QVOMODO?</strong> (How)</th>
<th><strong>QVIBVS AVXILIIS?</strong> (With what means and devices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>Content</td>
<td>Services</td>
<td>Location</td>
<td>Maintenance</td>
<td>Usability</td>
<td>Feasibility</td>
</tr>
</tbody>
</table>

Fig. 1. Ciceronian loci and dimensions of the 7Loci model (in Latin V stands for U)
The first dimension, Identity, regards the image that the organization projects and therefore all elements that come together in defining the identity of the owner of the site. Content and Services refer, respectively, to the information and services available for users. Location regards the visibility of a site; it also refers to the ability of the site to offer a space where users can communicate with each other and with the organisation. Maintenance comprises all activities that guarantee proper functioning and operability of the site. Usability determines how efficiently and effectively the site’s content and services are made available to the user. Feasibility includes all aspects related to project management.

**The process for evaluating quality in Web sites** A general model of the evaluation process envisages an initial setup phase in which requirements are established, a design phase in which the evaluation plan and techniques are defined, followed by the realization phase (Figure 2). The phases must be repeated in cycles using an iterative approach that characterizes the general vein of quality (see for example the “Plan-Do-Check-Act” cycle [23]). The main inputs in the evaluation process are (a) the purpose of the evaluation, (b) URL, mission, goals, type and domain of the site (or sites) to be evaluated, besides the site itself, at least the client side.

---

![Fig. 2. Evaluation process](image)

Regarding the first point, an evaluation can be born of very diverse needs: to extend services offered on an e-commerce site, to identify the reasons for unsuccessful marketing strategies, to compare the site with the competitors’ sites, etc. The mission of the site is a fundamental element of input given that it is impossible to evaluate a site if the “owner’s” goals for the site remain unclear. The main objective of the setup phase is to identify the stakeholders and particular attention must be placed on identifying the specific target group for the site. Then for all stakeholders identified it is necessary to define requirements, focusing on the objectives of the owner and of users of the site and on the user profiles (the set of characteristics common to the user group, such as nationality or city, age group, income level, interests and hobbies, etc.). The information in user profiles can be obtained in different ways through techniques much like those used for market research, using data obtained directly from the users and also data coming from traffic linked to the
site [24]. Obviously the requirements analysis for users becomes more difficult as the
user profiles for the site become more varied.

In the Set-up phase it is important to adopt a model of quality which according to
our approach would mean instantiating the 7Loci meta-model, which identifies the
quality attributes that relate to the dimensions considered. For some projects it is
possible to use a “standard” table. In some of our projects we have identified two or
three attributes for each dimension, and two sub-attributes for each attribute, for a
total of 26 characteristics to evaluate (see for example [14]). The specification of the
attributes and sub-attributes for each dimension of the model is the most delicate part
in the setup phase in that it determines the level of detail at which each dimension
must be analysed. Requirements that are classified can then be converted into
questions in order to obtain semantic models that take into account aspects related to
the domain and to the type of site. In doing so a generic question from a standard
table for the Content dimension (for example, “Is there enough of the necessary
information for the purposes of the site?”) for the site of a tourist destination can be
articulated as several questions having to do with the requirements for this dimension:
“Is there information on hotels? On the non-hotel accommodations sector? On
restaurants? On locally made products?” etc. It is evident that detailed models are
more precise and less subjective but also more costly. Indeed their design and creation
require abundant resources and they also require frequent updating given that they
become outdated as the profiles of users change, and also the aims of the owners, in
addition to technological changes.

The principal objective of the Evaluation Design phase is to identify the
appropriate assessment modalities for the attributes of the quality model in agreement
with the quality requirements defined in the setup phase. It is necessary at this point to
determine the survey modalities, which can vary depending on the techniques and
tools adopted as well as on the number and role (competency) of the evaluators. As
for the techniques, according to a classification proposed in the literature of the HCI
(Human Computer Interface), we can distinguish empirical and analytical techniques.
The decision to use one method rather than another depends on factors such as the
aims of the Web site, the data requested, etc., not to mention the time and resources
available. Nonetheless, diverse techniques should be employed, and to further
integrate the results, the evaluation should, for example, defer to experts for some of
the attributes and to on-line surveys for others. In some cases it may be useful to
analyze the log files of the Web site. This can show – for example - which pages are
the most popular in order to know where to place the priority for any eventual
modifications or for translation into other languages. Many evaluation techniques
involve choices resulting in the trade-off between a quantitative and a qualitative
evaluation, decisions where SC can make a decisive contribution. Other general
decisions in this phase can have to do with which language version of a site to
evaluate (if the site is available in more than one language), which sections of the site
to evaluate, the order in which to analyze the dimensions, etc.

The Realization phase moves on to the evaluation of the site, applying the
techniques of survey and the measurement modalities specified in the evaluation plan.
The main activities in this phase are: data gathering, analysis and interpretation of
results, and the compilation of reports. Data gathering is usually done by visiting the
site at least once but in some cases it may be necessary to access files or information
only available on the server side. Evaluation of some attributes can be supported by software tools, for example to validate the use of HTML (http://www.validator.w3.org). The results obtained must be compared — using appropriate methods — with the quality requirements specified in the setup phase. This comparison is not always straightforward, especially for qualitative evaluations. Particularly important for data analysis and the creation of a final report is the need for a global “score” for the site as a whole (used for example in comparisons with other sites or as a measurement of improvements stemming from the re-designing of a site) or to calculate “average values,” accounting for the “weight” of different attributes where the elements analyzed are not only qualitative but also numerous and different from each other.

### 3.2 Soft computing in Web site quality design and evaluation

In this section we discuss activities and issues regarding the process described in Figure 2 that could be supported by techniques of SC. We do not describe actual applications of these techniques, but we seek to provide a general schema of how an SC expert could contribute to improving the efficacy of evaluation projects. To this end it is important to underline again that the guiding principle of SC is to exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. [5] The aim of SC is to have tools that can deal with complex problems where detailed specification is virtually impossible and, therefore, conventional problem solving is unlikely to produce useful solutions. In other words, SC can be useful in providing systematic treatment to qualitative and subjective information and in particular when this information requires the application of computational models to classify, filter, make forecasts, optimize, plan, decide, and consider contradictory findings, etc.

Focusing on the activities necessary for an evaluation process, we find the following possible applications of SC:

**Set-up phase:**

- To classify the goals and objectives of stakeholders, which if numerous can be in conflict with each other.
- To identify the different profiles of users through clustering techniques that make it possible to use information provided directly by users as well as information garnered from an analysis of site traffic.
- To classify requirements in natural language, which as such are ambiguous, often incomplete and contradictory.
- To instantiate the quality model, calibrating the number of attributes on the basis of the evaluation objectives and of models available in a repository that contains information from prior evaluations.
- To assign weights to the different dimensions of the site considered in the quality model and articulated in the different attributes identified (permitting
experts or the owners to use adjectives that can then be translated with linguistic modifiers).

- Both attributes and weights can be gathered afterward from the evaluation results using data-driven techniques.

**Design phase:**

- To introduce qualitative “metrics” for those aspects that are intrinsically subjective, such as whether the graphic design is adequate for purposes of marketing.
- To make it possible to give qualitative scores based on scales of preference or expressed with linguistic tags.

**Realization phase:**

- To manipulate “linguistic” scores such as those assigned by experts or users for standard tables.
- To compare qualitative assessments for different sites or for repeated analysis of the same site.
- To evaluate different types of quality, and therefore to evaluate the gaps indicated in Figure 2, which require various types of data that come from different sources.
- To check for the presence of patterns in the results of evaluations of sites from different domains or from different categories of owners.
- To identify the points where SC can optimize further interventions on the site.

4 Conclusions

Einstein famously described a true paradox when he said: “Not everything that counts can be counted, and not everything that can be counted counts.” Soft Computing addresses this trade-off. Few approaches or evaluation projects can be found in the literature on Web site quality that apply techniques of SC, while there are applications for product quality in industry, and - closer to web site - to evaluate information quality [25]. On the other hand, similar considerations applied in the recent past for evaluations of the quality of software, which has existed for much longer. The results of the methodological analysis described above are in some ways surprising. In fact, in light of the state of the art of Web site quality evaluation, we might have expected that SC techniques would be useful mostly in “measuring” the attributes of Web sites, that is in the design and realization phases. The findings revealed, however, that SC can be used in all phases of the evaluation process - confirming the intrinsically qualitative and subjective nature of the process. The list obtained with this preliminary study represents a first step toward a collaboration with SC experts: they could suggest the Web Engineer the best technique for a given task. Experiences in SC date back about 30 years, with a wide application in many different fields, so we could trust it; that is we do not have to demonstrate that SC could be useful to Web site quality evaluation, but to apply it to improve our evaluation projects.
References

Testing Techniques applied to AJAX Web Applications

Alessandro Marchetto¹, Paolo Tonella¹, and Filippo Ricca²

¹ Fondazione Bruno Kessler - IRST, 38050 Povo, Trento, Italy
marchetto@itc.it
² Unità CINI at DISI, 16146 Genova, Italy
filippo.ricca@disi.unige.it

Abstract. New technologies for the development of Web applications, such as AJAX, support advanced, asynchronous interactions with the server, going beyond the submit/wait-for-response paradigm. AJAX improves the responsiveness and usability of a Web application but poses new challenges to the scientific community: one of them is testing. In this work, we try to apply existing Web testing techniques (e.g., model based testing, code coverage testing, session based testing, etc.) to a small AJAX-based Web application with the purpose of understanding their real effectiveness. In particular, we try to answer the following questions: “Is it possible to apply existing testing techniques to AJAX-based Web applications?”; “Are they adequate to test AJAX applications?”; and, “What are the problems and limitations they have with AJAX testing?”. Our preliminary analysis suggests that these techniques, especially those based on white-box approaches, need to be changed or improved to be effectively used with AJAX-based Web applications.

Keywords: AJAX-based Web applications, Web testing techniques.

1 Introduction

Traditional Web applications are based on the client-server model: a browser (client) sends a request asking for a Web page over a network (Internet, via the protocol HTTP) to a Web server (server), which returns the requested page as response. During this elaboration time the client must wait for the server response before visualizing the requested page on the browser, i.e., the model is based on synchronous communications between client and server.

During the last few years, several Web applications (such as Google Suggest, Yahoo Instant Search, Google and Yahoo Mail, etc.) have been developed using a new technology named AJAX [11, 4]. AJAX breaks the traditional Web page paradigm, in which a Web application can be thought of and modeled as a graph of Web pages, and invalidates the model of traditional Web applications. It introduces additional asynchronous server communication to support a more
responsive user interface: the user interacts directly with items within the page and the feedback can be immediate and independent of the server’s response.

The validation of Web applications that are based on asynchronous communication with the server and employ new technologies, such as AJAX, Flash, ActiveX plug-in components, is an area that deserves further investigation. For example, until now, little research efforts have focused on how to test Web applications employing AJAX. Since AJAX does not comply with the classical Web application model, several techniques presented in literature will not work any longer. This seems to be the case of model based Web testing: using a Web Crawler\(^3\) to extract the model seems not possible anymore. Before devising new techniques specific of AJAX, we think it is important to address the following questions: “Which testing techniques are able to work with AJAX-based Web applications?” and, “To what degree?”

In this paper, we apply some existing testing techniques to a simple Web application based on the AJAX technology. Our purpose is to detect effectiveness and advantages as well as limitations and problems, of each examined testing technique.

The rest of the paper is organized as follows. We start, in Section 2, explaining the main characteristics of AJAX. Section 3 presents the existing testing techniques and tools examined. In Section 4 we analyze each technique in order to understand and evaluate its applicability to AJAX-applications. In the same Section we apply the analyzed techniques to a small AJAX-application. Section 5 summarizes our preliminary results and, finally, in Section 6 we conclude the paper.

\section{AJAX}

AJAX (Asynchronous Javascript And XML) is a bundle of existing technologies used to simplify the implementation of rich and dynamic Web applications. HTML and CSS are used to present the information, the Document Object Model is used to dynamically display and interact with the information and the page structure, the \texttt{XMLHttpRequest} object is exploited to retrieve data from the Web server, XML is used to wrap data and Javascript is exploited to bind “everything together” and to manage the whole process. With AJAX developers can implement asynchronous communications between client and server. To achieve this, client-side scripts and a special AJAX component named \texttt{XMLHttpRequest} are used. Thanks to AJAX, Web developers can update parts of the client-side page independently: in AJAX the units of work are the page elements (e.g., text area, HTML form, DOM structure) rather than the whole page, as happening with traditional page-based Web applications. For this reason, AJAX breaks the traditional Web page paradigm. Every element of an HTML page may be tied to some AJAX \textit{action}; every action may generate a server request, associated with

\(^3\) A \texttt{Web crawler} (also known as a \texttt{Web spider} or \texttt{robot}) is a program that automatically traverses the Web’s hyperlink structure and retrieves some information for the user.
a URL, so that many HTTP requests can be performed by a single client-side page.

The main technological novelty of AJAX is the XMLHttpRequest object used to exchange request and data, wrapped into XML packages, between Web components. In particular, through this object a client-side component (e.g., HTML page) can send HTTP requests to the Web server and capture its response in order to update pieces of the same HTML page. XMLHttpRequest allows to send asynchronous GET/POST HTTP requests to a given Web server without showing any visible effect to the user and, more importantly, without stopping the component execution, since a special AJAX engine controls HTTP requests and responses in background, using an event listener. In other terms, it allows Web developers to specify event handlers that change the client-side component state whenever a server response is received asynchronously.

3 Web Testing Techniques

Existing approaches for Web Application testing can be divided into three classes: white-box, black-box and session-based testing.

3.1 White-Box testing

Similarly to traditional software, white-box testing of Web applications is based on the knowledge about the internal structure of the system under test. This approach can be applied to Web applications either by representing the structure at the high-level, by means of the navigation model (Model-based testing), or at the low-level, by means of the control flow model (Code-coverage testing). In the white-box category we consider also Mutation-based testing, which requires internal knowledge of the Web application under test.

1. Model-based testing [1, 3, 6]. In this approach, reverse engineering techniques and a Web crawler are used to build the Web model of the target application. The built model is a graph where each node represents a Web page and each edge represents a link (e.g., HTML links, submits, automatic redirections). Test cases are extracted by traversing the navigational system (i.e., the Web model) of the application under test. A test case is composed of a sequence of pages plus input values.

2. Code-coverage testing [10]. This approach is based on knowledge of the source code of the Web application under test [10]. In code coverage testing, the level of coverage reached by a given test suite can be determined by instrumenting (through trace instructions) the branches in the control flow model. Since the execution on the server involves one (or more) server side languages (e.g., Java, PHP, SQL) and the execution on the client involves additional languages (such as HTML and JavaScript), the control flow model
has different node kinds. Examples of tools used to perform code-coverage testing for Java software are Emma\(^4\) and Clover\(^5\).

3. **Mutation-based testing** [2, 8]. Code mutation has been used in the literature for several purposes. In the context of Web applications, it can be used to recover the Web model [2]. In this approach, code mutation is applied to the server-side code in order to automate the difficult task of model construction. More traditional uses of mutation consist of applying mutation operators to the source code in order to generate code mutants. Mutants are exercised through suites of test cases in order to evaluate their effectiveness in finding faults. For instance, Elbaum et al.[8] use a fault-based approach to evaluate the effectiveness of test suites constructed by means of a session-based approach. Finally, another use of this defect-injection method is the selection a subset of test cases, based on a fault-coverage criterion.

### 3.2 Black-Box testing

The test of the functional requirements can be conducted by considering the Web application as a black-box. Web applications may have documents describing the requirements at the user level, such as: use-cases, user stories, functional requirements in natural language, etc. From these documents, it is possible to create a list of test cases. Each test case, i.e., ordered list of Web pages plus user inputs, describes a scenario that can be accomplished by a Web visitor through a browser. Output pages obtained by navigating the application and providing the requested inputs are compared with the results expected from that interaction according to the requirements. In this work, we consider two black-box approaches: capture/reply and xUnit.

1. **Capture and Reply**. The most common class of black-box testing tools provide an infrastructure to support the capture and replay of particular user scenarios. During black-box testing of Web applications the interaction with the user can be simulated by generating the graphical events that trigger the computation associated with the application interface. One of the main methods used to obtain this result consists of recording the interactions that a user has with the Web application and repeating them during the (regression) testing phase. A lot of functional and regression testing tools, based on capture/replay facilities, are available as free and commercial software. Examples are Mercury WinRunner \(^6\), IBM Rational Robot \(^7\) and MaxQ \(^8\).

2. **xUnit testing**. Another approach to black-box testing is based on HttpUnit\(^9\). When combined with a framework such as Junit\(^10\), HttpUnit permits pro-

---

\(^4\) http://emma.sourceforge.net  
\(^5\) http://www.cenqua.com/clover  
\(^6\) http://www.merc-int.com  
\(^7\) http://www-306.ibm.com/software/awdtools/tester/robot  
\(^8\) http://www.bitmechanic.com  
\(^9\) http://httpunit.sourceforge.net  
\(^10\) Http://www.junit.org
grammers to write Java test cases that check the functioning of a Web application. HttpUnit is a Java framework well suited for black-box and regression testing of Web applications, which allows the implementation of automated test scripts based on assertions. In the xUnit family are also other tools, such as JsUnit\textsuperscript{11} and PHPUnit\textsuperscript{12}. Several xUnit tools (e.g., Selenium\textsuperscript{13}) use the capture and reply mechanism to record scripts of test. These scripts can be completed and enriched with assertions by the user.

### 3.3 Session-based testing

Another approach to testing Web applications is user-session based testing. It relies on capturing and replaying real user sessions. This approach avoids the challenge of building an accurate model of a Web application’s structure. Each user session is a collection of user requests in the form of URL and name-value pairs (i.e., input field names and values). A user session begins when a user makes a new request to a Web application and ends when the user leaves it or the session times out. To transform a user session into a test case, each logged request is changed into an HTTP message that can be sent to the Web server. A test case consists of a set of HTTP requests that are associated with each user session. Different strategies can be applied to construct test cases from the collected user sessions stored in a log file \cite{5}. The simplest strategy is transforming each individual user session into a test case. Other, more advanced, strategies are also possible \cite{7}. An example of access log is given in Figure 5.

### 4 Testing of AJAX-based Web Applications

In this section we analyze white-box, black-box and session-based testing in terms of their applicability to AJAX-applications, evaluating limitations and problems, considering existing tools and applying them to a small “case study”.

---

\textsuperscript{11} http://www.jsunit.net
\textsuperscript{12} http://www.phpunit.de
\textsuperscript{13} http://www.openqa.org/selenium

---

**Fig. 1.** customerID UML model
For our study we selected a small application named *customerID*\(^{14}\). Although small, it represents a typical AJAX application, which can be analyzed, tested and described in detail. Figure 1 shows the UML class diagram of this application. The *customerID* application is composed of:

- a client-side HTML page (*client.html*);
- an HTML form (*customer*) used by *client.html*;
- a Javascript code fragment (*JS*) used by *client.html*;
- a server-side page *Customer.jsp*.

*JS* retrieves the customer identification number written by the user (see Figure 2) in the HTML form and uses an *XMLHttpRequest* object (*activated* by the user through the “submit” button of the form) to send this number to the server-side page *Customer.jsp*. *Customer.jsp* is a JSP component that receives a customer id from the client and returns first and last name of the customer associated with the received id, using a XML-based object for data transmission. Then, the client-side component *JS* captures this XML package and uses the contained data to update the client page. In that operation, a fragment of the Javascript code *JS* is used to get the data and to update the state of the HTML page *client.html*.

4.1 Model-based testing

Model-based testing are only partially usable to test AJAX applications, since the associated model is not adequate for this kind of applications. Existing

\(^{14}\) http://www.ics.uci.edu/~cs122b/projects/project5/AJAX-JSPExample.html
model-based techniques describe a Web application through a navigation model composed of Web pages and hyperlinks. The problem is that AJAX applications are essentially composed of a single page that changes its state according to the user interactions. Other problems with the model-based approach are: First, existing Web crawlers, used to build the Web model, are not able to extract the set of dynamic links created by AJAX applications. Second, client and server components tend to exchange small XML data fragments instead of entire HTML pages. This peculiarity of AJAX makes it impossible to capture the server response in order to extract the hyperlinks of the navigation model, as done in conventional model-based testing. Third, existing Web models don’t capture the evolution of a Web page in terms of successive states (i.e. DOM configurations) and they are not able to represent the XML data exchanged among components. Summarizing, to perform model-based testing of AJAX-based applications one needs to:

1. improve the “capability” of the actual Web crawlers;
2. extend the Web model.

These limitations are evident when we apply model-based testing to \textit{customerID}. The model produced by the Web crawler is partial (Figure 3 shows the model extracted according to Ricca and Tonella [6]). It is composed only of the page \textit{client.html}, its HTML form and the page \textit{customer.jsp}. The result of executing \textit{customer.jsp} on the server is missing in the model because XML packages cannot be represented in this model.

4.2 Mutation-based testing

Two studies[2, 9] apply mutation-based testing to Web applications. Bellettini et al. [2] use mutation to recover the Web model, while Sprenkle et al. [9] use mutation to evaluate the effectiveness of a test suite by inserting artificial defects (mutations) into the original application. No work to the best of the authors’ knowledge tried to apply mutation to the client-side code of Web applications. The definition of mutation operators for AJAX could be quite complicated, due to the possibility of run-time changes of the DOM structure and page content. We think that mutation-based testing may be useful to support the testing approaches for AJAX applications. This involves studying and defining AJAX-specific mutation operators.

We tried to apply the technique proposed by Bellettini et al. [2] to \textit{customerID}, but we found a problem: This testing technique cannot be directly used with \textit{customerID} because it applies mutation operators only to server-side pages and because the response of the server is an XML package and not an entire page, as expected by the technique.

4.3 Code Coverage testing

In theory, it is possible to apply code coverage testing to AJAX-based applications. In practice there are some problems. The first is a technological problem.
Fig. 4. Output of our code-coverage tool applied to client.html
Coverage tools for Web applications need to trace Web code that, often, is a mix of languages such as HTML, Javascript, JSP and currently, tools with these characteristics are not available. Moreover, currently, it is impossible to use existing coverage tools to trace dynamic changes of the DOM structure or dynamically generated code. We think that this list of problems limits in practice the effectiveness of the code coverage testing approach.

Some tools such as Cobertura\textsuperscript{15}, JCover\textsuperscript{16}, Clover\textsuperscript{17} and Emma\textsuperscript{18} cannot be used to test complex and real Web applications because they can trace and analyze only Java (i.e., server-side) code. An example of code coverage tool for Javascript is the commercial tool Coverage Validator\textsuperscript{19}. It displays statistics for each Javascript file that is being monitored for code coverage.

A coverage tool able to work with a mix of Web languages is under development at our laboratory. Figure 4 shows the output of our tool applied to the \textit{client.html} page of the \textit{customerID}. The code (HTML and Javascript) exercised during test case execution is traced and shown in light gray. The tool reports (see top of the figure) some code-coverage metrics such as condition (60\%) and statement coverage (80\%).

When applying our tool to \textit{customerID}, some problems become evident. It is difficult to trace dynamic code and dynamic changes of Web pages (e.g., DOM changes). For instance, we cannot trace the run-time changes of the HTML form embedded in the \textit{client.html} page. The reason is that the AJAX component, used by the same page, dynamically updates the form (\textit{name} and \textit{lastname} fields in Figure 4) through asynchronous communications with the server page. Hence dynamic changes of the form field values (lines 38-39 on Figure 4) cannot be traced by the coverage tool.

127.0.0.1 - - [04/Dec/2006:11:32:03 +0100] “GET /ajax1/ HTTP/1.1” 200 1563

\textbf{Fig. 5.} Tomcat log-file of \textit{customerID}

4.4 Session-based testing

This testing approach is apparently applicable to AJAX applications, because in a log-file we can capture both kinds of HTTP requests: traditional (triggered by

\textsuperscript{15} http://cobertura.sourceforge.net
\textsuperscript{16} http://www.mmsindia.com/JCover.html
\textsuperscript{17} http://www.cenqua.com/clover
\textsuperscript{18} http://emma.sourceforge.net
\textsuperscript{19} http://www.softwareverify.com/javascript/coverage/feature.html
user clicks), as well as AJAX-specific (triggered by AJAX-components). However, session-based testing techniques are adequate to verify only non-interacting, synchronous request-response pairs, because of two kinds of limitations:

1. Using (only) log-files information it is not possible to reconstruct the state of Web pages that are modified during the execution of a given application. The reason is that the data exchanged between client and server in a given AJAX application are only “pieces of data” wrapped into XML messages, that are eventually turned into page changes;
2. Some techniques used to mix log file information (i.e., sequence of hyperlinks and input values) cannot be used to generate new navigation sessions (i.e., new sequences of links and input values) useful to exercise the application under test. In fact, it may be hard to reproduce the context where the log information used in the original application can be re-inserted in a different scenario.

Figure 5 shows a fragment of the Tomcat log-file for the customerID application. Given this log file, we can reply the two customerID usage sessions performed by the user but we cannot take advantage of the information in the XMLHTTPRequest object used by customerID. For this reason, we can not repeat its behavior. Moreover, we can use the log information to verify each single response to the HTTP requests stored in Tomcat log files. But, we cannot use it to test the entire application customerID, since it is impossible to reconstruct the state of the application when each HTTP request was issued and when the response was received. Furthermore, by mixing the log-extracted HTTP requests we might end up with inconsistent requests, since knowledge of the AJAX-objects used by customerID cannot be derived from log-files only. Some examples of test cases derived from the log-file in Figure 5 are the following:

1. GET client.html → verify the HTML code sent by the server on response;
2. GET client.html → send ID=134 to customer.jsp using a GET request → verify the returned XML package;
3. GET client.html → verify the HTML code sent by the server on response;
4. GET client.html → send ID=134 to customer.jsp using a GET request → send ID=234 to customer.jsp using a GET request → verify the returned XML package.

4.5 Capture and Reply

Capture and replay is in principle applicable to AJAX-based Web applications, since it exercises the application from a user point-of-view using the GUI. However, the real applicability to AJAX-based Web applications depends on the actual testing tool in use. Several implementations of this technique would need to be improved to be effectively used to test AJAX applications. To be successfully applied to AJAX applications, a capture and reply tool should:

1. support Javascript;
function testupdateFirstLastName() {
<?xml version='1.0' encoding='UTF-8'?>
<LogiTest:test
xmlns:LogiTest='http://www.logitest.org'>
<LogiTest:name>Untitled</LogiTest:name>
<LogiTest:description /></LogiTest:test>
</LogiTest:test>

Fig. 6. LogiTest applied to customerID

2. be able to capture dynamic events associated with user input;
3. be able to capture dynamic changes of the DOM structures;
4. be able to perform asynchronous HTTP requests to the Web server.

Examples of capture and reply tools are LogiTest\textsuperscript{20}, Maxq\textsuperscript{21} and Badboy\textsuperscript{22}. These tools don’t work well with AJAX applications. Logitest doesn’t support Javascript while Maxq cannot record dynamic events. Badboy supports Javascript but it is not able to capture some dynamic events (e.g., “onblur” on form input fields) and run-time changes of the DOM structures. Thus, it is not adequate to test AJAX applications. The commercial tool eValid\textsuperscript{23} and the tool Origsoft\textsuperscript{24} promise to test the AJAX applications. Indeed, they are able to capture dynamic events and store dynamic DOM changes.

\textsuperscript{20} http://logitest.sourceforge.net
\textsuperscript{21} http://maxq.tigris.org
\textsuperscript{22} http://www.badboy.com.au
\textsuperscript{23} http://www.soft.com
\textsuperscript{24} http://www.origsoft.com
We have tried to apply a tool of this category to customerID. Figure 6 shows the navigation-script stored by LogiTest for an usage session of our customerID. It is clear that this script is not adequate to test customerID since it doesn’t contain information related to the asynchronous HTTP requests performed by the application.

4.6 xUnit testing

The xUnit approach is also in principle applicable to test AJAX applications. Actually, it focuses on the functional behavior rather than the implementation. However, the real applicability to AJAX Web applications depends on the actual implementation of the tool in use. To be used with AJAX-applications, xUnit tools must support, at least, Javascript, asynchronous HTTP requests and DOM inspection.

Examples of xUnit testing tools are: Latka, HTTPUnit, InforMatrix, HTMLUnit, JsUnit, Canoo WebTest, squishWeb and Selenium. Latka\(^{25}\) and HTTPUnit\(^{26}\) cannot be used to test AJAX-applications because they are not able to manage asynchronous HTTP requests and DOM inspection. Some tools such as InforMatrix\(^{27}\), HTMLUnit\(^{28}\) and Canoo WebTest\(^{29}\) have been recently improved to support Javascript and AJAX components. Unfortunately, their Javascript support is still limited and in real AJAX applications they can hardly be applied. Instead, software such as squishWeb\(^{30}\) and Selenium\(^{31}\) can be used to test AJAX applications, since they fully support Javascript, asynchronous HTTP requests and DOM inspection.

We have tried to apply some tools of this category to customerID. InforMatrix and HtmlUnit can not be used to test customerID: they don’t support some DOM-events actions, while they partially support Javascript. Differently, Selenium, thanks to the assertion “wait conditions”, can be successfully used to test customerID. Figure 7 shows two screenshots of Selenium. The considered test case is the following:

1. load client.html → type the number “123” in the form ID → click the submit button (i.e., send data to the customer.jsp) → verify the form name has been updated with “John”.

In this example, we use two different assertions to verify the output of the above test case. In the first case, we use a conventional “assert” command instead of the correct “waitForValue” (see Figure 7, right). Without this “waitForValue”

\(^{25}\) http://jakarta.apache.org/commons/latka
\(^{26}\) httpunit.sourceforge.net
\(^{27}\) http://www.informatrix.ch
\(^{28}\) http://htmlunit.sourceforge.net
\(^{29}\) http://webtest.canoo.com
\(^{30}\) http://www.froglogic.com
\(^{31}\) http://www.openqa.org/selenium-ide/
the test case verification fails (see Figure 7, left) since the AJAX application customerID uses asynchronous communications and thereby a conventional “assert” cannot be used to capture the server response.

5 Discussion

<table>
<thead>
<tr>
<th>Testing</th>
<th>adequate problems</th>
<th>tools</th>
<th>customerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-based</td>
<td>no Web models extracted are partial; existing Web crawlers are not able to download site pages</td>
<td>research</td>
<td>not ok</td>
</tr>
<tr>
<td>Mutation-based</td>
<td>no mutant operators are never being applied to client Web code; the application of mutant operators is difficult</td>
<td>not-existing</td>
<td>not ok</td>
</tr>
<tr>
<td>Code Coverage</td>
<td>partially it is difficult to cover dynamic events and DOM changes; coverage tools managing a mix of languages are not available</td>
<td>Javascript: Coverage validator, Java: Cobertura, Emma, Clover, etc. Languages mix: not available</td>
<td>partially ok</td>
</tr>
<tr>
<td>Session-based</td>
<td>no it is impossible to reconstruct the state of the Web pages using only log-files</td>
<td>research</td>
<td>not ok</td>
</tr>
<tr>
<td>Capture&amp;Reply and xUnit</td>
<td>yes Javascript, asynchronous HTTP requests and DOM analysis are not always supported</td>
<td>not ok: Maxq, HTTPUnit, InforMatrix etc. partially ok: Badboy, HTMLUnit etc. ok: squishWeb, Selenium etc.</td>
<td>it depends on the tool implementation</td>
</tr>
</tbody>
</table>

Table 1. Web testing techniques applied to AJAX-based applications

Table 1 summarizes our preliminary analysis of existing testing techniques applied to AJAX Web applications:

1. **model-based**: test cases are derived from the Web model. The model doesn’t consider all the states that a single HTML page may reach during the execution of the Web application. So, this technique appears applicable but not adequate to test AJAX-applications.

2. **mutation-based**: to our knowledge, no work defines mutation operators that apply to client-side code of Web applications. Moreover the application of mutation operators is complicated by the specific nature of the AJAX Web applications (HTML pages can dynamically change their DOM structure and content). Additionally, specific tools for Web applications are not available. So, this technique is promising, but far from being available in practice.
3. **coverage-based**: this technique is applicable and adequate to test AJAX applications. However, its real effectiveness has to be verified in practice since it is difficult to instrument and trace AJAX dynamic code (in particular, dynamic events and DOM changes). Another problem is that coverage tools for Web applications must trace a mix of languages and currently, tools with these characteristics are not available. For this reason, this technique is considered only partially adequate to test this kind of software.

4. **session-based**: in this technique test cases are derived using the data recorded in the log-files. Since in a log-file we can capture traditional HTTP requests as well as AJAX-specific ones, this technique is apparently applicable to test AJAX applications. However, session-based testing is not fully adequate because it is hard to reconstruct the state of the Web pages that are exercised during the execution of a given application using only log-files information.

5. **capture/reply and xUnit**: this category of tools is adequate to test AJAX applications because it does not consider the internal structure of the target application. The capture/reply and xUnit tools verify the functionalities of the Web applications based only on requirements and expected output (black-box testing). However, often, existing tools don’t support AJAX or the support is still limited. A tool of this category, to be used with AJAX-based Web applications should support at least: Javascript, asynchronous HTTP requests and DOM inspection.

### 6 Conclusions

In this paper we have applied some existing testing techniques to a simple AJAX Web application with the purpose of detecting effectiveness and advantages, as well as limitations and problems, of each examined technique.

The results of our preliminary analysis can be summarized as follows: model and session-based testing techniques need to be modified and improved to let them test AJAX Web applications. Mutation-based testing needs to be adapted to be used with the client-side components of AJAX applications. Code-coverage testing can be only “partially” used with AJAX-applications: (1) the dynamism of this kind of technology limits its effectiveness; (2) tools are not available to manage mix of languages as required. Currently, capture/reply and xUnit testing are the only tools able to work with AJAX. However, some of them have to be extended/improved to support Javascript, dynamic changes/events of the DOM and AJAX-components.

Our preliminary analysis suggests that to test AJAX-applications new approaches/tools are needed, since the existing ones have severe limitations and problems, especially white-box and session-based testing techniques.
References

Automated Verification of XACML Policies Using a SAT Solver

Graham Hughes and Tevfik Bultan

Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
{graham, bultan}@cs.ucsb.edu

Abstract. Web-based software systems are increasingly used for accessing and manipulating sensitive information. Managing access control policies in such systems can be challenging and error-prone, especially when multiple access policies are combined to form new policies, possibly introducing unintended consequences. In this paper, we present a framework for automated verification of access control policies written in XACML. We introduce a formal model for XACML policies which partitions the input domain to four classes: permit, deny, error, and not-applicable. We present several ordering relations for access control policies which can be used to specify the properties of the policies and the relationships among them. We then show how to automatically check these ordering relations using a SAT solver. Our automated verification tool translates verification queries about XACML policies to a Boolean satisfiability problem. Our experimental results demonstrate that automated verification of XACML policies is feasible using our approach.

1 Introduction

Web-based applications today are used to access all types of sensitive information such as bank accounts, employee records and even health records. Given the ease of access provided by the Web, it is crucial to provide access control mechanisms for Web-based applications that deal with sensitive information. Moreover, due to increasing use of service oriented architectures, it is necessary to develop techniques for keeping the access control policies consistent across heterogeneous systems and applications spanning multiple organizations.

XACML (eXtensible Access Control Markup Language) [12] provides a common language for combining, maintaining and exchanging access control policies. XACML is an XML-based language for expressing access rights to arbitrary objects that are identified in XML. XACML provides rule and policy combining mechanisms for constructing policies from rules and metapolicies from policies, respectively.

Policies built using such mechanisms will inevitably become quite large and complex as they are used to combine access control rules and subpolicies in an organization and especially across organizations. It is possible, even likely, that the act of creating a metapolicy out of numerous disparate smaller policies could leave it vulnerable to unintended consequences. In this paper, we investigate statically verifying properties of access control policies to prevent such errors.

⋆ This work is supported by NSF grants CCF-0341365 and CCF-0614002.
We translate XACML policies into a mathematical model, which we reduce to a normal form by separating the conditions that give rise to access permitted, access denied, and internal error results. We define partial orderings between access control policies, with the intention of checking whether a policy is over- or under-constrained with respect to another one. We show that these ordering relations can be translated to Boolean formulas which are satisfiable if and only if the corresponding relation is violated. We use a SAT solver to check satisfiability of these Boolean logic formulas. Using our translator and a SAT solver we can check if a combination of XACML policies does or does not faithfully reproduce the properties of its subpolicies, and thus discover unintended consequences before they appear in practice.

In Section 2, after giving an overview of XACML, we develop a formal model for access control policies written in XACML and discuss how to transform these models into a normal form that distinguishes access permitted, access denied, and error conditions. In Section 3 we define partial ordering relations among access control policies which are used to specify their properties. We show how to check these properties automatically in Section 4. Finally, we report the results of our experiments in Section 5 and give our conclusions in Section 7.

2 Policy Specifications

An access request is a specially formatted XML document that defines a set of data that we call the environment. Given an environment, an XACML policy specification yields one of four results: Permit (Per), meaning that the access request is permitted; Deny (Den), meaning that the access request will not be permitted; Not Applicable (NoA), meaning that this particular policy says nothing about the request; and Indeterminate (Ind), which means that something unexpected came up and the policy has failed. XACML additionally defines obligations, which are actions that the policy must perform in some circumstances; we do not handle obligations in this work.

In XACML three classes of objects are used to specify access control policies: 1) individual rules, 2) collections of rules called policies, and 3) collections of policies called policy sets. XACML rules are the most basic object and have a goal effect—either Permit or Deny—a domain of applicability, and conditions under which they can yield Indeterminate and fail. The domain of applicability is realized in a series of predicates about the environmental data that must all be satisfied for the rule to yield its goal effect; the error conditions are embedded in the domain predicates, but can be separated out into a set of predicates all their own. Policies combine individual rules and also have a domain of applicability; policy sets combine individual policies with a domain of applicability.

XACML predicates can be constructed using primitive functions such as equality, set inclusion, and ordering within numeric types, and also more complex functions such as XPath matching and X500 name matching.

Let us consider a simple example policy for an online voting system. The policy states that to be able to vote a person must be at least 18 years old and a person who has voted already cannot vote. Our environment (i.e., the set of information we are interested in) consists of the age of the person in question and whether they have voted already. We can represent this as a Cartesian product of XML Schema [13] basic types,
as follows:

\[ E = \mathcal{P}(\text{xsd:int}) \times \mathcal{P}(\text{xsd:boolean}) \times \mathcal{P}(\text{xsd:string}) \]

Here, \( E \) denotes the set of all possible environments. The first component of an environment is the age of the person, the second component is whether or not they have voted already, and the third component is the action they are attempting (perhaps voting, but perhaps something else). We use power sets here because in XACML all attributes describe sets of values, never singletons.

The XACML policy for this example is shown in Figure 1. We will explain the semantics of this policy using a simple mathematical notation. We write all environment sets in the form \( \{ e \in E : C \} \) where \( C \) is a predicate whose only free variables are the components of the environment tuple \( e \). Since we do not at this time constrain the predicate \( C \), this does not cost us any generality.

The goal for our example policy is that if a person is doing something other than voting, we do not really care what happens, and we require that there be only one age and one voting record presented. To do this we can divide \( E \) into four sets, \( E_a, E_v, E_p \) and \( E_d \) as follows (note that the notation \( \exists! x \) \( P \) asserts that there is a unique \( x \) that
satisfies a condition \( P \):
\[
E_a = \{ (a, v, o) \in E : \exists ! a_0 \in a \land \exists ! v_0 \in v \} ,
E_v = \{ (a, v, o) \in E_a : \exists x \in o \; x = \text{vote} \} ,
E_p = \{ (\{a_0\}, \{v_0\}, o) \in E_a : a_0 \geq 18 \land \neg v_0 \} ,
E_d = E_v - E_p = \{ (\{a_0\}, \{v_0\}, o) \in E_a : a_0 < 18 \lor v_0 \} .
\]

Here, \( E_a \) is the set of all environments whose inputs are not erroneous, \( E_v \) is the set of all environments where voting is attempted, \( E_p \) is the set of all environments where the person can vote (their attempt to vote is permitted), and \( E_d \) is the set of all environments where the person cannot vote (their attempt to vote is denied).

**A Formal Model for XACML Policies:** Let \( R = \{ \text{Per}, \text{Den}, \text{NoA}, \text{Ind} \} \) be the set of valid results permit, deny, not applicable and indeterminate, respectively. We define the set of valid policies \( P \) as follows (semantics will be defined below):

\[
\text{Per} \in P, \text{Den} \in P
\]
\[
\forall p \in P : \forall S \subseteq E : \text{Sco}(p, S) \in P \land \text{Err}(p, S) \in P
\]
\[
\forall p, q \in P : p \oplus q \in P \land p \ominus q \in P \land p \otimes q \in P \land p \odot q \in P
\]

Informally, we regard \( \text{Per} \) and \( \text{Den} \) as basic policies that ignore the environment and always yield \( \text{Per} \) or \( \text{Den} \), respectively. Along these same lines, \( \text{Sco} \) and \( \text{Err} \) attach conditions to policies depending on the environment they are evaluated in: \( \text{Sco}(p, S) \) yields \( p \)'s answer if the current environment is in \( S \), or \( \text{NoA} \) otherwise (i.e., \( \text{Sco} \) is used to define the scope of a policy); \( \text{Err}(p, S) \) yields \( \text{Ind} \) if the current environment is in \( S \) or \( p \)'s answer otherwise (i.e., \( \text{Err} \) is used to define the error conditions for a policy). The other four symbols (\( \oplus, \ominus, \otimes, \odot \)) are combinators, that combine two policies as:

- **Permit-overrides:** \( p \oplus q \) always yields \( \text{Per} \) if either \( p \) or \( q \) yield \( \text{Per} \).
- **Deny-overrides:** \( p \ominus q \) always yields \( \text{Den} \) if either \( p \) or \( q \) yield \( \text{Den} \).
- **Only-one-applicable:** \( p \otimes q \) requires that one of \( p \) or \( q \) yield \( \text{NoA} \) and then yields the other half’s answer.
- **First-applicable:** \( p \odot q \) yields \( p \)'s answer unless that answer is \( \text{NoA} \), in which case it yields \( q \)'s answer.

Our \( \otimes \) and \( \odot \) operators are exactly equivalent to the only-one-applicable and first-applicable rules in XACML. However, the \( \oplus \) and \( \ominus \) operators we use in this paper are slightly different than the permit-overrides and deny-overrides rules in XACML. In the cases where the sub-rules do not yield \( \text{Ind} \), these operators are exactly equivalent to the corresponding XACML rules. For the remaining cases, the corresponding XACML rules can be mapped to our operators with some extra work.

We formalize the semantics of these combinators in Figure 2 by defining a function \( \text{eff} : E \times P \rightarrow R \) that, given an environment and a policy, produces a result.

Using this notation, we can now model the XACML policy given in Figure 1 as:

\[
S_0 = \{ (a, v, o) \in E : \forall x \in a \; x < 18 \} \quad (1)
\]
\[
S_1 = \{ (a, v, o) \in E : \forall x \in v \} \quad (2)
\]
\[
S_2 = \{ (a, v, o) \in E : \exists x \in o \; x = \text{vote} \} \quad (3)
\]
\[
S_3 = \{ (a, v, o) \in E : \neg \exists! a_0 \in a \} \quad (4)
\]
\[
S_4 = \{ (a, v, o) \in E : \neg \exists! v_0 \in v \} \quad (5)
\]
\[
r_1 = \text{Err}(\text{Sco}(\text{Den}, S_0), S_3) \quad (6)
\]
\[
r_2 = \text{Err}(\text{Sco}(\text{Den}, S_1), S_4) \quad (7)
\]
\[
p = \text{Sco}(r_1 \ominus r_2 \oplus \text{Per}, S_2) \quad (8)
\]
\[
\text{eff}(e, \text{Per}) = \text{Per} \quad \text{eff}(e, \text{Den}) = \text{Den} \\
\text{eff}(e, \text{Sco}(p, S)) = \begin{cases} 
\text{eff}(e, p) & \text{if } e \in S \\
\text{NoA} & \text{otherwise}
\end{cases} \\
\text{eff}(e, \text{Err}(p, S)) = \begin{cases} 
\text{Ind} & \text{if } e \in S \\
\text{eff}(e, p) & \text{otherwise}
\end{cases} \\
\text{eff}(e, p \oplus q) = \begin{cases} 
\text{Per} & \text{if } \text{eff}(e, p) = \text{Per} \lor \text{eff}(e, q) = \text{Per} \\
\text{Ind} & \text{if } \text{eff}(e, p) = \text{Ind} \land \text{eff}(e, q) \neq \text{Per} \lor \text{eff}(e, q) = \text{Ind} \land \text{eff}(e, p) \neq \text{Per} \\
\lor (\text{eff}(e, q) = \text{Den} \land \text{eff}(e, p) \neq \text{Per} \land \text{eff}(e, q) \neq \text{Ind}) \\
\lor (\text{eff}(e, q) = \text{Per} \land \text{eff}(e, p) \neq \text{Den} \land \text{eff}(e, p) \neq \text{Ind}) \\
\lor (\text{eff}(e, q) = \text{NoA} & \text{otherwise}
\end{cases} \\
\text{eff}(e, p \otimes q) = \begin{cases} 
\text{eff}(e, p) \text{ if } \text{eff}(e, q) = \text{NoA} \\
\text{eff}(e, q) \text{ if } \text{eff}(e, p) = \text{NoA} \\
\text{Ind} & \text{otherwise}
\end{cases} \\
\text{eff}(e, p \otimes q) = \begin{cases} 
\text{eff}(e, p) \text{ if } \text{eff}(e, p) \neq \text{NoA} \\
\text{eff}(e, q) & \text{otherwise}
\end{cases}
\]

Fig. 2. Semantics of policies

where \( S_0 \) is the set of environments that fail the age requirement, \( S_1 \) is the set of environments that fail the voting requirement, \( S_2 \) is the set of environments where someone’s trying to vote, etc. Note that, \( r_1 \) corresponds to the XACML rule between lines 19-27 in Figure 1, \( r_2 \) corresponds to the XACML rule between lines 28-36, and \( p \) corresponds to the XACML policy between lines 2-38.

Policy Transformations: We convert the access control policies to an intermediate normal form before we verify them. This enables us to decouple the verification backend of our tool from its front-end. This decoupling means that it is possible to support other access control languages in our verification framework as long as they can be translated to the same normal form.

We first define the equivalence between two policies:

\[ P_1 \equiv P_2 \text{ iff } \forall e \in E \text{ eff}(e, P_1) = \text{eff}(e, P_2) \]

We call a function \( f \) that takes a policy and returns another policy an eff-preserving transformation if \( \forall p \in P \text{ eff}(f(p)) \equiv p \).

For any given policy, we want to regard the subset of \( E \) that will give a Per result, the subset of \( E \) that will give a Den result, and the subset of \( E \) that will give an Ind result independently. We define the shorthand \( \langle S, R, T \rangle \), where \( S, R \) and \( T \) are pairwise disjoint, as follows:

\[ \langle S, R, T \rangle = \text{Err}(\text{Sco(Per, S)} \otimes \text{Sco(Den, R), T}) \]

Hence, \( \langle S, R, T \rangle \) is simply a policy that yields Per for any environment in \( S \), Den for any environment in \( R \), Ind for any environment in \( T \), and NoA for any remaining environment. We call this triple notation and refer to an \( \langle S, R, T \rangle \) as a triple.

Now that we have a framework for transforming policies, we would like to transform an entire policy with Sco, Err and combinators alike into a single triple. For any policy
A triple $P_T$ that is equivalent to it can be written as:

$$P_T = \langle \{ e \in E : \text{eff}(e, P) = \text{Per} \}, \{ e \in E : \text{eff}(e, P) = \text{Den} \}, \{ e \in E : \text{eff}(e, P) = \text{Ind} \} \rangle.$$  

However, this is not a constructive definition. In [4], we developed an eff-preserving transformation $T : P \rightarrow \mathcal{P}(E) \times \mathcal{P}(E) \times \mathcal{P}(E)$, such that given a policy $p$, $T(p)$ returns a triple that is equivalent to $p$. Our transformation works in two stages. In the first stage, the input policy is transformed to a set of subpolicies in our triple notation combined with $\oplus, \ominus, \otimes$ and $\oslash$. In the second stage, the triples joined by combinators are transformed into a single triple. For example, applying $T$ to the policy $p$ defined in Equation (8) leads to the following:

$$p = \text{Sco}(\text{Err}(\text{Sco}(\text{Den}, S_0), S_3) \ominus \text{Err}(\text{Sco}(\text{Den}, S_1), S_4) \ominus \text{Per}, S_2)$$

$$T(p) = \langle S_2 \setminus (S_0 \cup S_1 \cup S_3 \cup S_4), ((S_0 \setminus S_3) \cup (S_1 \setminus S_4)) \cap S_2, ((S_3 \setminus S_4) \setminus ((S_0 \setminus S_3) \cup (S_1 \setminus S_4))) \cap S_2 \rangle$$

### 3 Properties of Policies

In this section we will show that properties of policies can be expressed using several partial ordering relations. For example, we might want to prove that a (possibly very complex) policy at least protects as much as some simpler policy, and similarly we might want to guarantee that a (possibly very complex) policy does not say anything outside of its scope. Such properties can be expressed using the ordering relations defined below.

Let $P_1 = \langle S_1, R_1, T_1 \rangle$ and let $P_2 = \langle S_2, R_2, T_2 \rangle$ be two policies. We define the following partial orders:

$$P_1 \subseteq_P P_2 \text{ iff } S_1 \subseteq S_2,$$

$$P_1 \subseteq_D P_2 \text{ iff } R_1 \subseteq R_2,$$

$$P_1 \subseteq_E P_2 \text{ iff } T_1 \subseteq T_2.$$

Note that we can define a partial order for any combination of of $P$, $D$ and $E$. We use $P_1 \sqsubseteq D_P P_2$ as a shorthand for $P_1 \subseteq_P P_2 \sqsubseteq_D P_2$. We can regard $P_1 \subseteq D_P P_2$ as stating that for any $e \in E$ where $\text{eff}(P_1, e) \neq \text{NoA}$, $\text{eff}(P_2, e) = \text{eff}(P_1, e)$.

To demonstrate the use of these ordering relations, let us create a new policy for our online voting example. People are permitted to check the current results of the election, for exit polls. We encode this with the following policy

$$S_5 = \{ (a, v, o) \in E : \exists x \in o x = \text{getresult} \}, \quad r_3 = \text{Sco}(\text{Err}(\text{Per}, S_4), S_5)$$

where $S_4$ is defined in Equation (5). Now, we can create a composite policy $p_c = p \oplus r_3$, where $p$ is defined in Equation (8). This policy has a bug—specifically, it permits people under 18 to vote in certain circumstances—and we will demonstrate the usefulness of our technique by showing this. First, we perform our translations on this new policy as above, getting:

$$T(r_3) = \langle S_5 \setminus S_4, \emptyset, S_4 \cap S_5 \rangle$$

$$T(p_c) = \langle ((S_2 \setminus (S_0 \cup S_1 \cup S_3 \cup S_4)) \cup (S_5 \setminus S_4)), ((S_0 \setminus S_3) \cup (S_1 \setminus S_4)) \cap S_2 \setminus (S_4 \setminus S_5), ((S_4 \setminus S_5) \cup ((S_3 \setminus S_4) \setminus ((S_0 \setminus S_3) \cup (S_1 \setminus S_4))) \cap S_2 \setminus (S_2 \setminus (S_0 \cup S_1 \cup S_3 \cup S_4)) \cup (S_5 \setminus S_4)) \rangle$$

383
where \( S_0, S_1, S_2, S_3 \) and \( S_4 \) are from Equations (1) to (5).

Now, we insist that this combined policy deny anyone trying to vote who is under 18. This is itself a policy, which we call \( p_v \):

\[
p_v = (\emptyset, (S_0 \cap S_2) \setminus (S_3 \cup S_4), (S_3 \cup S_4) \cap S_2)
\]

The property we wish to verify here is whether or not \( p_v \sqsubseteq_p c \), i.e., does the policy \( p_c \) deny every input that is denied by \( p_v \). That would mean that everyone trying to vote who is under 18 is denied, and that our policy combination has not done any harm. However, the environmental tuple

\[
e = \langle \{17\}, \{\text{true}\}, \{\text{vote, getresult}\} \rangle
\]

demonstrates that that is not the case. Input \( e \) passes the second part of the Per requirement and so is permitted by \( p_c \) (which means that it is not denied by \( p_c \)) but denied by \( p_v \), i.e., \( e \) demonstrates that \( p_v \not\sqsubseteq D p_c \). The error is that we do not enforce that only one action be given in the third component of the input, and because of this we have the surprising result that someone who is under eighteen and has already voted, but asks for the voting results at the same time as trying to vote will be permitted, and so can cast any number of ballots. To fix this, we could insist upon a new condition, that \( \exists! x \in o \); or we could use \( \otimes \) instead of \( \oplus \), which would ensure that only one of the sub-policies could be definitive on any given point (and so turn \( \text{eff}(e, p_v) \) into an Ind result instead of a Per); or we could decide that only people who have voted already can check the results.

4 Automated Verification

In this section we first formalize the syntax of formulas we use to specify sets of environments. Then we discuss how policies constructed using these formulas and policy combinators can be translated to Boolean logic formulas. After this translation we show that we can check properties of access control policies using a SAT solver.

In Section 2, we defined our formal model using subsets of the set of possible environments \( E \). We showed that each policy can be expressed in triple form \( P = \langle S, R, T \rangle \) where \( S, R, \) and \( T \) are subsets of \( E \). We also declared that all these subsets of \( E \) are either of the form \( \{ e \in E : C \} \), or some combination of subsets of \( E \) using \( \cup, \cap \) or \( \setminus \).

Since \( \{ e \in E : C_1 \} \cup \{ e \in E : C_2 \} = \{ e \in E : C_1 \lor C_2 \} \) and similarly other set operations can also be expressed using logical connectives, we can regard all subsets of \( E \) as of the form \( \{ e \in E : C \} \).

Given a set \( S \) in the form \( S = \{ e \in E : C \} \), our goal is to generate a boolean logic formula \( B \) which encodes the set \( S \). The encoding will map each \( e \in E \) to a valuation of the boolean variables in \( B \), and \( B \) will evaluate to true if and only if \( e \in S \). Based on such an encoding we can convert questions about different policies (such as if one subsumes the other one) to SAT problems and then use a SAT solver to check them. For example, we can generate a boolean formula which is satisfiable if and only if an access policy is not subsumed (i.e., \( \not\sqsubseteq \)) by another one. If the SAT solver returns a satisfying assignment to the formula, then we can conclude that the property is false, and generate a counterexample based on the satisfying assignment. If the SAT solver declares that the formula is not satisfiable then we can conclude that the property holds. We will discuss the details of such a translation below.
Fig. 3. Translation of the basic predicates and the constraints to Boolean logic formulas.

For elements \( e \in E \), we name the components of \( e \{0\}, \ldots, e\{n\} \). We use \( s, s_0, \ldots, s_n \) to denote set variables, \( a, a_0, \ldots, a_n \) to denote scalar variables, and \( A, A_0, \ldots, A_n \) to denote constants. \( BP \) is a set of basic predicates which we define as:

\[
SC \rightarrow A \mid a \\
BS \rightarrow s \mid e [i] \\
SE \rightarrow BS \mid \{ SC \} \mid SE \cup SE \mid SE \cap SE \mid SE \setminus SE \\
BP \rightarrow true \mid false \\
\]

The above grammar is sufficient for specifying policies with finite domain types and the operations \( \neg, =, \in, \subseteq \). We will discuss extension to other domains later in this section.

Assuming that all subsets of \( E \) are specified in the form \( \{ e \in E : C \} \), where there are no free variables save \( e \) in \( C \), \( C \) is defined as follows:

\[
C \rightarrow BP \mid C \cap C \mid C \cup C \mid \neg C \mid \forall a \in BS C \mid \exists a \in BS C \mid \exists! a \in BS C
\]

Recall that we use \( \exists! \) to mean there exists exactly one instance that holds. We can express all set definitions on unordered and enumerated types that are permitted in XACML using the expressions above.

We will explain our translation of a constraint \( C \) defined by the above grammar to a Boolean logic formula using attribute grammars. We will first discuss the translation of the basic predicates \( BP \). In order to simplify our presentation we will assume that domains of all scalar variables have the same size \( k \). We will encode a set of values from any domain using a Boolean vector of size \( k \). Given a Boolean vector \( v \), we will denote its components as \( v [1], v [2], \ldots, v [k] \) where \( v [i] \leftrightarrow true \) means that element \( i \) is a member of the set represented by \( v \) whereas \( v [i] \leftrightarrow false \) means that it is not.

We encode a set variable \( s \) and each component of the environment tuple \( e \) using the
same encoding, i.e., as a vector of Boolean values. To simplify our presentation we also
encode a scalar variable \( a \) as a set using a vector of Boolean values but restrict it to be
a singleton set by making sure that at any time only one of the Boolean values in the
vector can be true. In our actual implementation scalar variables are represented using
\( \log_2 k \) Boolean variables where \( k \) is the size of the domain.

The production rules 1 to 14 in Figure 3 show the attribute grammar for basic predi-
cates. Each production rule has a corresponding semantic rule next to it. The semantic
rules describe how to compute the attributes of the nonterminal on the left hand side of
the production rule using the attributes of the terminals and nonterminals on the right
hand side of the production rule. In the attribute grammar shown in Figure 3, the nonter-
minals \( SC \), \( BS \) and \( SE \) have two attributes. One of them is a Boolean vector \( v \) denoting
a set of values, and the other one is a Boolean logic formula \( f \) which accumulates the
frame constraints.

Rule 1 in Figure 3 states that a scalar constant \( A \) is encoded as a singleton set that
contains only \( A \). This singleton set is represented as a Boolean vector \( v \), such that \( v[A] \)
is set to true and all the rest of the elements of the vector are set to false. This condition
is stored in the frame constraint \( f \). Rule 2 states that a scalar variable is also encoded as
a Boolean vector \( v \). The frame constraint \( f \) makes sure that the elements of the Boolean
vector \( v \) are same as the elements of the Boolean vector representing the scalar variable
\( a \) and exactly one of the elements in \( a \) or \( v \) is set to true in any given time. Rules 3 and 4
show that the set variables (s) and components of the environment tuple (\( e[i] \)) are also
coded as Boolean vectors.

Rule 5 creates a singleton set from a scalar constant \( SC \). However, since we encode
scalar constants as singleton sets, this simply means that the Boolean vectors encoding
the scalar constant (\( SC.v \)) and the set (\( SE.v \)) are equivalent and the frame constraint
\( SE.f \) expresses this constraint. Note that in the attribute grammar shown in Figure 3,
the frame constraint of a nonterminal on the left hand side of a production is a conjunc-
tion of the frame constraints of the nonterminals on the right hand side of the production
plus some other constraints that are added based on the production rule.

Rules 7, 8 and 9 encode the set operations: union, intersection and set difference.
Each set operation on two set expressions \( SE_1 \) and \( SE_2 \) results in the creation of a
new Boolean vector \( SE.v \). The value of an element in \( SE.v \) is defined based on the
corresponding elements in \( SE_1.v \) and \( SE_2.v \). For example, for the union operation,
\( SE.v[i] \) is true if and only if \( SE_1.v[i] \) is true or \( SE_2.v[i] \) is true. The intersection and
set difference are defined similarly.

The nonterminal \( BP \) corresponds to the basic predicates and it has two attributes.
One of them is a boolean variable \( b \) representing the truth value of the predicate and
the other one is a Boolean logic formula \( f \) that accumulates the frame constraints.
Rules 10 and 11 create two basic predicates which have the truth value true and false,
respectively. Rule 12 is a basic predicate that corresponds to an equality expression
comparing two scalars. Since scalars are expressed as Boolean vectors, the Boolean
variable encoding the truth value of the predicate is true if and only if all elements
of the Boolean vectors encoding the two scalar values are the same. This constraint is
added to the frame constraint of the basic predicate.
Rule 13 creates a basic predicate that corresponds to a membership expression testing membership of a scalar to a set expression. Rule 14 creates a basic predicate that corresponds to a subset expression testing if a set expression is subsumed by another set expression. Since we encode scalars as singleton sets, the frame constraints generated for rules 13 and 14 are very similar. They state that if a value is a member of the set on the left hand side, then it should also be a member of the set on the right hand side.

The production rules 15 to 21 in Figure 3 show the attribute grammar for the constraints. The nonterminal $C$ has two attributes. One of them is a boolean variable $b$ representing the truth value of the constraint, and the other one is a Boolean logic formula $f$ that accumulates the frame constraints. Again, the frame constraint of a nonterminal on the left hand side of a production is a conjunction of the frame constraints of the nonterminals on the right hand side of the production plus some other constraints that are added based on the production rule.

Rule 15 is just a syntactic rule expressing that a constraint can be a basic predicate. Rule 16 defines the negation operation. As expected the frame constraint states that the value of the constraint on the left hand side of the production rule is the negation of the value of the constraint on the right hand side of the production rule. Rules 17 and 18 define the disjunction and conjunction operations. The frame constraints generated in Rules 17 and 18 state that the value of the constraint on the left hand side of the production rule is the disjunction or the conjunction of the values of the constraints on the right hand side of the production rule, respectively.

Rules 19, 20 and 21 deal with quantified constraints. In the rules 19, 20 and 21, $a$ denotes a scalar variable which is quantified over a basic set expression $BS$ which is either a set variable $s$ or a component of the environment tuple $e[i]$. The quantified variable $a$ can appear as a free variable in the constraint expression on the right hand side ($C_1$). Universal quantification is expressed as a conjunction which states that for all the members of the set $s$ or $e[i]$, the constraint $C_1$ should evaluate to true. This is achieved by restricting the value of the scalar variable $a$ to the value of a different member of the set for each conjunct. Existential quantification is expressed similarly as a disjunction by restricting the value of the scalar variable $a$ to the value of a different member of the set for each disjunct.

Rule 21 is an existentially quantified constraint which evaluates to true if and only if the constraint $C_1$ evaluates to true for exactly one member of the set $s$ or $e[i]$. This is expressed by first stating that there is at least one member of the set $s$ or $e[i]$ for which the constraint $C_1$ evaluates to true (which is equivalent to existential quantification) and then adding an extra conjunction which states that the constraint $C_1$ does not evaluate to true for two different members of the set $s$ or $e[i]$.

The translation we described above does not handle domain specific predicates, e.g., ordering relations on types such as integers. When we translate sets described using such predicates to boolean logic formulas, we represent them as uninterpreted Boolean functions. We create a Boolean variable for encoding the value of the uninterpreted boolean function and we generate constraints which guarantee that the value of the function is the same if its arguments are the same. Other than this restriction the variables encoding the functions can get arbitrary values. Note that this introduces some
imprecision to our analysis. It is possible that counterexamples may be spurious, and will need to be validated against the original policy.

Note that, it is possible to fully interpret ordering relations in order to reduce the imprecision in the analysis. We can encode a type with a domain of $n$ ordered elements using $n^2$ boolean variables, one for each pair of values in the domain, representing the ordering relations. However, XACML uses many complex functions such as XPath matching and X500 name matching which can be lead to very complex formulas if one tries to fully interpret them in the Boolean logic translation. Hence, we believe that using uninterpreted functions for abstracting such complex functionality is a justified approach which enables us to handle a significant portion of the XACML language. Also, we would like to note that the imprecision caused by abstraction of such complex functions has not led to any spurious results in the experiments we performed so far.

**Property Verification:** As discussed in Section 3, we specify properties of policies using a set of partial ordering relations. These partial ordering relations can be used to state that a certain type of outcome for one policy subsumes the same type of outcome for another policy. In this section we will only focus on the $\sqsubseteq$ relation. Translation of properties specified using other relations are handled similarly.

Given a query like $P_1 \sqsubseteq P_2$, our goal is to generate a Boolean logic formula which is satisfiable if and only if $P_1 \not\sqsubseteq P_2$. As we discussed earlier our tool first translates policies $P_1$ and $P_2$ to triple form, such that $P_1 = \langle S_1, R_1, T_1 \rangle$ and $P_2 = \langle S_2, R_2, T_2 \rangle$ where each element of each triple is specified with a constraint expression as follows:

\[
S_1 = \{ e \in E : C_{S_1} \}, \quad R_1 = \{ e \in E : C_{R_1} \}, \quad T_1 = \{ e \in E : C_{T_1} \}
\]

\[
S_2 = \{ e \in E : C_{S_2} \}, \quad R_2 = \{ e \in E : C_{R_2} \}, \quad T_2 = \{ e \in E : C_{T_2} \}
\]

After translating policies $P_1$ and $P_2$ in to the triple form our translator generates boolean logic formulas for the constraints $C_{S_1}$, $C_{R_1}$, $C_{T_1}$, $C_{S_2}$, $C_{R_2}$ and $C_{T_2}$ based on the attribute grammar rules described in Figure 3. For example, after this translation the truth value of the constraint $C_{S_1}$ is represented with the Boolean variable $C_{S_1}.b$ and the frame constraint $C_{S_1}.f$ states all the constraints on the Boolean variable $C_{S_1}.b$.

Recall that, given $P_1 = \langle S_1, R_1, T_1 \rangle$ and $P_2 = \langle S_2, R_2, T_2 \rangle$, $P_1 \sqsubseteq P_2$ holds if and only if $S_1 \subseteq S_2$ and $R_1 \subseteq R_2$ and $T_1 \subseteq T_2$. Based on this, we generate a formula $F$ such that $F = \text{true}$ if and only if $P_1 \not\sqsubseteq P_2$ as follows:

\[
F = (C_{S_1}.f \land C_{R_1}.f \land C_{T_1}.f \land C_{S_2}.f \land C_{R_2}.f \land C_{T_2}.f) \rightarrow
((C_{S_1}.b \rightarrow C_{S_2}.b) \land (C_{R_1}.b \rightarrow C_{R_2}.b) \land (C_{T_1}.b \rightarrow C_{T_2}.b))
\]

Finally, we send the property $\neg F$ to the SAT solver. If the SAT solver returns a satisfying assignment for the Boolean variables encoding the environment tuple $e$ (which are the only free variables in the formula $\neg F$), the satisfying assignment corresponds to a counter-example environment demonstrating how the property is violated. If the SAT solver states that $\neg F$ is not satisfiable, then we conclude that the property holds, i.e., $P_1 \not\sqsubseteq P_2$.

We could use this same translation to verify logical properties of a policy directly at the cost of introducing a new language that our users would be forced to learn. We feel that subsumption is sufficiently powerful and the advantages of using only one language are sufficiently compelling that we do not support this at this time.

Since the majority of the SAT solvers expect their input to be expressed in Conjunctive Normal Form (CNF), the last step in our translation (before we send the formula
\(\neg F\) to the SAT solver) is to convert \(\neg F\) to CNF. For conversion to CNF we have implemented the structure preserving technique from [10].

5 Experiments

Our tool generates a Boolean formula in Conjunctive Normal Form (CNF), which we then give to a SAT solver; in particular, we use the zchaff [9] tool. To demonstrate the value of our tool we conducted some experiments. One of the policies we used for our experiments is the CONTINUE example [7], encoded into XACML by Fisler et al. [3]. CONTINUE is a Web-based conference management tool, aiding paper submission, review, discussion and notification. In addition, we used the Medico example from the XACML [12] specification, which models a simple medical database meant to be accessed by physicians. Finally, we have encoded our online voting example from Section 3 into XACML and applied our tool to the discovery of the error which we know to exist. We tested 11 properties:

- C1 tests whether the conference manager denies program committee chairs the ability to review papers he/she has a conflict with.
- C2 and C7 test properties concerning reviews for papers co-authored by program committee members.
- C3 and C8 test properties concerning access to the conference manager if the user has no defined role.
- C4 and C5 test properties regarding read access to information about meetings.
- C6 tests whether program committee members can read all parts of a review.
- C9 tests which roles can set meetings.
- C10 and C11 test under what conditions program committee members can see reviews.
- M1 and M2 test whether the unified Medico policy upholds the required access properties about the medical records.
- V1 is the voting property we discussed in Section 3.

The performance results shown in Table 1 indicate that analysis time is dominated by the initial parsing of the policies and by the conversion from triple form to a Boolean formula; sometimes the Boolean conversion is strongly dominant, as in the Medico examples. The resulting formulas are unexpectedly small and analysis time is so small the

<table>
<thead>
<tr>
<th>Property</th>
<th>IO Transform Boolean SAT</th>
<th>Lines of XACML</th>
<th>Variables</th>
<th>Clauses</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1.35s 0.17s 1.35s 0.11s</td>
<td>131157</td>
<td>56</td>
<td>114</td>
<td>Property holds</td>
</tr>
<tr>
<td>C2</td>
<td>2.07s 0.19s 1.41s 0.39s</td>
<td>13108</td>
<td>51</td>
<td>108</td>
<td>Property holds</td>
</tr>
<tr>
<td>C3</td>
<td>1.88s 0.16s 1.36s 0.12s</td>
<td>13103</td>
<td>52</td>
<td>106</td>
<td>Property holds</td>
</tr>
<tr>
<td>C4</td>
<td>1.94s 0.17s 1.33s 0.11s</td>
<td>13108</td>
<td>79</td>
<td>166</td>
<td>Property holds</td>
</tr>
<tr>
<td>C5</td>
<td>1.82s 0.16s 2.00s 0.16s</td>
<td>13108</td>
<td>79</td>
<td>166</td>
<td>Property holds</td>
</tr>
<tr>
<td>C6</td>
<td>2.12s 0.15s 2.53s 0.15s</td>
<td>13150</td>
<td>89</td>
<td>190</td>
<td>Property fails</td>
</tr>
<tr>
<td>C7</td>
<td>2.56s 0.34s 3.70s 0.10s</td>
<td>13203</td>
<td>95</td>
<td>218</td>
<td>Property fails</td>
</tr>
<tr>
<td>C8</td>
<td>1.99s 0.18s 1.21s 0.11s</td>
<td>13101</td>
<td>42</td>
<td>83</td>
<td>Property fails</td>
</tr>
<tr>
<td>C9</td>
<td>1.92s 0.16s 1.49s 0.11s</td>
<td>13107</td>
<td>51</td>
<td>106</td>
<td>Property fails</td>
</tr>
<tr>
<td>C10</td>
<td>1.88s 0.19s 3.47s 0.11s</td>
<td>13107</td>
<td>108</td>
<td>250</td>
<td>Property fails</td>
</tr>
<tr>
<td>C11</td>
<td>1.89s 0.16s 5.18s 0.15s</td>
<td>13151</td>
<td>129</td>
<td>297</td>
<td>Property fails</td>
</tr>
<tr>
<td>M1</td>
<td>0.75s 0.02s 15.10s 0.22s</td>
<td>457</td>
<td>109</td>
<td>280</td>
<td>Property holds</td>
</tr>
<tr>
<td>M2</td>
<td>1.00s 0.03s 14.78s 0.13s</td>
<td>405</td>
<td>108</td>
<td>279</td>
<td>Property holds</td>
</tr>
<tr>
<td>V1</td>
<td>0.73s 0.14s 5.86s 0.12s</td>
<td>102</td>
<td>52</td>
<td>123</td>
<td>Property fails</td>
</tr>
</tbody>
</table>

Table 1. Results for the CONTINUE (C1-11), Medico (M1-2) and voting (V1) examples.
startup and I/O overhead of the zchaff tool is probably dominating. This was unexpected; our tool goes to some length to simplify the Boolean formula on the assumption that run times would be dominated by the SAT solver. The results show that our assumption was wrong. These results are very encouraging in terms of the scalability of the proposed approach. Among the different components of our analysis, SAT solving is the one with worst case complexity. Since the examples we tested so far were easily handled by the SAT solver we believe that our approach will be feasible for analysis of very large XACML policies.

There appears to be no relationship between lines of XACML and the number of Boolean formulas required to represent them, which is counterintuitive. This reflects a difference in structure between the Medico and voting example and the CONTINUE conference manager. The CONTINUE conference manager was written by Fisler et al. [3] for their Margrave tool, which supports only simple conditionals in the `<Target>` block of an XACML specification. Accordingly, the policy files require far more text to describe simple Boolean combinations than would be the case if `<Condition>` elements were used. We use their example because it is the largest XACML example that we could find, but it is instructive that the Medico example from the XACML specification is as or more complex despite using an order of magnitude less lines of XACML.

The number of variables in our Boolean formulas is quite large, approximately half the number of clauses. We have made a deliberate tradeoff to get this; our translation machinery from Section 4 introduces large numbers of tightly constrained variables, and our CNF conversion uses the structure preserving technique [10] which generates even more variables. In exchange we get a relatively small formula, and the search space is not so large as might be presumed because of the constraints. A different CNF conversion that would embody a different tradeoff between the CNF conversion and SAT solving might be worth exploring.

Our experimental results clearly demonstrate that the subsumption property is practical to analyze, and we believe total runtime could be lowered by optimizing the Boolean formula generation and CNF transformation steps.

6 Related Work

There has been earlier work on automated analysis of access control policies. [11] and [14] analyze role based access control schemas using the Alloy analyzer. However [11] uses Alloy to verify that the composition of specifications is well formed and is silent about their content, whereas we introduce a formal model of and a partial ordering on XACML specifications specifically designed for analyzing the semantics. Zao, Wee et al. [14] model RBAC schema in Alloy and then check these models against predicates, also written in Alloy. We introduce a formal model for XACML with a partial ordering on policies that we then automatically check using a SAT solver as a back end; we do not insist that the user write predicates in another language and operate solely on XACML.

The Alloy Analyzer also uses a SAT solver as a back-end to solve verification queries [5, 6]. Hence, translating XACML policies to Alloy in order to verify them is in effect an indirect way of using a SAT solver for verification. In fact, we also used Alloy analyzer for verification of XACML policies in our earlier work [4]. However our experience has shown that a direct translation to SAT is much more effective then translating
the verification queries to Alloy. In our experience the Alloy Analyzer is not always capable of dealing with the sizes of problems we are dealing with. It is certainly the case that our direct translation generates a customized encoding of the problem, whereas the translation from the Alloy Analyzer is optimized for a more general class of models; hence it may not necessarily be efficient for types of verification queries we are interested in. Mankai and Logrippo [8] also use Alloy Analyzer to analyze interactions and conflicts among access control policies expressed in XACML. The translation to Alloy appears to have been done by hand, in contrast to our automated translator; as well, cited runtimes are around a minute for simple policies whereas our current approach takes seconds to analyze the most complex policies we could find.

Zhang, Ryan and Guelev [15] have developed a new language named RW, on which they can perform verification through an external program, and which can be compiled to XACML. It is not obviously possible to translate arbitrary XACML policies to RW, and so no analysis on arbitrary XACML policies can be done within their framework, unlike ours.

Bryans [2] modeled XACML using the Communicating Sequential Processes (CSP) process algebra, and then used the FDR model checker to provide some automatic verification, including comparing policies. Bryans uses process interleavings to model rule and policy combination operations which is likely to add unnecessary nondeterminism and increase the state space. In fact, Bryans does not handle all policy combination operations in XACML due to efficiency concerns.

Recently, Fisler et al. [3] used multi-terminal decision diagrams to verify properties of XACML policies with the Margrave tool. Verification queries in [3] are expressed in the Scheme language. We use relationships between policies instead, and since this does not require learning a separate query language, we believe this makes our tool easier to use. Margrave does not handle as much of XACML as we do, and so our tools are not directly comparable; we handle more datatypes, and also complex conditionals as in <Condition> elements, whereas Margrave can only handle simple conditionals in the <Target> block. For example, the predicate \( x < 18 \) as in our subpolicy \( S_0 \) cannot be expressed in Margrave, not even as an uninterpreted Boolean variable, because it can only be written in a <Condition> element. Of our examples, none of M1, M2 or V1 can be expressed using Margrave. The verification underpinnings of our tools are also different; a verification approach that uses decision diagrams is more likely to be successful for incremental analysis techniques, and so are probably the appropriate representation to use for the change-impact analysis presented in [3]. However, for the type of verification queries we discuss in this paper we expect a verification approach based on SAT solvers to perform better than a verification approach based on decision diagrams.

Agrawal et al. [1] discuss techniques for analyzing interactions among policies and propose algorithms for policy ratification. They use techniques from constraint, linear and logic programming domains for policy analysis. Compared to the approach presented in [1] we focus on the XACML language and use a Boolean SAT solver for analysis. Unlike the approach discussed in [1], we are not proposing new mechanisms for combining different policies. Rather, the approach we present in this paper is useful for automated analysis of existing policies and finding possible errors in them.
7 Conclusions

We have presented a formal model for access control policies, and shown how to verify interesting properties about such models in an automated way. In particular we translate queries about access control policies to Boolean satisfiability problems and use a SAT solver to obtain an answer. We express properties about access control policies as subsumption queries between two policies. We implemented a tool that implements the proposed approach and our experimental results indicate that automated verification of nontrivial access control policies is feasible using our approach.

References