Aspect-Oriented Modeling of Ubiquitous Web Applications: The *aspectWebML* Approach

A. Schauerhuber*
Women's Postgraduate College for Internet Technologies
Vienna University of Technology, Austria
andrea@wit.tuwien.ac.at

M. Wimmer
Business Informatics Group
Vienna University of Technology, Austria
wimmer@big.tuwien.ac.at

E. Kapsammer
W. Retschitzegger
Information Systems Group
University of Linz, Austria
{ek,werner}@ifs.uni-linz.ac.at

W. Schwinger
Department of Telecooperation
University of Linz, Austria
wieland@schwinger.at

ABSTRACT
Ubiquitous web applications (UWA) are required to be customizable, meaning that their services need to be adaptable towards the context in which they are used, indicated by, e.g., user, location, time, and device. Considering UWA's from a software engineering point of view, a systematic development on basis of models is crucial. Current web modeling languages, however, often disregard the crosscutting nature of customization potentially affecting all parts of a web application, i.e., its content, hypertext and presentation levels, and often tangle customization functionality and other, non-ubiquitous core services of a web application. This leads to inefficient development processes, high maintenance overheads and a low potential for reuse.

To cope with this, we regard customization as a crosscutting concern in the sense of the aspect-oriented paradigm. As a proof of concept, we extend the prominent web modeling language WebML on basis of our reference architecture for aspect-oriented modeling. This allows for customization mechanisms to influence all parts of a web application, maintaining at the same time a clear separation between the core services and customization functionality, and – as a spin-off – demonstrates how to bridge existing (domain-specific) modeling languages with aspect-oriented concepts.

Categories and Subject Descriptors
D.2.2 Design Tools and Techniques

General Terms
Design, Standardization, Languages, Theory

Keywords
Aspect-oriented modeling, domain-specific language, web application modeling, adaptation, context-awareness

1. INTRODUCTION
With the emergence of mobile devices as new access channels to the Internet, we are now facing a new generation of web applications, called ubiquitous web applications. UWAs are characterized by the *anytime/anywhere/anymedia paradigm*, taking into account that services are not exclusively accessed through traditional desktop PCs but through mobile devices with different capabilities, by users with various interests at anytime from anyplace around the globe. Services provided by UWAs are adapted to the actual context of use in order to preserve or even enhance their semantic value for users. Thus, knowing the context, e.g., user, location, time, and device, and providing *adaptation operations* for web pages and their different kinds of contents, e.g., text, images, and links, are the main prerequisites for *customization* of web applications towards ubiquity. Customization then denotes the *mapping* of the required *adaptation* of an application's services with respect to its *context* [6].

Considering UWA's from a software engineering point of view, a systematic development on basis of models is crucial. There are already some approaches dealing with the ubiquitous nature of web applications and the model-driven development thereof, the most prominent examples being WebML [3], UWE [7], and OO-H [5][1]. Concerning customization modeling, however, they are still in their early stages due to the following reasons. First, the provided customization mechanisms frequently do not allow to deal with all different parts of a web application in terms of its content, hypertext and presentation levels and their structural and behavioral features (cf. Figure 1), thus, disregarding the crosscutting nature of customization. Second, customization is often tangled with the core web application, thus, neither a *context model* nor *adaptation operations* enter web application models in an explicit, self-contained and extensible way. This leads to inefficient development processes, high maintenance overheads and a low potential for reuse.

---

[*] 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] For an overview of methods and tools for web application development we refer to [17].

To cope with these problems, we propose *aspectWebML* using aspect-orientation as driving paradigm to incorporate customization in ubiquitous web applications at the modeling level [13] (cf. Figure 1). As a proof of concept, we use our reference architecture for aspect-oriented modeling (cf. [14] and [15]), which describes the necessary concepts of aspect-oriented modeling (AOM), as a blueprint for extending the MOF-based [9] metamodel of WebML [16], a prominent domain-specific language for modeling data-intensive web applications.

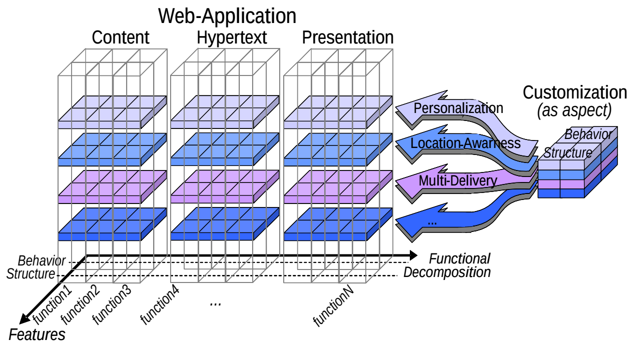

Figure 1: Customization as an Aspect

The benefits of this approach are fourfold. First, it takes into account the crosscutting nature of customization, allowing to influence all parts of a web application. Second, despite this omnipresence, a clear separation between the core services and customization functionality can be maintained. The core services of the web application remain oblivious to the need for customization, allowing even to make existing, non-ubiquitous web applications context-aware. Third, while our motivation for extending WebML has been driven by the need to separately capture customization, the extensions made also allow modeling of other aspects than the customization aspect. Finally, as a spin-off, it demonstrates how to bridge existing (domain-specific) modeling languages with aspect-oriented concepts.

The remainder of this paper is organized as follows. In Section 2 we outline our contributions with respect to related work and briefly introduce the WebML language using as a running example a *Museum* web application in Section 3. In Section 4, we report on how to bridge WebML to AOM according to the AOM reference architecture and present the specific AOM extensions to the WebML metamodel in terms of *aspectWebML*. In Section 5, we compare the original modeling approach of WebML with *aspectWebML* by extending a *Museum* web application with customization functionality and report on our prototype modeling editor. Finally, we conclude with an outlook on future work in Section 5.

2. RELATED WORK

Currently, the majority of AOM approaches is first, based on UML and second, designed as general-purpose languages with respect to the application domain [15]. We currently know of three UML-based approaches specific to a certain domain. In [4] and [12] two UML profiles have been proposed, the first one for modeling the notification aspect in CORBA applications and the second one for AOM in the web service domain. A third approach applies AOM in the domain of web application modeling [1], [20]. More precisely, while in [20], the UML-based web modeling language UWE has been extended with aspect-oriented concepts to model the *access control aspect* in web applications, the approach presented in [1] is closely related to our work in that it identifies *adaptivity* as a crosscutting concern in web applications. In particular, an extension of UWE's metamodel with aspect-oriented modeling techniques has been proposed and allows making navigation in web applications adaptive. Our approach, however, differs in three ways. First, we are building on a lean MOF-based metamodel of WebML, which has been established during our previous work [16], thus avoiding the unnecessary overhead of the huge UML metamodel. Second, modeling customization in UWE [1] currently is limited to the hypertext level of web applications and does neither support the content level nor the presentation level. Third, the aspect-oriented extensions applied to UWE are tailored to a specific aspect, only, being the access control aspect 0 and the navigation adaptivity aspect [1], respectively. In contrast to that, our approach is to use the AOM reference architecture as a blueprint to extend the WebML metamodel with AOM concepts, thus, allowing to model different aspects with one coherent set of concepts.

3. A WebML PRIMER

WebML is one of the most prominent modeling languages in the web modeling field due to existing tool support including a model editor, a code generation facility, and a runtime environment in form of the commercial WebRatio tool[2] and applications in real world projects. Following, we give a brief introduction into its modeling concepts using a *Museum* web application as a running example. The *Museum* web application is based on [2] and will be extended with customization functionality in Section 5.

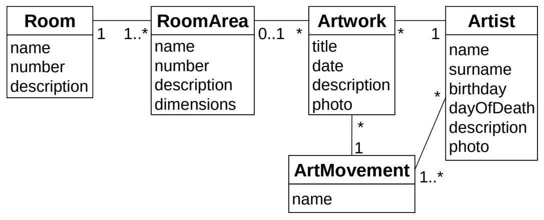

Figure 2: Museum Content Model[3]

The content level of the *Museum* web application is represented by the *content model*, which – in WebML – is based on the Entity-Relationship model (cf. Figure 2). The museum possesses a collection of *Artworks*, some of them being exhibited in certain *RoomAreas* of one of the museum's *Rooms*. A specific piece of Artwork belongs to a certain *ArtMovement* and has been created by a certain *Artist*.

The *hypertext model* of the *Museum* web application is based on the content model. Figure 3 shows eight web *Pages*, the majority of them containing so called *ContentUnits*, which allow to query the content model and to display the result on the Page. The *Home* Page links four Pages. The *RoomList*, *ArtworkList*, and *ArtistList* pages each contain one ContentUnit, a so called *IndexUnit*, which presents multiple instances of an entity type from the content model as a list. From these IndexUnits, a user then can navigate to the *RoomDetails*, *ArtworkDetails*, and *ArtistDetails* Pages, presenting further information according to a single instance of a

[2] http://www.webratio.org and http://www.webml.org

[3] For readability purposes the UML notation for multiplicities is used in this paper.

Room, an *Artwork*, or an *Artist*. For example, the *RoomDetails* Page contains information about the *Room* itself, which is derived from the content model using a so called *DataUnit* named *Room*, i.e., a ContentUnit. ContentUnits select the information from the content model using a *Selector*, e.g., *Room* for DataUnit *Room*, and optionally several *SelectorConditions* depicted in square brackets. Additionally, the Page contains two IndexUnits listing *Artists* and *Artworks* exhibited in the specific *Room*. WebML also provides the container concept *Area*, which allows grouping Pages that deal with some related topic [3].

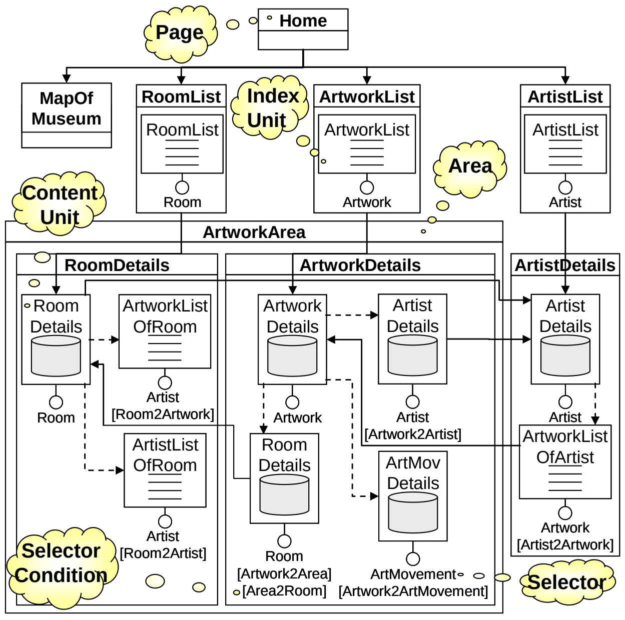

Figure 3: Museum Hypertext Model[4]

4. BRIDING WebML TO ASPECT-ORIENTED MODELING

In this work, we make a step towards bridging WebML to AOM using as a basis our AOM reference architecture. Subsequently, we briefly introduce the WebML metamodel in Section 4.1 and our AOM reference architecture in Section 4.2. In Section 4.3, we provide detailed information on how we applied the AOM reference architecture to the WebML language.

4.1 The WebML Metamodel

A prerequisite for bridging WebML to AOM is the existence of a proper metamodel of the web modeling language, which allows to seamlessly hook up the aspect-oriented concepts. Similar to most web modeling languages, WebML – originally focusing on notational aspects – has been designed without using expressive object-oriented meta-modeling techniques, employing DTD's, only [19]. To further complicate things, recent WebML language concepts – most notably its customization mechanisms [2] – have not been introduced into the WebML DTD but rather hard-coded directly within the WebML modeling tool. To cope with these problems, in previous work [16], we semi-automatically constructed a MOF-based metamodel draft for WebML on basis of the WebML DTD. For our purpose of modeling UWA's, we manually extended this metamodel by introducing also WebML's concepts for customization (cf. Section 5.1)[5].

4.2 The AOM Reference Architecture

Our primary goal in designing the AOM reference architecture [14], [15] was to establish a common understanding in the field of AOM. The reference architecture has been defined in terms of a UML class diagram [11] and identifies the basic ingredients of aspect-orientation, abstracted from specific modeling languages. In this respect, it captures the important AOM concepts, their interrelationships and even more importantly their relationships to an arbitrary modeling language, e.g., a general-purpose modeling language such as UML or any other domain-specific modeling language such as WebML. The AOM reference architecture, however, does not represent a language specification in terms of a metamodel itself, but rather can be used as a blueprint for designing new AOM languages or for extending existing (domain-specific) modeling languages with concepts of the aspect-oriented paradigm.

The AOM reference architecture comprises four major building blocks, each subsuming related concepts (cf. Figure 4). In the following we point out the most important concepts and refer the interested reader to [14], [15].

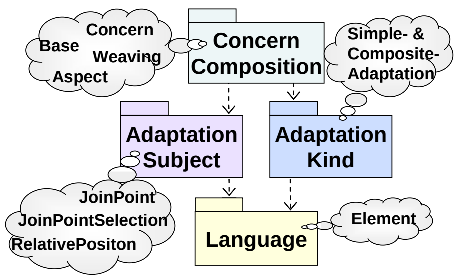

Figure 4: AOM Reference Architecture

The *ConcernComposition* package deals first, with the separation of a system's *Concerns* into appropriate units of modularization, i.e., *Base* and *Aspect*, and second, with their interrelationships, i.e., their composition by means of a *Weaving* specification. In the *AdaptationSubject*, we summarize concepts for identifying *where* to introduce an aspect's adaptation including *JoinPoint*, *JoinPointSelection*, and *RelativePosition*[6]. The *AdaptationKind* package subsumes concepts to describe *how* an aspect adapts a concern, i.e. *Adaptation*. Finally, the *Language* package represents the language including its modeling *Elements* to be extended with aspect-oriented concepts.

4.3 The *aspectWebML* Metamodel

For designing *aspectWebML* we used our AOM reference architecture as a *basis*, meaning that its concepts and their interrelationships have not been adopted one-to-one. This is due to reasons concerning syntax on the one hand and reasons concerning design goals on the other hand. First, the AOM

[4] Please note, that the clouds in the Figure 3 represent comments and are not part of the hypertext model.

[5] The WebML's metamodel versions and a change log are published at http://big.tuwien.ac.at/projects/aspectwebml/.

[6] A relative position denotes where to insert an aspect's adaptation relative to a join point, e.g., *before*, *after*, and *around*.

reference architecture has been defined in terms of a UML class diagram, while the WebML metamodel is MOF-based. Thus, we had to capture concepts available in UML, only, differently in the MOF-based *aspectWebML* metamodel. For example, we had to resolve association classes and replace aggregation associations with either composition associations or references. Second, in order to keep the language simple for the time being, we made some design decisions resulting in a more restrictive AOM language compared with our AOM reference architecture. For example, we currently allow aspects to be woven into bases but not into aspects (cf. Figure 5).

4.3.1 The WebML Package

The AOM reference architecture assumes the modeling language to have a root element from which every modeling concept of the language inherits. This is necessary, since first, both *Base* and *Aspect* including its *Adaptations* are formalized by any set of modeling elements of the modeling language (cf. Figure 5: containment references from *Concern* to *ModelElement* and from *Aspect* to *Adaptation*), and second, *JoinPoints*, i.e., the locations where an aspect introduces its adaptations, are representations of elements of the modeling language. Since WebML originally did not provide such a root element, we reorganized the metamodel by introducing the abstract meta class *ModelElement*[7], having an attribute *isAdaptable* of type Boolean. This attribute – if set to *true* – allows to define the join point model of the AOM language, i.e. the meta classes of the modeling language that are allowed to serve as join points for aspects. Currently, we are still investigating what kinds of adaptations in terms of aspect are meaningful within the realms of *aspectWebML*. Thus, we did not yet restrict the join point model to a subset of WebML's modeling concepts, meaning that every modeling concept can be subject of adaptations in *aspectWebML* models. This decision also reflects the ongoing discussion about join point models and adaptation effects in AOM.

4.3.2 The ConcernComposition Package

A model in *aspectWebML* consists of *Concerns*, which are either an instance of *Base* or of *Aspect*. An *Aspect* can be woven in to a *Base* by means of a *Weaving* specification. More specifically, the *Weaving* has *AdaptationRules*, which determine *where* (cf. Section 4.3.3) the *Aspect's Adaptations* have to be introduced in the *Base* and what kind of effect (cf. *AdaptationEffectKind* in Figure 5) these *Adaptations* imply.

4.3.3 The AdaptationSubject Package

The adaptation hooks of a *Base* are represented by *JoinPoints*, which are identified by a *SimpleJoinPointSelection*[9]. In addition, an *AdaptationRule* optionally may specify a *RelativePosition* where to insert *Adaptations* with respect to the selected join points. For reuse purposes, we allow *SimpleJoinPointSelections* to be composed to *CompositeJoinPointSelections* by means of *AND* and *OR* operators. Currently, our mechanism to select join points is limited to a manual identification of each single join point. Thus, for defining an instance of *SimpleJoinPointSelection* at modeling level, the user will instantiate join points from *JoinPoint* and link them to instances of *ModelElement*. The investigation of more elaborated join point selection mechanisms, such as OCL [10] or Join Point Designation Diagrams (JPDD) [1], and their applicability in *aspectWebML* is subject to future work.

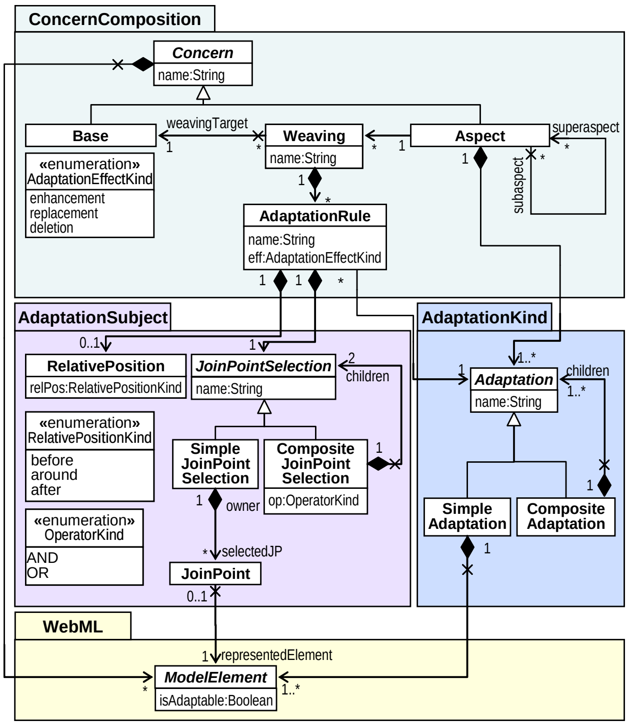

Figure 5: The *aspectWebML* Metamodel

4.3.4 The AdaptationKind Package

Adaptations consist of WebML *ModelElements*. For reuse purposes we distinguish between *SimpleAdaptations* and *CompositeAdaptations*, the latter allowing to combine existing *Adaptations* to form more complex ones.

5. MODELING CUSTOMIZATION

In this section, we show how customization of the *Museum* web application (cf. Section 3) currently can be modeled with the original WebML language and point out the specific problems of the approach in Section 5.1. In Section 5.2, we present how to model the same application using *aspectWebML* and report on the prototype implementation of a model editor for *aspectWebML*[10].

5.1 Modeling Customization in WebML

In [2], WebML has recently been extended with concepts for modeling context-awareness, illustrated in a *Museum* web application example for which also a demo implementation has been provided[11]. Following, we explain the necessary extensions

[7] While this represents an elegant solution, it required a change of WebML's metamodel. This could be avoided by simply duplicating all necessary references, e.g., from JoinPoint to the required modeling element of the language.

[9] A join point selection corresponds to the concept of a *pointcut*.

[10] The *aspectWebML* model editor and the Museum web application example can be downloaded from http://big.tuwien.ac.at/projects/aspectwebml/.

[11] http://dblambs.elet.polimi.it/Demos/indexen.htm

to the original application (cf. Section 3) in order to model location-awareness, i.e., customization according to the location context. In particular, we want to model the following situation: If the visitor requests the *ArtworkDetails* Page, the specific *Artwork* of the *RoomArea* the visitor is currently in, shall be displayed. If, however, no *Artwork* is exhibited in the visitor's *RoomArea*, the visitor is redirected to the *RoomDetails* Page, which presents information about the room the visitor is currently in. In addition, the same set of adaptations shall be applied, if the visitor requests the *RoomDetails* Page.

It is assumed that an RFID-based location-sensing mechanism is available in the museum, that each visitor – or rather the mobile device s/he is using – has a unique RFID tag, and that the location-sensing infrastructure will continuously update the content model with the visitor's current location information.

5.1.1 *Customization in the Content Model*

In WebML, the required context information is simply added to the ContentModel in terms of new Entity types, their Attributes, and their Relationships. In the *Museum* web application, we need to know the user's location, i.e., the *RoomArea*. Thus, a *User* Entity type is introduced having a Relationship with *RoomArea*.

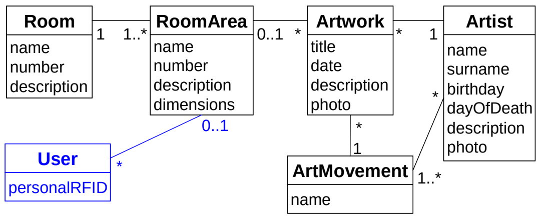

Figure 6: Location-Aware Museum Content Model

5.1.2 *Customization in the Hypertext Model*

In the HypertextModel, we use three of WebML's new concepts for modeling location-awareness: First, *ArtworkArea*, *RoomDetails*, and *ArtworkDetails*, are marked as context-aware Areas and Pages, each having a so called *ContextUnit*. The semantics of *ContextUnits* is that they encapsulate context-aware behavior – also called *context clouds* – of Areas and Pages, which is executed before the actual Page computation, i.e., the computation of *ContentUnits*. When a context-aware Page is requested, then the context clouds of its containers from the outermost to the innermost are evaluated before the Page's context cloud. Second, *GetArea* and *GetArtwork*, so called *GetDataUnits*, allow querying the ContentModel, without displaying the content like other ContentUnits but providing it for further computation to the context cloud. Third, the *IFUnit* represents a control structure, which allows evaluating conditions and thus, may trigger different behavior in the context cloud.

Following, we describe the necessary additions to the HypertextModel of Figure 3 in order to model location-awareness (cf.).

① We add a ContextUnit to *ArtworkArea*, which now retrieves the users's location via *GetArea* every time either the *RoomDetails* or the *ArtworkDetails* Pages are requested. The *currentUser* represents a global parameter in the model, which can be retrieved by *GetUser*, a GetUnit.

② We add a ContextUnit to *ArtworkDetails*, which retrieves the specific *Artwork* of the *RoomArea* the visitor is currently in, using the *GetArtwork* GetDataUnit. If, however, no *Artwork* is exhibited in the visitor's *RoomArea*, the visitor is redirected to the *RoomDetails* Page using an *IFUnit*.

③ We replace the default SelectorCondition[13] of DataUnit *RoomDetails*, which always uses the ID for retrieving an Entity instance of *Room*, with a SelectorCondition *[RoomArea2Room]*, since the *RoomDetails* Page now has to present information about the visitor' current room.

④ The same set of adaptations shall be applied for the *RoomDetails* Page. Thus, we only need to add a ContextUnit to the *RoomDetails* Page which then links to the previously added *GetArtwork* GetDataUnit.

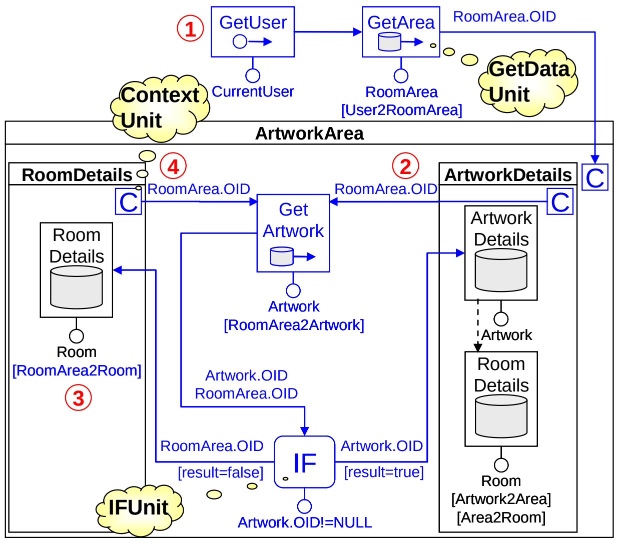

Figure 7: Location-Aware Museum Hypertext Model[14]

5.1.3 *Deficiencies of the WebML approach*

Currently, if customization functionality is introduced to a web application model in WebML by enhancing, replacing, or deleting modeling elements, developers face the following problems: First the original web application model is lost. Second, it is not clear what modeling elements make up customization functionality. And third, customization functionality, that is scattered across WebML models, hampers their readability.

5.2 Modeling Customization in *aspectWebML*

Unlike WebML, *aspectWebML* allows introducing new functionality into all parts of a web application model but – at the same time – maintains a clear separation between the original model and the new functionality in terms of *Aspects* as is exemplified in Figure 8. For want of a concrete syntax for *aspectWebML*, we currently present *aspectWebML* models in

[13] This default SelectorCondition is not shown in WebML models.

[14] For readability reasons, we omitted several parts of the original hypertext model (cf. Figure 3).

terms of UML object models and trees, i.e., our model editor's view (cf. Section 5.2.3).

In Figure 8 (b), we present an overview of the *Museum* web application model defined in *aspectWebML*. This specific *aspectWebML* model consists of the *Museum Base*, i.e., the original *Museum* web application consisting of a *ContentModel*, a *HypertextModel*, and a *PresentationModel* (cf. Section 3), the *Location Aspect*, and the *Weaving* specifying the connections between the *Museum Base* and the *Location Aspect*. In Figure 8 (a), the same information is presented in form of the *aspectWebML* model editor's view.

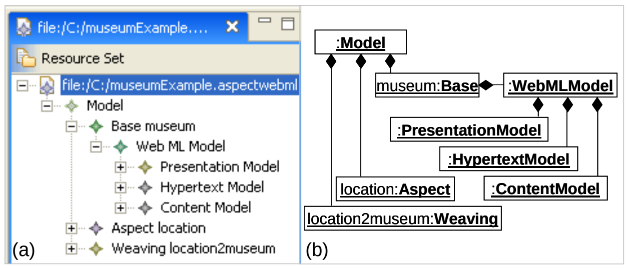

Figure 8: The Location-Aware *Museum* Model

In the following, we present details of both the *Location Aspect* and the specific *Weaving* with respect to necessary adaptations in the content model on the one hand (cf. Section 5.1.1) and in the hypertext model on the other hand (cf. Section 5.1.2).

5.2.1 *Customization in the Content Model*

As in the WebML approach (cf. Section 5.1.1), the ContentModel needs to be extended with a *User* Entity, having an Attribute named *personalRFID* and a Relationship with the *RoomArea* Entity. This is realized using two AdpatationRules (cf. Figure 9):

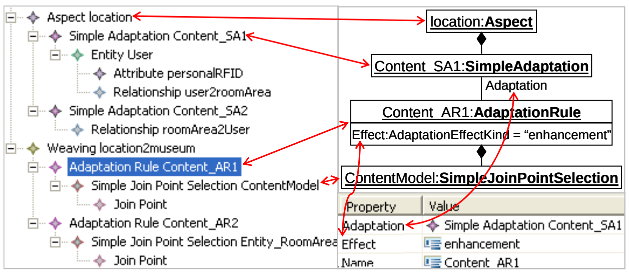
Aspect location
Simple Adaptation Content_SA1
Entity User
Attribute personalRFID
Relationship user2roomArea
Simple Adaptation Content_SA2
Relationship roomArea2User
Weaving location2museum
Adaptation Rule Content_AR1
Simple Join Point Selection ContentModel
Join Point
Adaptation Rule Content_AR2
Simple Join Point Selection Entity_RoomArea
Join Point
location:Aspect
Content_SA1:SimpleAdaptation
Adaptation
Content_AR1:AdaptationRule
Effect:AdaptationEffectKind = "enhancement"
ContentModel:SimpleJoinPointSelection
Property
Value
Adaptation
Simple Adaptation Content_SA1
Effect
enhancement
Name
Content_AR1

Figure 9: The Location-Aware Content Model

a. *Content_AR1* uses SimpleAdaptation *Content_SA1* of the Aspect *Location* to introduce the *User* Entity, its *personalRFID* Attribute, and the uni-directional Relationship *user2roomArea* using the ContentModel as JoinPoint and thus, having an *enhancement* effect.

b. *Content_AR2* uses SimpleAdaptation *Content_SA2* of the Aspect *Location* to introduce uni-directional Relationship *roomArea2user* using the Entity *RoomArea* as JoinPoint and thus, having an *enhancement* effect on the Base.

The reason for modeling two AdaptationRules instead of one is as follows: In WebML, every modeling concept is contained by another one, e.g., ContentModel contains Entity, which contains Relationship and Attribute. However, bi-directional Relationships are realized as a combination of two uni-directional Relationships in WebML, each being part of a *different* Entity, except for reflexive Relationships. Thus, while the *User* Entity and its contained parts, i.e., *personalRFID* and *user2roomArea*, shall be contained by the ContentModel, the *roomArea2user* Relationship shall be contained by the *RoomArea* Entity, thus resulting in two SimpleAdaptations for two different JoinPoints.

5.2.2 *Customization in the Hypertext Model*

As in the WebML approach (cf. Section 5.1.2), we now define the necessary AdaptationRules for applying the four necessary modifications of the HypertextModel (cf. Figure 10).

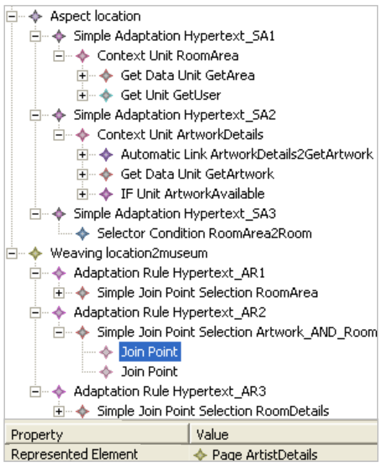
Aspect location
Simple Adaptation Hypertext_SA1
Context Unit RoomArea
Get Data Unit GetArea
Get Unit GetUser
Simple Adaptation Hypertext_SA2
Context Unit ArtworkDetails
Automatic Link ArtworkDetails2GetArtwork
Get Data Unit GetArtwork
IF Unit ArtworkAvailable
Simple Adaptation Hypertext_SA3
Selector Condition RoomArea2Room
Weaving location2museum
Adaptation Rule Hypertext_AR1
Simple Join Point Selection RoomArea
Adaptation Rule Hypertext_AR2
Simple Join Point Selection Artwork_AND_Room
Join Point
Join Point
Adaptation Rule Hypertext_AR3
Simple Join Point Selection RoomDetails
Property
Value
Represented Element
Page ArtistDetails

Figure 10: The Location-Aware Hypertext Model

① *Hypertext_AR1* uses SimpleAdaptation *Hypertext_SA1* to add as an *enhancement* a ContextUnit, which contains a GetUnit *GetUser* and a GetDataUnit *GetArea* to retrieve the users's location every time either the *RoomDetails* or the *ArtworkDetails* Pages are requested, to *ArtworkArea* as JoinPoint. The AdaptationRule, thus, realizes modification 1 (cf. Section 5.1.2).

② *Hypertext_AR2* applies SimpleAdaptation *Hypertext_SA2* to two JoinPoints, namely the *RoomDetails* and *ArtworkDetails* Pages. Thus, the rule realizes modification 2 and 4 (cf. Section 5.1.2). In particular, the *enhancement* consists of a ContextUnit, which contains a GetDataUnit *GetArtwork*, to retrieve the specific *Artwork* of the *RoomArea* the visitor is currently in. Furthermore the ContextUnit contains an IFUnit *ArtworkAvailable*, which evaluates a Condition to check whether a piece of *Artwork* is exhibited in the *RoomArea* and depending on the result activates one of the OKLinks to either the *RoomDetails* DataUnit or the *ArtworkDetail* DataUnit.

③ *Hypertext_AR3* applies SimpleAdaptation *Hypertext_SA3* to *replace* the default SelectorCondition of DataUnit *RoomDetails*, with a SelectorCondition *[RoomArea2Room]*, thus, solving modification 3 (cf. Section 5.1.2).

5.2.3 *The aspectWebML Model Editor*

For the implementation of *aspectWebML*'s metamodel, we were using *Ecore*, a MOF implementation in Java, which is provided by the Eclipse Modeling Framework (EMF)[15]. The reason for employing Ecore was mainly the wide-spread utilization of EMF

[15] http://www.eclipse.org/emf

and that currently no standardized implementation of MOF 2.0 is available. Another benefit was that having an Ecore-based metamodel, we have been able to generate a tree-based model editor for *aspectWebML* using EMF's code generation facilities.

6. CONCLUSIONS AND OUTLOOK

In this work, we proposed to use aspect-orientation as driving paradigm for capturing customization of ubiquitous web applications at the modeling level. We extended WebML, a domain-specific language designed for the model-driven development of data-intensive web applications, with concepts from the aspect-oriented modeling field according to our reference architecture for aspect-oriented modeling. Furthermore, we compared the original modeling approach of WebML with our *aspectWebML* approach by extending a *Museum* web application with customization functionality and report on our prototype modeling editor.

Future work includes, first the investigation of more elaborated join point selection mechanisms, such as OCL or Join Point Designation Diagrams, and their applicability in *aspectWebML*, second, the definition of a weaving mechanism for Aspect and Base Models in *aspectWebML*. In the long run, we intend to design a concrete syntax for *aspectWebML* and provide elaborate tool support including code generation facilities.

REFERENCES

[1] H. Baumeister, A. Knapp, N. Koch, and G. Zhang. Modelling Adaptivity with Aspects. In Proc. *5th Int. Conf. on Web Engineering*, LNCS 3579, 406-416, July 2005.

[2] S. Ceri, F. Daniel, M. Matera, F. Facca. Model-driven Development of Context-Aware Web Applications. To appear in *ACM Transactions on Internet Technology (ACM TOIT*), 7(2), May 2007.

[3] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, M. Matera. *Designing Data-Intensive Web Applications*. Morgan-Kaufmann, 2003.

[4] J. Conejero, J. Hernández, and R. Rodríguez. UML Profile Definition for Dealing with the Notification Aspect in Distributed Environments. In Proc. *6th Int. Workshop on Aspect-Oriented Modeling, AOSD'05*, Chicago, Illinois, March 2005.

[5] I. Garrigós, S. Casteleyn, J. Gómez. A Structured Approach to Personalize Websites using the OO-H Personalization Framework in Web Technologies Research and Development. In Proc. *7th Asia-Pacific Web Conference (APWeb 2005)*, Shangai, China, March-April 2005.

[6] G. Kappel, B. Pröll, W. Retschitzegger, and W. Schwinger. Customisation for Ubiquitous Web Applications - A Comparison of Approaches. *Int. Journal of Web Engineering and Technology*, 1(1), Inderscience Publishers 2003.

[7] Nora Koch. Transformations Techniques in the Model-Driven Development Process of UWE. In Proc. *2nd Model-Driven Web Engineering Workshop, ICWE'06*, Stanford Linear Accelerator Center, Palo Alto, CA, July 2006.

[8] N. Moreno, P. Fraternalli, and A. Vallecillo. A UML 2.0 Profile for WebML Modeling. In Proc. *2nd Model-Driven Web Engineering Workshop, ICWE'06*, Stanford Linear Accelerator Center, Palo Alto, CA, July 2006.

[9] Object Management Group (OMG). *Meta Object Facility (MOF) 2.0 Core Specification Version 2.0*. http://www.omg.org/docs/ptc/04-10-15.pdf, Oct. 2004.

[10] Object Management Group (OMG), *OCL Specification Version 2.0*, http://www.omg.org/docs/ptc/05-06-06.pdf, June 2005.

[11] Object Management Group (OMG). *UML Specification: Superstructure Version 2.0*. http://www.omg.org/docs /formal/05-07-04.pdf, Aug. 2005.

[12] G. Ortiz, Juan Hernández, P. Clemente, and P. A. Amaya. How to Model Aspect-Oriented Web Services. In Proc. *1st Model-driven Web Engineering Workshop, ICWE'05*, Sydney, Australia, July 2005.

[13] A. Schauerhuber, aspectUWA: Applying Aspect-Orientation to the Model-Driven Development of Ubiquitous web Applications, *Student Extravaganza: Spring School, AOSD'06*, Bonn, Germany, Available at: http://wit.tuwien.ac.at/people/schauerhuber/, 2006.

[14] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, and M. Wimmer. Towards a Common Reference Architecture for Aspect-Oriented Modeling. In Proc. *8th Int. Workshop on Aspect-Oriented Modeling, AOSD'06*, Bonn, Germany, March 2006.

[15] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, M. Wimmer, and G. Kappel. *A Survey on Aspect-Oriented Modeling Approaches* .To be submitted, July 2006.

[16] A. Schauerhuber, M. Wimmer, and E. Kapsammer. Bridging existing Web Modeling Languages to Model-Driven Engineering: A Metamodel for WebML. In Proc. *2nd Model-Driven Web Engineering Workshop, ICWE'06*, Stanford Linear Accelerator Center, Palo Alto, CA, July 2006.

[17] W. Schwinger, N. Koch. Modeling web Applications. In G. Kappel, B. Pröll, S. Reich, W. Retschitzegger (eds), *Web Engineering - Systematic Development of Web Applications*.Wiley, 2006.

[18] D. Stein, S. Hanenberg, and R. Unland. Expressing Different Conceptual Models of Join Point Selections in Aspect-Oriented Design. In Proc. *5th Int. Conf. on Aspect-Oriented Software Development*, Bonn, Germany, Marc 2006.

[19] World Wide Web Consortium (W3C). *Extensible Markup Language (XML) 1.1 Specification*. http://www.w3.org/TR/xml11/, April 2004.

[20] G. Zhang, H. Baumeister, N. Koch, A. Knapp. Aspect-Oriented Modeling of Access Control in Web Applications. In Proc. *6th Int. Workshop on Aspect Oriented Modeling, AOSD'05*, Chicago, USA, Mar. 2005.