TU VIENNA

Technische Universität Wien

Vienna University of Technology

PhD Thesis

aspectUWA

Applying Aspect-Orientation to the Model-Driven Development of Ubiquitous Web Applications

Conducted for the purpose of receiving the academic title
'Doktorin der Sozial- und Wirtschaftswissenschaften'

Supervisors

o.Univ.-Prof. Dipl.-Ing. Mag. Dr. Gerti Kappel
Institute of Software Technology and Interactive Systems
Vienna University of Technology

a.Univ.-Prof. Mag. Dr. Werner Retschitzegger
Institute of Bioinformatics
Johannes Kepler University Linz

Co-Supervisor

Mag. Dr. Wieland Schwinger, MSc
Department of Telecooperation
Johannes Kepler University Linz

Submitted to the
Vienna University of Technology
Faculty of Informatics
by

Andrea Schauerhuber

0026186
Pfarrgasse 13
3462 Absdorf

Vienna, October 22nd, 2007

To all that are dear to me.

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bei jenen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Allen voran ein besonderes Danke an meine BetreuerInnen Gerti Kappel, Werner Retschitzegger und Wieland Schwinger. Ohne ihre Führung, den Ansporn durch intensive Diskussionen & kritische Anmerkungen, und ihren Zuspruch während so mancher Durststrecke wäre diese Arbeit wohl nicht zustande gekommen.

Ich möchte mich auch besonders bei meinem Kollegen Manuel Wimmer bedanken, der mir jederzeit als aktiver und kritischer Diskussionspartner zur Verfügung stand und mit dem ich eine spannende Zusammenarbeit bei gemeinsamen Publikationen aber auch in der Lehre erfahren konnte.

Vielen Dank auch an Cornelia Tomasek und Gerhard Preisinger, die im Rahmen ihrer Diplomarbeiten mit der Implementierung des *aspectWebML Modeling Environments* zu dieser Arbeit beigetragen haben.

Für ein nettes Arbeitsklima und für ihre Freundschaft bedanke ich mich bei allen KollegInnen des Wissenschafterinnenkollegs Internettechnologien und der Business Informatics Group.

Ganz besonders möchte ich mich auch bei meinen Eltern Julius und Brigitte, und bei meinen Geschwistern Julia, Olivia, Lukas, Nora und Laura, für ihre Unterstützung und ihren Zuspruch in schwierigen Zeiten bedanken.

Nicht zuletzt ein liebevolles Dankeschön an Jürgen Flandofer für seine Geduld, sein Verständnis und für seine unglaubliche Unterstützung, die es mir erlaubt hat besonders in der letzten Phase mich voll und ganz meiner Dissertation widmen zu können.

Eidestattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Dissertation selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe. Diese Dissertation habe ich bisher weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt.

Wien, 22. Oktober 2007 ______________________________

Abstract

Ubiquitous web applications (UWA) are a new type of web applications which are accessed in various contexts, i.e., through different devices, by users with various interests, at anytime from anyplace around the globe. In this respect, *customization* functionality exploits information on this *context* of use in order to *adapt* the application's services accordingly. In web application development, customization is considered a new dimension which increases complexity by ”crosscutting” the content, hypertext, and presentation levels of a web application. Hence, from a software engineering point of view, a systematic development of UWAs on the basis of models is crucial. In model-driven engineering (MDE), models are employed ”as programs” to (semi-) automatically generate applications, which results in more efficient development processes as well as better maintainability and evolution of software. Customization functionality, however, is typically intermingled with the core functionality in a web application model, having a negative effect on understandability, reuse, maintenance and evolution. The *aspect-orientation* paradigm provides a new way of modularizing crosscutting concerns such as customization within so-called *aspects*, as well as the necessary means for *composing* the previously separated concerns in order to obtain the complete application model. There are already some web modeling approaches dealing with the ubiquitous nature of web applications, amongst them, first proposals to use aspect-orientation. Nevertheless, these approaches suffer from the following problems: First, they don't consider the crosscutting nature of customization comprehensively, but for the hypertext level, only. Second, just very basic aspect-oriented modeling (AOM) concepts are used, resulting in less powerful mechanisms for separating customization. Third, composition of concerns is not regarded for the modeling level. And fourth, model-driven development of UWAs in the sense of MDE is still limited due to missing metamodel specifications and lack of tool support.

The overall aim of this thesis is the exhaustive use of aspect-orientation as driving paradigm for comprehensively modeling customization aspect separately from all web application levels as well as providing means for composing the aspect with the web application model. Therefore, this thesis proposes the *aspectUWA* approach, which suggests a generic framework for extending existing web modeling languages with AOM concepts within the realms of MDE. In the context of this thesis, *aspectUWA* is applied to the web modeling language *WebML*, which doesn't allow to separately model customization. The major contributions of this thesis are as follows: (i) A *Conceptual Reference Model* (CRM) for AOM has been developed to form the aforementioned general framework. (ii) A *metamodel for WebML* has been semi-automatically generated from an existing DTD-based language specification in order to allow for MDE. (iii) The *aspectWebML* language has been desiged on basis of the CRM and represents WebML's port to AOM allowing for modeling customization separately as well as for composing the customization aspect with the rest of the web application model. (iv) An *initial set of guidelines* to be used for modeling customization within aspectWebML is proposed. And (v), the *aspectWebML Modeling Environment* provides the often missing tool support for modeling crosscutting concerns as well as their composition.

Kurzfassung

Ubiquitäre Web-Anwendungen (UWA) stellen speziell auf die aktuelle Situation angepasste Informationen und Dienste zur Verfügung. Diese *Anpassung* an den aktuellen *Kontext*, z.B. an verschiedene Endgeräte, Benutzer mit diversen Interessen, deren Aufenthaltsort sowie zeitliche Aspekte, wird als *Customization* Funktionalität bezeichnet. Die Entwicklung einer UWA unter Berücksichtigung von Customization, die sich quer durch die Kontent-, Hypertext- und Präsentations-Ebenen einer Web-Anwendung zieht, gestaltet sich jedoch äußerst komplex und bedarf einer systematischen Entwicklung auf Basis von Modellen im Sinne des Model-driven Engineerings (MDE). Dabei verspricht MDE effizientere Entwicklungsprozesse sowie bessere Wartbarkeit und Weiterentwicklung von Software durch die semi-automatische Generierung von Software aus Modellen. Im Modell einer Web-Anwendung ist Customization jedoch inhärent mit der Kernfunktionalität vermischt und behindert dadurch die Verständlichkeit, Wiederverwendbarkeit, Wartbarkeit und Weiterentwicklung. Die *Aspekt-Orientierung* bietet dafür einerseits neue Konzepte um *Querschnittsfunktionalität* wie Customization in sogenannten *Aspekten* zu modularisieren und andererseits die notwendigen Mechanismen für die *Integration* dieser separierten Funktionalitäten, um ein verwendbares Gesamtmodell zu erhalten. Einige wenige Web Modellierungssprachen unterstützen bereits die aspekt-orientierte Modellierung (AOM) von Customization jedoch mit folgenden Restriktionen: Erstens wird Customization nur für die Hypertext-Ebene, getrennt von der Kernfunktionalität der Web-Anwendung behandelt. Zweitens werden nur wenige grundlegende Konzepte der AOM eingesetzt, was in limitierten Mechanismen zur Trennung von Customization resultiert. Drittens, wird die spätere Integration der Aspekte in Modelle nicht unterstützt. Und viertens ist die model-getriebene Entwicklung einer UWA im Sinne von MDE aufgrund fehlender Metamodell-Spezifikationen und Werkzeugunterstützung oft nicht möglich.

Das Ziel der vorliegenden Dissertation ist der umfassende Einsatz von AOM Konzepten, um einerseits den Customization Aspekt auf allen Ebenen einer Web-Anwendung getrennt modellieren und andererseits den Aspekt auch wieder mit der Kernfunktionalität integrieren zu können. Als Lösung wird der *aspectUWA* Ansatz vorgestellt, welcher die Erweiterung existierender Web Modellierungssprachen um AOM Konzepte auf Basis eines generischen Rahmenwerks diskutiert. Die Anwendung des *aspectUWA* Ansatz wir anhand der Web Modellierungssprache *WebML* gezeigt. Dabei sind die wesentlichen Beiträge dieser Dissertation: (i) Das sogenannte *Conceptual Reference Model* als generisches Rahmenwerk für die Erweiterung von Web Modellierungsprachen um aspekt-orientierte Modellierungskonzepte. (ii) Eine *Metamodell-Spezifikation für WebML*, welche semi-automatisch aus einer existierenden Sprachspezifikation auf Basis von DTDs entwickelt wurde. (iii) Die *aspectWebML Sprache*, die auf dem konzeptuellen Referenzmodell und dem WebML Metamodell aufbauend entwickelt wurde und die Modellierung von Aspekten als auch deren Integration unterstützt. (iv) *Richtlinien* für die Modellierung von Customization mit aspectWebML. Und (v) die *aspectWebML Entwicklungsumgebung* als Werkzeugunterstützung für die Modellierung und Integration von Aspekten mit aspectWebML.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Contents

1.1 Motivation

1.1.1 Ubiquitous Web Applications

Looking back in history, the World Wide Web is characterized by different categories of web applications. At the very beginning of the World Wide Web in 1989, *document-centric* web sites were employed providing read-only access to static information. Over time, web applications, including *interactive* and *workflow-based* web applications as well as *portals*, have increased in complexity offering users support in accomplishing various tasks. According to this evolution of web applications, in this thesis, the following definition of a web application is adopted:

> A Web application is a software system based on technologies and standards of the World Wide Web Consortium (W3C) that provides Web specific resources such as content and services through a user interfaces, the Web browser [KPRR06].

In Figure 1.1, the previously mentioned categories of web applications including examples are depicted depending on their *development history* and their *degree of complexity* [KPRR06]. In the past few years, the increase of complexity of web applications has also been driven by the emergence of mobile devices as new access channels to the Internet offering new possibilities & challenges and creating a new category of web applications, called *ubiquitous web applications* (UWA). UWAs are characterized by the anytime/anywhere/anymedia paradigm, taking into account that services are not exclusively accessed through traditional desktop PCs but also through mobile devices with different capabilities, by users with various interests at anytime from anyplace around the globe. One typical example for UWAs, are tourism guides which support users during their vacation by providing them with appropriate information and services. Users of such a tourism guide should be offered points of interests, e.g., restaurants and sights, according to the current context in which the web application is accessed. For example, a list of nearby restaurants should be presented considering the user's current location. The UWA should also consider the time context,

* This research has been 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.

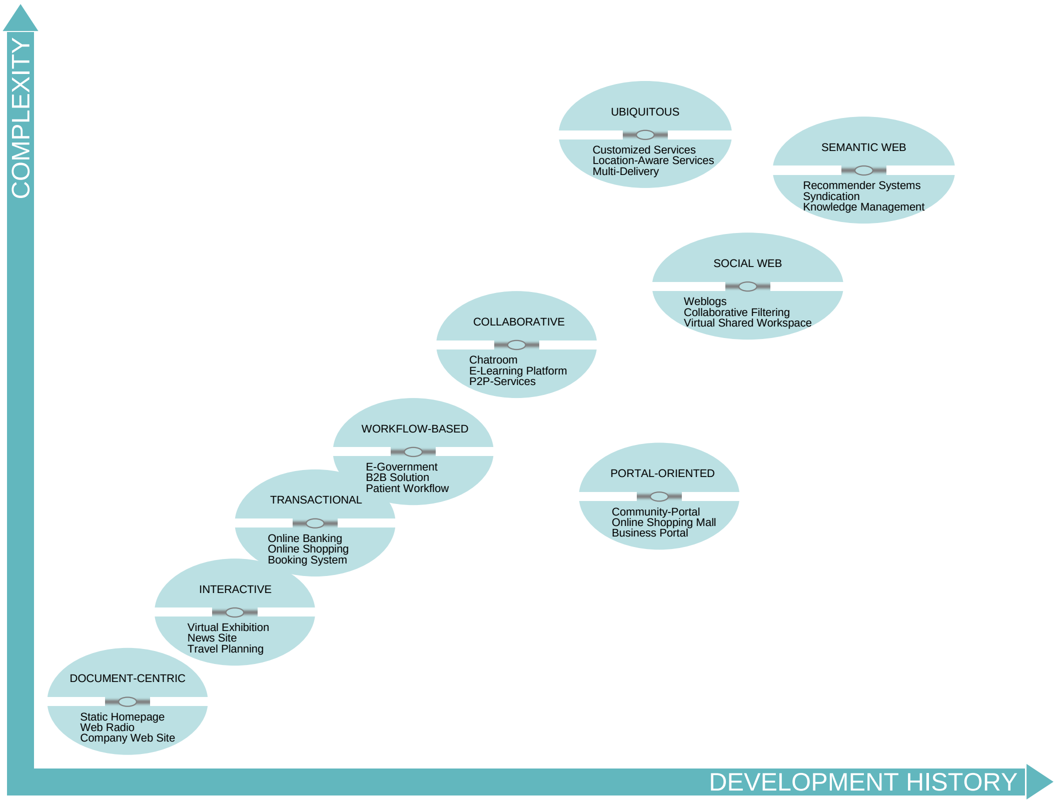


Figure 1.1: Categories of Web Applications in History [KPRR06]

by only displaying restaurants that are currently open. Furthermore, the restaurants should be ordered according to the user's preferences by first listing those that are of more interest to the user. And finally, the list of restaurants should be presented in a way that is suitable for the current device used to access the web application, such as the user's own PDA or a normal desktop PC in an Internet cafe. In any situation, the user should be able to efficiently interact with the web application which reacts accordingly in order to preserve the *semantic equivalence* of its services in the current situation of use or benefits from additional information on the current situation in order to achieve *semantic enhancement* of its services for the user [KPRS03]. Consequently, knowing the *context* in which a web application's services are requested, e.g., information on the user, the location, the device, the network, as well as the time, is the prerequisite for a web application to react with appropriate *adaptations* of its services, e.g., filtering information according to the user's interests and selecting images according to the device's display resolution. In this thesis, we adopt the term *customization* [Sch01] to denote the mapping between the required adaptation of a web application's services towards its context of use.

Considering the notion of customization from a historical point of view, *personalization* and *mobile computing* represent two areas of research posing a major influence on customization [KPRS03]. It has to be emphasized, that customization is considered more comprehensive than traditional personalization which rather focuses on the user context and usage data [Kob01], only. Beyond this, customization as well encompasses context properties that have already been considered in various sub-branches of mobile computing, including location-based services [WS01], multi-channel delivery [EVP01], and network adaptation [BFK$^+$00].

1.1.2 Model-driven Development of Ubiquitous Web Applications

When considering the development of UWAs, the original "one-size-fits-all" approach in the development of web applications needs to be abandoned for the sake of UWAs' objective of communicating the right information and services at the right moment. As a consequence, in the following sub-sections, the different dimensions relevant in the realms of developing UWAs are presented. In this respect, the role of models when developing UWAs as well as the need to deal with customization in models is discussed. Finally, the motivation of this thesis is concluded with a discussion of the current challenges in customization modeling.

1.1.2.1 Dimensions of Ubiquitous Web Application Development

In the realms of general web application development, three different orthogonal dimensions can be distinguished [SK06] (cf. Figure 1.2). The *levels* dimension breaks down a web application into three different *levels*: The content level's purpose is similar to that of a non-web application, i.e., capturing the underlying information and application logic. Still, the hypertext level is particular to web applications which is due to the document-centric character of web applications and the non-linearity in using them. The hypertext level comprises all kinds of navigation possibilities on the basis of the content level. Finally, the presentation level represents the user interface or page layout. At each of the three levels, structure and behavior are regarded as is indicated by the second dimension, i.e., *features*. Analogous to the software engineering domain, web applications are built during several *phases*, though there is not yet a consensus on a certain development process.

In addition to these three general dimensions, UWAs demand for a further dimension. *Customization* as a fourth orthogonal dimension influences all of the other three dimensions, i.e., the structure and behavior at the content, hypertext, and presentation levels and should be taken into account during all phases of the development process.

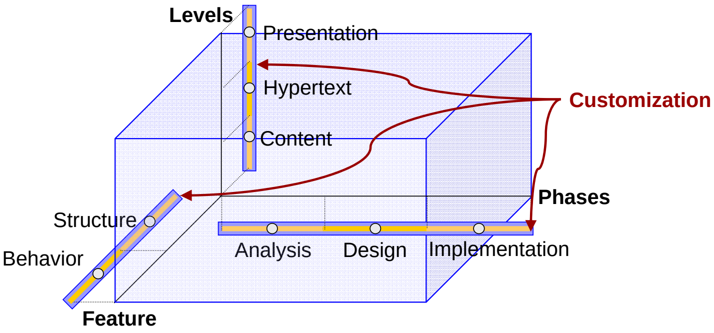


Figure 1.2: Dimensions of Web Application Development [Sch01]

Considering today's UWAs and their degree of complexity through the introduction of customization, from a software engineering point of view, the importance of their model-driven development following a sound development process as well as sound techniques is crucial.

1.1.2.2 Role of Models in Model-Driven Web Engineering

In software engineering, Model-Driven Engineering (MDE) [Sch06b] has received considerable attention during the last years and is well on its way to becoming a promising paradigm. In MDE, models replace code as the primary artifacts in software development processes. This means that in MDE, the purpose of models goes beyond using *models as sketches* for merely communicating ideas and alternatives in the application as well as using *model as blueprints* where all design decisions are already laid out and implementation is straightforward. Instead, MDE propagates the employment of *models as programs* allowing to automatically generate the final application for several programming platforms. More specifically, developers are forced to focus on modeling the problem domain and not on programming one possible platform-specific solution. This abstraction from specific programming platforms decouples the business functionality from the technology-specific code, while the definition of model transformations allows generating several platform-specific implementations. This kind of automation in software development promises more efficient development of software, higher quality of software products as well as better maintainability and evolution of software. In this respect, the key prerequisite for MDE is the employment of a modeling language definition standard such as the Object Management Group's (OMG) Meta Object Facility (MOF) [OMG04], allowing for standardized storage (e.g., Eclipse Modeling Framework - EMF [BSM+04]), exchange (e.g., XML Metadata Interchange Format - XMI [OMG05b]), and transformation of models (e.g., Query/View/Transformation - QVT [OMG05a]).

Recently, MDE's pendant in the web engineering community, i.e., Model-Driven Web Engineering (MDWE) [KRV05] gains more and more attention as well. Considering MDE in the area of web application development in general, various modeling approaches have been proposed in the past [CFB+03], [FHB06], [GCP01], [RS06], [PFPA06], [Koc07], [CPT06], [ISB95], [GPS93], [BM02], [Con02], [MG06] each of them aiming at counteracting a technology-driven and ad hoc development of web applications. These web modeling approaches originally have emerged as proprietary languages rather focused on notational aspects using models as sketches. Moreover, some of them already provide techniques and tools for modeling web applications in a platform-independent way. If existent, the tool's code generation facilities mostly support only one specific platform, however, yielding transformations from models directly to code. In the past few years, these approaches have started to more and more support model-driven development of web applications in the sense of MDE by providing language specifications in terms of metamodels [KK03], [SWK+07], [BM02] or UML Profiles [MFV07] and by considering model transformations [Koc07]. Considering UWAs and the model-driven development thereof in particular, modeling customization needs specific attention.

1.1.2.3 Customization Modeling

Customization specializes a web application called *core web application* towards the situation of use [Sch01], as indicated in Figure 1.3. For capturing customization at modeling level this means, the description of different situations of use have to be captured in the web application model, e.g., in terms of a so-called context model. As proposed in [Sch01], such a context model captures information on the context in which a web application is accessed over time, including information on the user, device, time, location, and network. On the other hand, the changes to a web application in terms of adaptations need to be described as well. Finally, a model for a UWA needs to capture which situation requires which adaptation, i.e., the *mapping* between a certain *context* and the required *adaptation*. Such a mapping is often modeled using as a modeling formalism *Event-Condition Action* rules [CDF06], [GCG05], [Sch01] originally stemming from the area of active database systems [PD99]. This way, customization modeling allows incorporating variability into a web application.

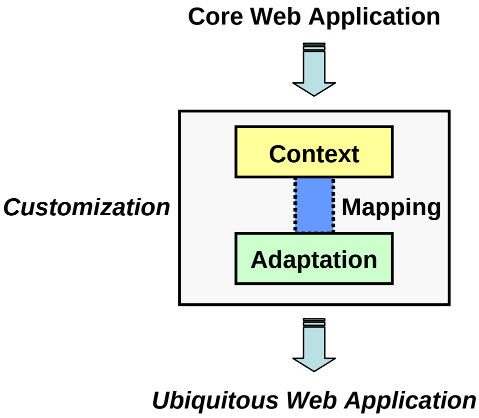


Figure 1.3: Customization Issues [Sch01]

Concerning appropriate means for modeling customization, we find that there are already some web modeling approaches [CFB$^+$03], [FHB06], [TL98], [RS06], [KK02a], [GCP01], [PFPA06] dealing with the ubiquitous nature of web applications. They often focus on certain facets of customization such as personalization (by exploiting the context information of the user), multi-delivery (by considering the user's device), or location-awareness (by regarding the user's current location). Context information is typically captured at the content level. Some approaches even propose ways to indicated what parts of the content level represent contextual information. Furthermore, some have introduced new modeling concepts allowing developers to specify adaptations at one or several web application's levels.

Still, means for customization modeling currently neither cover all relevant context factors in an explicit, self-contained, and extensible way, e.g., within a dedicated context model, nor allow for a wide spectrum of extensible adaptation operations. Furthermore, the provided customization mechanisms frequently do not allow dealing with all different parts of a web application in terms of its content, hypertext, and presentation levels as well as their structural and behavioral features.

While the expressiveness of a web modeling language's customization mechanism is undoubtedly important, this work's focus will be on the equally pressing challenge of *customization's crosscutting nature*. As pointed out before, customization is considered a new orthogonal dimension in web application development, influencing all web application levels. With respect to modeling customization this means that customization needs to be considered when modeling the content, hypertext, and presentation levels in a web application model. Disregarding its crosscutting nature, in current web modeling approaches, customization is inherently tangled with and scattered across all levels of a web application model:

- For example, a modeler might want to specify that for some pages of the web application additional information might be provided to the user, e.g., if the location of the user is available, the user might be presented with nearby points of interests. This customization scenario *crosscuts* all levels of a the web application: In order to specify this kind of customization in a model, the developer will need to model the necessary context information, i.e., the location of the user, at the content level. Furthermore, at the hypertext level, the necessary adaptations to filter the points of interest according to the user's location need to be modeled as well. Finally, at presentation level, the developer might specify with appropriate adaptations to particularly highlight the three nearest points of interests by presenting them in a special way (e.g., by using a larger font or small pictures). This way, customization takes up a large part in a web application model and requires specific attention.

- More specifically, customization is *tangled* with the structure and behavior for each web application level, i.e., content, hypertext and presentation. As an example, contextual information, e.g., on the user, the device, and the location, is often modeled at the content level. It is intermingled with the application data instead of being modeled separately within a dedicated context model. Likewise, adaptations are typically intermingled with the rest of the web application model. As a consequence, it is not always clear which modeling elements in a model are part of a customization scenario and which contribute to the core functionality of a web application. In this respect, this intermingled representation of customization contributes to the complexity of web application models and hampers their readability as well as their understandability. Furthermore, it is not possible to reuse parts of the customization functionality within other web application models. For example, a separate context model could be reused in web applications requiring similar context information.
- Furthermore, customization is *scattered* across the structure and behavior of each web application level. Similar to the previous customization scenario for presenting nearby points of interest at several pages to the user, a modeler might want to restrict the display of pictures, e.g., on pages displaying more than one picture. On all these pages the pictures shall be resized to smaller ones. For both scenarios this means that, certain adaptations need to be considered in several places of the model. They are redundantly modeled in the web application model requiring the modeler to visit all these places in case a change to the adaptation is necessary. In this respect, customization has a negative effect on the web application model's maintainability and evolution.

In the sense of *Separation of Concerns* (SoC) [Dij76], [Par72], the fact that current web modeling approaches currently do not provide means to fully separate the customization concern from the rest of the web application model has a negative effect on the models' understandability and leads to inefficient development processes, high maintenance overheads as well as a low potential for reuse of context and adaptation specifications. In this respect, *aspect-orientation* seems to be a promising paradigm providing a new way of *separating* crosscutting concerns, such as customization, from non-crosscutting ones, meaning the core functionality of a web application in this case. In the following, the ideas of the aspect-orientation paradigm and their relevance for modeling customization shall be explained.

1.2 Addressing Crosscutting Concerns With Aspect-Orientation

Aspect-Oriented Software Development (AOSD)[1] is an equally young research field as the area of web modeling languages. It is an emerging area of research that aims at promoting *Advanced Separation of Concerns* (ASoC) throughout the whole software development lifecycle. Aspect-orientation originally emerged at the programming level [KLM+97] and has proven to be a promising mechanism providing a new way of modularization by clearly separating crosscutting concerns, so-called *aspects*, from non-crosscutting ones (cf. Chapter 3). Still, being able to specify the different concerns of an application within separate modules, appropriate means for composing them to obtain a working system are necessary as well. Such a mechanism is often called a *weaver* which is responsible for *weaving* the aspects ”back” into the application.

Past experience has shown that the evolution of a new programming style often propagates the new concepts from the programming level to earlier phases in the software development lifecycle. Meanwhile, aspect-orientation also stretches over earlier development phases. For the design phase, there already exist several general-purpose approaches to aspect-oriented modeling (AOM) of which some representatives have been investigated in Chapter 3.

Considering UWAs, customization can be seen as a crosscutting concern in the sense of aspect-orientation. The left-hand side of Figure 1.4 shows how, or rather *where*, customization affects a web application model, i.e., typically the structural and behavioral parts of all the content, hypertext, and presentation levels.

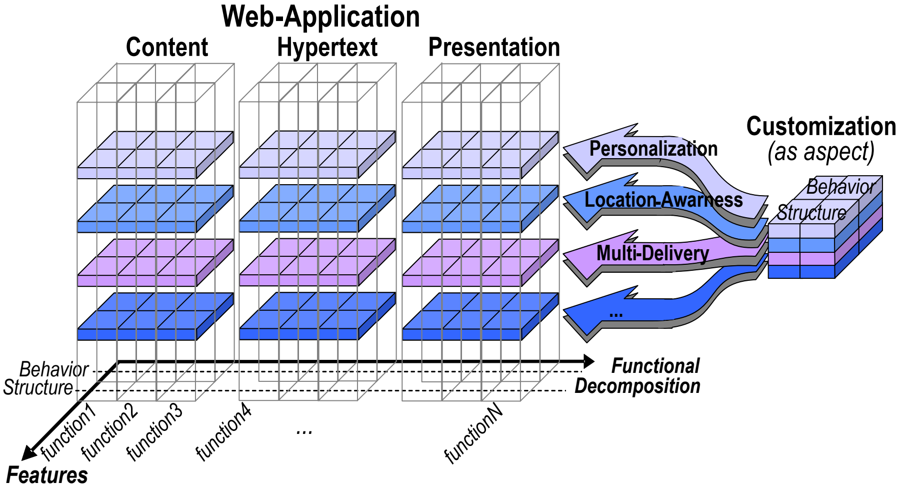


Figure 1.4: Customization as a Crosscutting Concern [Sch06a]

Consequently, customization can be encapsulated within one or more *aspects* as is illustrated on the right-hand side of Figure 1.4. Being able to modularize customization functionality at the modeling level within aspects implies the typical positive effects attributed to the SoC principle, e.g., reduction of complexity, higher maintainability due to better locality of change, as well as reusability. Similar to the programming level, to obtain a ”working model”, e.g., one that can be fed to code generation facilities, the previously separated concerns need to be integrated into a composed model. The left-hand side of Figure 1.4 therefore also indicates the composed model

[1] www.aosd.net

of a web application, where the customization aspect(s) again is (are) tangled with and scattered across the rest of the web application model.

Currently, two web modeling approaches [BKKZ05], [CWH07] have already proposed the use of concepts from the aspect-orientation paradigm to separately capture customization at modeling level. Still, these approaches suffer from the following problems[2]:

- They do not consider the crosscutting nature of customization comprehensively for all levels of the web application and thus, do not provide the means for fully separating the customization concern. More specifically, they allow for separately modeling adaptations from the hypertext level, only.
- They have been extended with a minimal set of aspect-oriented modeling (AOM) concepts, only, i.e., *aspect*, *pointcut*, *advice*. This results in less powerful AOM languages for separately modeling crosscutting concerns such as customization.
- They do not consider the composition of concerns at the modeling level which allows for exploiting existing tool support available for composed models, e.g., code generation facilities. Concerning related approaches, the composition semantics are either not specified at all [BKKZ05], or not considered at the modeling level but for a specific programming platform [CWH07].

In contrast to these approaches, the overall aim of this thesis is the exhaustive use of aspect-orientation as driving paradigm for comprehensively modeling the customization aspect separately from all levels as well as providing means for composing the aspect with the web application model.

Furthermore, web modeling languages in general often lack a proper MDE foundation and tool support:

- The majority of existing web modeling approaches are not yet defined using a language definition standard in the sense of MDE, although first proposals for a transition to the MDE paradigm in web engineering have been made as has already been pointed out before. As a consequence, MDE techniques and tools cannot be deployed for such languages which prevents exploiting the full potential of MDE in terms of standardized storage, exchange, and transformation of models as well as profiting from MDE's benefits.
- There is also a lack in appropriate tool support for modeling web applications in general and for modeling customization in particular. Currently, four web modeling approaches are accompanied with tool support which is either publicly available [KK02a], [GCP01] or can be obtained under a commercial license [CFB$^+$03], [FHB06]. Besides this general support for modeling web applications, concerning UWAs, some approaches that have developed concepts for modeling customization have also reported on corresponding tool support [CDMF07], [GCG07]. Nevertheless, in all cases, this tool support has not yet left the status of prototypes and has not been made publicly available.

Concerning these problems, in this thesis, metamodels shall be employed as a language specification formalism using current MDE technologies, developed under the hood of the Eclipse Modeling Framework (EMF) [BSM$^+$04]. In this respect, EMF's code generation facilities shall be exploited for tool support purposes.

[2] For a detailed discussion see Chapter 8.

1.3 The aspectUWA Solution at a Glance

1.3.1 Goals of This Thesis

This thesis proposes *aspectUWA - Applying* ***Aspect****-Orientation to the Model-Driven Development of* ***U****biquitous* ***W****eb* ***A****pplications*. The *aspectUWA* approach aims at the exhaustive use of *aspect-orientation* as driving paradigm for comprehensively capturing *customization* separately from all levels of a web application model. In the sense of the aspect-oriented paradigm, the *aspectUWA* approach also proposes to provide means for composing the customization aspect with the web application model. Furthermore, in order to benefit from MDE's advantages, the *aspectUWA* approach advocates its realization within the realms of MDE.

aspectUWA suggests the general idea of extending any existing web modeling language with concepts from the aspect-orientation paradigm in order to separate customization functionality from the rest of the web application model. To do so, a kind of framework called the *Conceptual Reference Model* (CRM) for AOM is developed, allowing the extension of any web modeling language with AOM concepts through a set of extension points. To demonstrate the applicability, the *aspectUWA* idea shall be applied to a representative web modeling language that supports customization modeling but does not allow modeling customization separately. In this respect, the web modeling language *WebML* [CFB$^+$03] is one of the most prominent representatives of current web modeling languages, being already supported by the commercial tool *WebRatio*[3], and recently has been extended with concepts for modeling customization [CDMF07]. In order to better support the development of UWAs within the *WebML* approach, a further goal of this thesis is the design of the *aspectWebML* web modeling language as a result of bridging *WebML* to aspect-orientation according to the CRM provided by *aspectUWA*. In particular, this also includes the specification of the language's composition semantics required to integrate previously separated concerns into a composed model. Furthermore, the *aspectWebML* language shall be accompanied by a tool allowing to model UWAs in terms of several concerns as well as their composition. On the basis of a case study, the original *WebML* approach to modeling UWAs shall be compared to the *aspectWebML* approach in order to point out *aspectWebML*'s strengths and weaknesses.

1.3.2 Methodology - A Roadmap to aspectWebML

In order to achieve these goals, the following methodology will be employed, which also resembles the subsequent structure of the thesis:

Investigation of the State-of-the-Art in Modeling Ubiquitous Web Applications. As already stated before, current web modeling languages are limited with respect to developing UWAs. To provide a proper foundation for this thesis, the state-of-the-art in modeling customization with current web modeling approaches is investigated and the approaches' strengths and weaknesses are identified. Moreover, in the context of this thesis, the survey shall provide the foundation for selecting a web modeling language to be extended with AOM concepts in order to capture customization functionality separately from the rest of a web application model. In this respect, the selected web modeling language shall provide proper support for modeling customization through appropriate means of modeling but fail to model customization separate from the rest of a web application model (cf. Chapter 2). As already

[3] www.webratio.com

mentioned before, the WebML approach matches these requirements and has been selected accordingly in order to be bridged to AOM.

Design of a Conceptual Reference Model for Aspect-Oriented Modeling. Before a web modeling language can be extended with AOM concepts, the basic ingredients of AOM need to be identified. The domain of AOM is still a young research field, however. Having diverse origins, current general-purpose AOM languages differ in terminology and concepts, including different composition mechanisms, i.e., the way previously separated concerns are to be integrated into a composed model. In order to tackle the problem of different terminologies and a broad variety of aspect-oriented concepts, the aspectUWA approach proposes the *Conceptual Reference Model for Aspect-Oriented Modeling* (CRM). The CRM will be defined in terms of a UML class diagram and identify the basic ingredients of AOM, abstracted from specific modeling languages as well as from specific composition mechanisms. In this respect, it represents the basis of the *aspectUWA* approach by capturing the important AOM concepts, their interrelationships, and even more importantly, their relationships to an arbitrary modeling language representing the extension points of the framework (cf. Chapter 3). Thus, the CRM serves as an important input to the design of new AOM languages or for the extension of existing (domain-specific) modeling languages with concepts of the aspect-oriented paradigm like it is done for *WebML* in this thesis.

Design of a Metamodel for WebML. The *aspectUWA* approach advocates its realization within the realms of MDE, in order to profit from MDE's proclaimed advantages. As a consequence, a language specification of the web modeling language to be bridged to AOM needs to be available in terms of a metamodel based on the Meta Object Facility (MOF) [OMG04]. The WebML language, however, has been partly specified in terms of XML Document Type Definitions (DTD) [W3C06] and partly hard-coded within the tool accompanying the language. Consequently, in order to support model-driven development of web applications in the sense of MDE, a MOF-based metamodel needs to be designed for the *WebML* language. In this respect, the existing DTD-based language specification as well as constraints hard-coded within the language's modeling tool *WebRatio* shall be reused within a semi-automatic process for metamodel generation from DTDs (cf. Chapter 4).

Design of the aspectWebML Language. The artifacts produced in the previous two steps serve as input to this step, i.e., the extension of the *WebML* language with concepts from the aspect-orientation paradigm. More specifically, the CRM will serve as a blueprint for designing *aspectWebML* on top of the *WebML* language. The design of the *aspectWebML* language will not only include the language's metamodel but also a proposal for a concrete modeling notation for the aspect-oriented concepts introduced in *aspectWebML* (cf. Chapter 5).

Specification of the aspectWebML Composition Semantics. In order to be able to compose the different concerns that can be modeled separately from each other on the basis of *aspectWebML*'s modeling concepts, the composition semantics of the *aspectWebML* language need to be specified. The composition semantics will be explained by means of small and illustrative modeling examples and an overview of the implementation of the *aspectWebML* composition algorithm will be given in terms of UML activity diagrams (cf. Chapter 5).

Evaluation of the Proposed Solution Through a Case Study. For demonstrating *aspectWebML*'s applicability and advantages, the original *WebML* approach to modeling UWAs will be compared with the *aspectWebML* approach in a case study. In the case study, the *Context-Aware*

Museum web application previously defined for illustrating *WebML*'s customization modeling concepts [CDMF07] will serve as the example UWA. In order to compare the approaches in the context of a UWA with complex customization functionality, the example will be considerably extended in the case study (cf. Chapter 6).

Development of Modeling and Composition Tool Support. The *aspectWebML Modeling Environment* will be developed and provide modelers with tool support for modeling *aspectWebML* models and compose separate concerns defined therein. The *aspectWebML Modeling Environment* will be built on top of current MDE technologies, developed under the hood of the Eclipse Modeling Framework (EMF) [BSM$^+$04] (cf. Chapter 7).

1.4 Contributions of This Thesis

Having elaborated on the steps required to achieve the proclaimed goals, this thesis' contributions can be summarized as follows:

The Conceptual Reference Model for Aspect-Oriented Modeling. As already mentioned before, the problem of different terminologies and a broad variety of aspect-oriented concepts is reflected by a still missing generally acknowledged understanding of AOM concepts. In this respect, the CRM represents a contribution to the AOM community by providing a taxonomy as well as a conceptual model for AOM. More specifically, the CRM abstracts from different composition mechanisms known in literature while providing their refinement in specialized packages of the CRM as well (cf. Chapter 3). In this thesis the CRM's applicability has been shown in two ways:

- The *aspectWebML* web modeling language has been designed on top of the *WebML* language using the CRM as a blueprint (cf. Chapter 5).
- For the structured evaluation of a set of AOM approaches, a catalogue of criteria has been derived from the CRM (cf. Chapter 3).

The WebML Metamodel. The design of a MOF-based *WebML* metamodel represents an important prerequisite and thus, an initial step towards the employment MDE techniques within the *WebML* approach. It also enables interoperability with other MDE tools and is another step towards a common reference metamodel for web modeling languages (cf. Chapter 4).

The DTD2MOF Framework. The existing DTD-based language specification of WebML as well as constraints hard-coded within the language's modeling tool have been reused within a semi-automatic process for metamodel generation from DTDs. The DTD2MOF framework is a generic framework for semi-automatically generating MOF-based metamodels from arbitrary DTD-based language specifications. Though its design has been motivated by the necessity of developing a metamodel for the DTD-based WebML language, it has been designed for generality. The work on the DTD2MOF framework includes the elaboration on the *deficiencies of DTDs* when used as means for specifying a modeling language instead of using metamodels. Moreover, a set of *rules and heuristics* for transforming arbitrary DTDs into MOF-based metamodels is provided and appropriate *tool support* for a semi-automatic transformation process from DTD to MOF has been developed. In this respect, the transformation approach enables the "visual" representation of any DTD-based language in terms of MOF-based metamodels and thus, enhances their understandability (cf. Chapter 4).

The aspectWebML Web Modeling Language. The specification of the *aspectWebML* language in terms of a *MOF-based metamodel* is the major contribution of this thesis. It has been designed in order to improve the development of UWAs by allowing to model customization separately from the remaining concerns of a web application (cf. Chapter 5). The design of the aspectWebML language also includes a proposal for a *concrete modeling notation* for the aspect-oriented concepts introduced in aspectWebML. Furthermore, in order to be able to compose the different concerns that can be modeled separately from each other on the basis of aspectWebML's modeling concepts, the *composition semantics* of aspectWebML have been specified by means of small and illustrative modeling examples. Moreover, an overview of the implementation of the aspectWebML composition algorithm, which has been implemented in Java and is integrated within the modeling environment for aspectWebML, has been given in terms of UML activity diagrams.

An Initial Set of Guidelines for Modeling Customization in aspectWebML. On the basis of the case study used to compare *WebML* and *aspectWebML*, modelers are provided with an initial set of guidelines to be used in aspect-oriented modeling of customization with the *aspectWebML* approach. Moreover, first extensions to the original WebML development process in order to better support the development of UWAs have been proposed (cf. Chapter 6).

The aspectWebML Modeling Environment. Tool support is typically rare, both in the web modeling field as well as in the AOM field. Thus, initial modeling support is provided on the basis of a tree-based editor for which a preliminary version has been automatically generated using EMF's code generation facilities. This preliminary tool support has been extended with special features for a better support of developing UWAs with aspectWebML (cf. Chapter 7):

- The composition semantics have been implemented in Java and have been integrated within the editor. In this respect, *aspectWebML* is one of those few languages that provides modelers not only with means for decomposing concerns but also with means for their composition.
- Inspired by the *AspectJ Development Tools*[4] a so-called *Cross Reference View* has been implemented and is intended to ease aspect-oriented modeling by visualizing interrelationships, e.g., between aspects and the web application model.
- Though based on a tree-editor, the aspectWebML Modeling Environment provides first modeling support for *WebML*'s customization modeling concepts, which are not yet considered within the *WebRatio* tool.

1.5 Thesis Outline

As already mentioned before, the structure of this thesis follows the methodology and contribtuions described previously. Following, an overview on the structure of this thesis is given.

Chapter 2: On Model-driven Development of Ubiquitous Web Applications
This chapter presents a survey of seven web modeling approaches supporting the model-driven development of UWAs as well as summarizes the lessons learned with respect to the approaches'

[4] www.eclipse.org/ajdt/

strengths and weaknesses. Furthermore, it serves as the foundation for selecting the *WebML* approach as the web modeling language which shall be extended with AOM concepts following the *aspectUWA* approach.

Chapter 3: State-of-the-art in Aspect-oriented Modeling
In this chapter, the *Conceptual Reference Model for AOM* is proposed. Furthermore, a survey of eight representative AOM approaches is presented. Again, the lessons learned report on the state-of-the-art in AOM.

Chapter 4: Bridging WebML to Model-driven Engineering
The *WebML* metamodel is proposed in this chapter. Moreover, the generic DTD2MOF framework is presented, which has been designed to semi-automatically generate the MOF-based metamodel for WebML from the existing DTD-based language specification.

Chapter 5: aspectWebML - Applying aspectUWA to WebML
This chapter proposes the *aspectWebML* metamodel as a result of applying the CRM of *aspectUWA* to the *WebML* metamodel. In addition, a first proposal for a modeling notation for the aspect-oriented concepts of the aspectWebML language is provided. Furthermore, the composition semantics of the language are detailed.

Chapter 6: The Context-Aware Museum Case Study
In this chapter, the focus is on a case study used to compare the *WebML* approach with the *aspectWebML* approach on the basis of a modeling example. This involves complex customization functionality in terms of eight customization scenarios and provides initial guidelines for modeling customization aspect(s) with *aspectWebML*.

Chapter 7: The aspectWebML Modeling Environment
The chapter introduces the initial tool support accompanying the *aspectWebML* language in terms of a tree-based modeling editor that also implements the language's composition semantics and allows for composing separately modeled concerns of a web application.

Chapter 8: Related Work to aspectWebML
This chapter is dedicated to a discussion of related work, i.e., on three web modeling approaches that have recently started to incorporate concepts from the aspect-orientation paradigm.

Chapter 9: Conclusion
Finally, in this last chapter, the contributions of the thesis are summarized and a discussion of current limitations to be addressed in future research is provided.

2 On Model-driven Development of Ubiquitous Web Applications

Contents

Today's web applications are full-fledged, complex software systems for which a methodologically sound engineering approach is crucial. Web engineering has emerged as an independent branch of software engineering and ”comprises the use of systematic and quantifiable approaches in order to accomplish the specification, implementation, operation, and maintenance of high quality web applications” [KPRR06]. During the past 10 years, academia has provided various web modeling approaches, each aiming at counteracting a technology-driven and ad hoc development of web applications. These web modeling approaches originally have emerged as proprietary languages rather focused on notational aspects. Some of them are supported with a modeling tool and possibly code generation facilities. As the types of web applications have evolved over time so have the web modeling approaches have come up with appropriate concepts for them. Thus, increasingly more web modeling approaches are supporting the development of UWAs by providing new modeling concepts that capture customization functionality. Additional, the rise of Model-Driven Engineering (MDE) already has had impact on current web modeling languages. Consequently, at the same time, some of the approaches are supported with a modeling tool and possibly code generation facilities and have also aimed at providing for a model-driven development in the sense of MDE, i.e., on the basis of MDE techniques and technologies including metamodels and model transformations.

Based on previous work [SSW$^+$07], this chapter is dedicated to present the state-of-the-art in model-driven development of UWAs and to throw some light on the current limitations with respect to developing web applications in the sense of MDE, as well as the identified problems of missing tool support, limited customization mechanisms, and disregarded crosscutting nature of customization. Furthermore, the results of the evaluation shall serve as a foundation for choosing a web modeling language to be extended with aspect-oriented modeling concepts according to the *aspectUWA* approach. More specifically, an in-depth comparison of seven web modeling approaches currently supporting the development of UWAs has been provided. In Section 2.1, the evaluation set-up is presented, i.e., the selection of web modeling approaches as well as a detailed and well-defined catalogue of evaluation criteria used for the structured comparison of the approaches in Section 2.2. Moreover, the actual evaluation by means of a catalogue of criteria is supported by a modeling example, i.e., a tourism information web application, used to provide an initial insight into the concepts for modeling customization of each approach as well as to facilitate their comparability. A set of five customization scenarios has been defined, to be tested with each web modeling approach. This per-approach evaluation is furthermore complemented with an extensive report on lessons learned (cf. Section 2.3), pointing out the approaches' strengths and shortcomings. Thereafter, in Section 2.4 existing related work and the contributions of this survey are discussed. Finally, in Section 2.5, the chapter is closed with a brief summary.

2.1 Evaluation Set-Up

This survey's goal is primarily to provide a literature study of existing web modeling languages having as specific focus the model-driven development of UWAs. In particular, the focus is on design level concepts for modeling customization as well as on tool support for the model-driven development of UWAs. As a consequence of this focus, there will be no report on other possible strengths of the investigated approaches that lie beyond customization, e.g., support for workflow-based web applications.

In the following, this section is dedicated to providing a motivation of the selection of web modeling approaches to be investigated, an introduction of the criteria catalogue to be used as well as a presentation of the modeling example supporting the evaluation of each approach.

2.1.1 Selection of Approaches

With respect to the selection of evaluation candidates, in literature, several well-established and well-published web modeling languages having different origins and pursuing different goals can be found. In Schwinger et al. [SK06], a categorization of fourteen approaches into data-oriented, hypertext-oriented, object-oriented, and software-oriented web modeling approaches has been proposed (cf. Figure 2.1).

Still, since focused on customization modeling, in this survey, only web modeling approaches that also provide concepts for customization modeling will be considered. In the following, seven different out of the fourteen approaches will be investigated. From the data-oriented category, the WebML [CFB$^+$03] and the Hera [FHB06] approaches, from the hypertext-oriented category, the WSDM [TL98] approach, and from the object-oriented category the OOHDM [RS06], the UWE [KK02a], the OO-H [GCP01], and the OOWS [PFPA06] approaches are evaluated. All of them can be considered to be well-established since published in about 25 up to 50 publications including reviewed papers, books, and manuals.

2.1.2 Catalogue of Evaluation Criteria

In the following, a catalogue of criteria for the structured evaluation of web modeling approaches is proposed, having a particular focus on criteria for evaluating the support for customization modeling and for model-driven development. The criteria are, on the one hand, the result of a top-down approach considering the four dimensions of web application development (cf. Chapter 1.1, Figure 1.2) and on the other hand the result of a bottom-up approach taking into account interesting issues from related surveys as well as from previous work [KPR$^+$01], [KPRS03], [SK06].

The overall emphasis of the catalog of criteria is on *functional* criteria. Since this survey is based on a literature study, the inclusion of non-functional criteria in terms of several "-ilities" such as

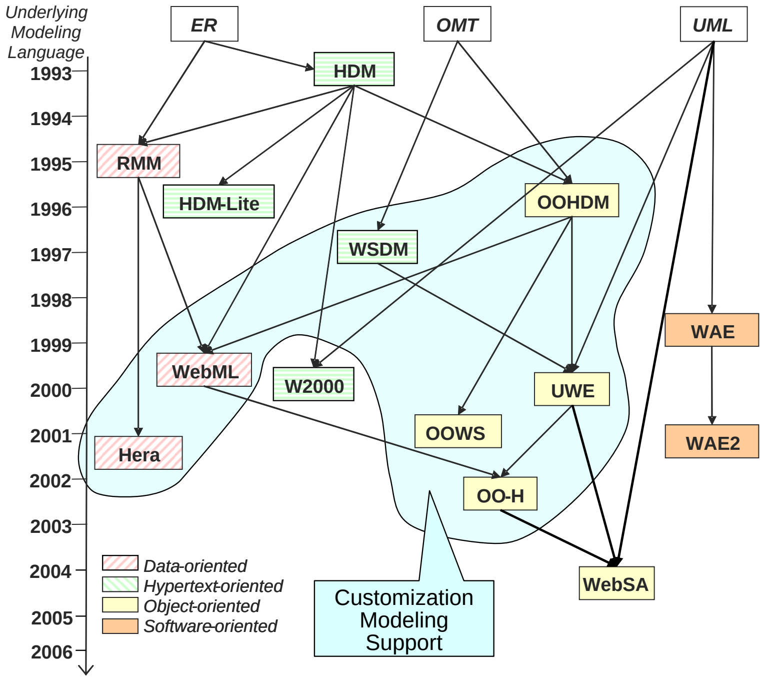


Figure 2.1: Overview on Web Modeling Approaches

evolvability, scalability, traceability, reusability, understandability, maintainability, or flexibility is not considered within the scope of this evaluation. Still, this survey paves the ground for a later evaluation in real-world projects allowing to investigate the web modeling approaches with respect to the aforementioned ”-ilities”. Nevertheless, this would also raise currently unaddressed questions associated with empirical evaluations in web engineering, e.g., how to get an unbiased set-up for the evaluation including control groups.

Aiming at a solid definition of criteria, for each criterion its *name* and a *definition* along with the appropriate *measures* are given. The *abbreviation* of the criterion allows for referencing it during evaluation of the approaches in Section 2.2.

Furthermore, these criteria are grouped into five categories with three out of them being inferred from corresponding functional requirements depicted in Figure 1.2 and two additional categories, one providing general criteria on the *Maturity* of an approach and the other one providing criteria related to the *Tool Support* of an approach. The catalogue of criteria is presented along with its categories in the following.

2.1.2.1 Maturity

As already indicated by the category’s name, the *Maturity* criteria basically serve for providing general information on the approaches’ level of maturity.

Topicality (G.T). This criterion provides for each approach the *year of introduction* as well as when the *most recent piece of work* has been published in order to indicate whether the approach is still under development or not.

Modeling Examples (G.ME). One indication for the maturity of an approach is the *number of different modeling examples* discussed. Admittedly, besides evaluating the number of existing examples, their depth would also be of interest. Such a depth measure could be composed of the number of modeling concepts used, i.e., the number of content classes, nodes, links, etc. Still, this is not feasible, since often only parts of the examples are shown in available literature, or the examples have been simplified for readability purposes.

Application in Real-World Projects (G.A). Another indication for a high level of maturity of an approach is its employment in designing real-world applications. This criterion evaluates whether real-world applications *exist* or *do not exist*.

2.1.2.2 Web Modeling

The *Web Modeling* category covers criteria for evaluating the dimensions of web application development, namely *levels*, *features*, and *phases* (cf. Figure 1.2 in Chapter 1).

Web Application Levels (W.L). This criterion indicates which web application levels (i.e., content, hypertext, and presentation) are considered by an approach and which *formalisms / types of diagrams* are employed.

Interfaces (W.I). How the interrelationships between the web application levels are modeled, is indicated by the *Interfaces* criterion. In case the interface specification is defined separately from the levels, this criterion names the mechanism used for content-hypertext and hypertext-presentation interfaces in terms of *graphical notations*, *natural language*, or *text-based query languages*. If an interface specification is not defined separately but as part of one of the two levels in question, this criterion additionally evaluates to *intermingled*.

Feature Modeling (W.F). For each web application level, this criterion investigates if modeling of *structural* and / or *behavioral* features of web applications are supported by the web modeling approach or not.

Development Phases (W.Ph). This criterion checks which phases of web application development, i.e., *requirements elicitation*, *analysis*, *design*, and *implementation*, are supported by the evaluated approaches.

Development Process (W.Pr). To which extent a developer is supported by a development process is covered by the *Development Process* criterion. Specifically, it distinguishes whether a well-defined development process is a *proprietary* one or is based on a *standard* development process, e.g., the Rational Unified Process (RUP) [Kru00]. Furthermore, it lists the detailed *steps*, output *artifacts*, and involved *actors*.

2.1.2.3 Customization Modeling

The *Customization Modeling* category explicitly deals with characteristics of the customization dimension in web application development. This includes criteria investigating support for modeling context information as well as the necessary adaptations. The following criteria are based on previous work [KPRS03] but specifically focus on the modeling level.

Context Properties (C.P). Although the relevant kind of context is specific to each UWA, a web modeling approach should support a set of common context properties including *user*, *location*, *device*, *time*, and *network*. Consequently, this criterion evaluates if the approach *supports* explicit concepts for modeling context and context properties, and what context properties have been *used in modeling examples* illustrating the approach.

Context Extensibility (C.CE). It is required that built-in modeling concepts for context properties can be easily extended by additional ones in case a UWA needs further context information (e.g., temperature). This criterion, thus evaluates to *supported* or *not supported.*

Chronology (C.C). This criterion tells if the approach *offers* (or *does not offer*) concepts that allow modeling how contextual information changes over time. For example, considering video streaming, information on how the bandwidth changed in the past can be used to infer how the bandwidth will develop or how stable it can be considered in the future in order to be able to tune the resolution of the video accordingly.

Complex Context (C.CC). Complex context information is aggregated using different context properties, e.g., ”Vienna at night”. In this respect, this criterion evaluates if an approach *supports* or *does not support* appropriate modeling concepts to specify complex context.

Separation of Context (C.SC). Customization modeling should ensure the separation of context information and not just intermingle context with adaptation or the web application itself, i.e., usually the content model. This criterion tests whether an explicit representation, e.g., in terms of a *separate context model*, would allow for reusability of already defined context information across several UWAs.

Adaptation Operations(C.O). This criterion evaluates if the approach supports *predefined modeling concepts for adaptation operations*, e.g., filter some content, add links, change resolution of an image, change hypertext, etc.

Adaptation Extensibility (C.AE). Similar to the required extensibility of modeling concepts for context properties, this criterion evaluates whether adaptation operations can be extended by user-defined adaptation operations. The criterion evaluates to *supported* or *not supported.*

Complex Adaptations (C.CA). Complex adaptations define multiple adaptation operations performed on the same or on different subjects, e.g., change color and resize an image. This criterion evaluates if an approach *provides* or *does not provide* appropriate means of modeling complex adaptations.

Adaptation of Levels (C.L). Customization influences all levels of web applications. Content adaptation changes the information that is presented to the user by adding or removing content (e.g., filtering), hypertext adaptation changes the navigation structure (e.g., disabling a link), and presentation adaptation changes the way information is presented to the user (e.g., changing colors or modality). This criterion investigates at *which web application levels* an approach offers concepts to model adaptations.

Adaptation of Interfaces (C.I). Customization may also influence the interfaces between levels, e.g., queries to underlying web application levels may need to change due to a certain context. Consequently, this criterion evaluates if modeling of adaptations with respect to interfaces is *supported* or *not supported.*

Granularity (C.G). The granularity of adaptation indicates the number of modeling concepts affected by a certain adaptation. While *micro* adaptation is concerned with fine-grained adaptations by affecting a single application element only (e.g., disabling a certain link on a certain page), *macro* adaptation means that rather large parts of a model are adapted, thus affecting multiple modeling concepts (e.g., changing the language or changing the prevailing access structures).

Separation of Adaptation (C.SA). Customization modeling should ensure the separation of adaptation modeling and thus, prevent an intermingled representation of adaptations within the content, hypertext, and presentation levels. This criterion tests whether separation of adaptation is *supported* and thus, allows for reusability of already defined adaptations within the same or across several UWAs, or if it is *not supported*.

Customization Phases (C.CP). This criterion evaluates during *which development phases* an approach considers customization modeling. Ideally, customization is considered during all development phases.

2.1.2.4 Model-Driven Engineering

The *Model-Driven Engineering* criteria focus on modeling language definitions, model transformations, and platform descriptions as a prerequisite for successfully employing MDE.

Language Definition (M.L). This criterion evaluates if a web modeling language has been defined explicitly in terms of a *metamodel* (including *UML profiles*), a *grammar*, a *semantic description* in terms of semantic web technologies, or if such a definition is missing.

Model Transformation Types (M.T). The investigated approaches might *support* or *not support* various types of model transformations such as transformations between platform-independent models (*PIM2PIM*), transformations between platform-independent and platform-specific models (*PIM2PSM*), transformations between platform-specific models and code (*PSM2-Code*), and transformation from platform-independent models to code directly (*PIM2Code*).

Platform Description Model (M.P). This criterion evaluates if the information about a platform is represented separately within a *platform description model*, or if this information is *implicitly* captured within the transformation rules.

2.1.2.5 Tool Support

For those approaches offering tool support, this category of *Tool Support Criteria* provides detailed information about the tool.

Tool Base (T.B). This criterion checks if the tool is developed as a *stand-alone application* or as a *plug-in/extension* to an existing tool.

Open Tool (T.O) Whether the offered tool support can be tailored to the developer's needs or not, is tested by this criterion. Either the tool explicitly foresees *extension possibilities* or the tool's source code is publicly available under an *open-source license* and thus can be changed.

Version (T.V). In order to know how far the tool has developed, this criterion records the *current version* of the tool at the time of evaluation.

Costs (T.C). A tool might be free of charge or it might require a license fee to be paid. This criterion evaluates to either *freeware* or *commercial*.

Modeling Support (T.M). A tool supporting a specific approach further can be evaluated if web modeling and/or customization modeling is *supported* or *not supported*.

Model Pre-Generation Support (T.MG). Furthermore, a tool might also assist a developer in modeling by pre-generating (parts of) models to be refined later on. This criterion evaluates to *true* or *false*.

Consistency Check (T.CC). The criterion tests if a tool has a *built-in consistency check* to verify the correctness of the models or not.

Code Generation (T.CG). Tools may generate code from the models the developer has defined. The criterion evaluates to *true*, if a code generation facility is available, otherwise it evaluates to *false*.

Process Support (T.P). The kind of guidance through the development process within the tool is evaluated by this criterion. Firstly, it indicates whether the process supported by the tool is realized for the approach as described in literature. Secondly, this criterion details if the developer is bound to a *step-wise* procedure, or has the possibility to go back and forth between the development steps without loss of information (i.e., *wizard-like*), or can freely choose where to start and what to model as long as all *phases* are processed.

Collaboration (T.Co). The evaluated tool might *support* or might *not support* version control and thus, allow collaboratively working on a project in a team.

2.1.3 Modeling Example: A Tourism Information Web Application

2.1.3.1 Motivation

In order to support the textual comparison on the basis of a structured set of criteria, an example is provided, which is modeled by means of the concepts of each web modeling approach. The example further enhances the evaluation in that it first, provides an initial insight into each approach and second, facilitates the comparison of the approaches' modeling means in terms of their notation, especially with respect to modeling customization. It has to be emphasized, however, that the role of the modeling example in this survey is of a supportive nature, only. It is not intended to give a comprehensive introduction to all of a web modeling language's features. In particular, the example shall provide insight to the approaches' means for modeling customization functionality.

The web modeling field, however, generally lacks reference modeling examples, which can be used to "assess" individual approaches, and in particular lacks modeling examples including comprehensive customization. Nevertheless, there have already been some attempts proposing some kind of reference modeling example. A first attempt has been conducted at the *International Workshop on Web-Oriented Software Technologies (IWWOST 2001)*[1] by introducing a *Conference Management System* modeling example. The second attempt initiated at the *Workshop on Model-driven Web Engineering (MDWE 2005)*[2] is based on a *Travel Agency* modeling example. Furthermore, while surveying the different web modeling approaches presented in the following, it became obvious

[1] www.dsic.upv.es/˜west/iwwost01

[2] http://www.lcc.uma.es/˜av/mdwe2005

that some modeling examples have been used particularly often across several web modeling approaches, amongst them library systems, e-stores, and art galleries.

The main reason for not adopting one of these running examples is that the modeled web applications typically are limited with respect to the ubiquitous nature of those examples. If at all, customization functionality is very simple, e.g., encompassing content adaptations with respect to an isolated context factor such as the user, the device, or the location, only. Instead, the aim of this survey was to choose an example from a domain that possibly does not create any biases, e.g., due to reusing existing examples in some of the investigated approaches. The domain of tourism information systems is able to provide such examples requiring customization, especially, when considering mobile devices. Consequently, a hypothetical *Tourism Information Web Application* (TIWA) has been chosen, which has been inspired by *www.tiscover.at*, the official Austrian tourism platform,. In the following, a reduced description of the example system is given, which includes the core functionalities of a typical tourism information system. Most importantly, the system's required customization functionality are described in terms of five *customization scenarios*, which have been designed to support the respective criteria of the catalogue of criteria.

2.1.3.2 Core Functionalities

A typical TIWA is supposed to allow guest users, i.e., unauthenticated users, to browse and search for hotels, regions, activities as well as information about the weather. Registered users will be allowed to book rooms and to browse their prior bookings. Administrators are responsible for the web application's content and therefore have to execute typical CRUD operations for hotels, regions, users, activities, and bookings. The essential use cases are illustrated in Figure 2.2.

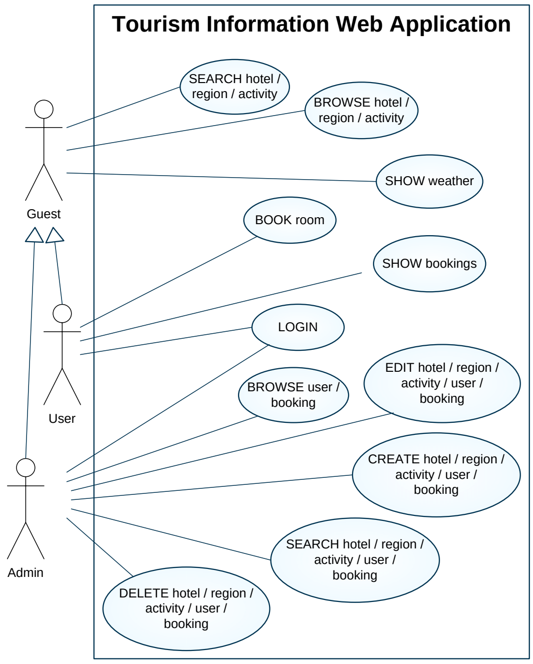
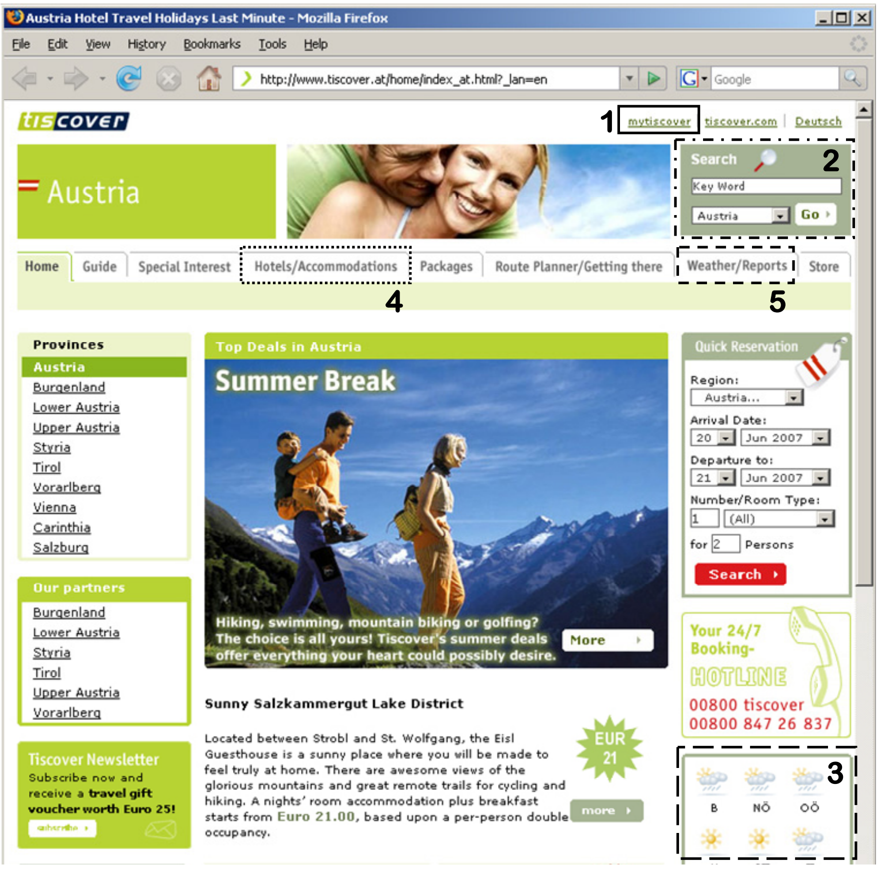

Tourism Information Web Application
SEARCH hotel / region / activity
BROWSE hotel / region / activity
SHOW weather
Guest
BOOK room
SHOW bookings
LOGIN
User
BROWSE user / booking
EDIT hotel / region / activity / user / booking
CREATE hotel / region / activity / user / booking
Admin
SEARCH hotel / region / activity / user / booking
DELETE hotel / region / activity / user / booking

Figure 2.2: Use Cases of the Tourism Information Web Application



Region: Austria...

Arrival Date: 20 Jun 2007

Departure to: 21 Jun 2007

Number/Room Type: 1 (All)

for 2 Persons

Hiking, swimming, mountain biking or golfing? The choice is all yours! Tiscover's summer deals offer everything your heart could possibly desire.

Sunny Salzkammergut Lake District

Located between Strobl and St. Wolfgang, the Eisl Guesthouse is a sunny place where you will be made to feel truly at home. There are awesome views of the glorious mountains and great remote trails for cycling and hiking. A nights' room accommodation plus breakfast starts from Euro 21.00, based upon a per-person double occupancy.

Your 24/7 Booking-HOTLINE 00800 tiscover 00800 847 26 837

Tiscover Newsletter Subscribe now and receive a travel gift voucher worth Euro 25! subscribe

Figure 2.3: The Tiscover Start Page

How such a web application could look like, is indicated with the following screenshot mock-up. Figure 2.3 shows the home page of the TIWA with dedicated areas for (1) login, (2) search, and (3) weather information. Furthermore, the menu allows navigating to, e.g., (4) hotels as well as more detailed (5) weather information. The TIWA allows users to browse the offer of hotels either for special regions or as the result of the user's search. These lists of hotels will provide essential information, e.g., the hotel's name, a picture, a short description, a quick link for booking, etc. Whichever hotel the user picks leads to an according detail page, allowing the user to obtain all the information available for the chosen hotel as well as the possibility to start a booking process. Likewise, the user will be able to browse regions as well as activities and request more information on either subject by following links to the according detail pages. Furthermore, if already logged in to the TIWA, the user is able to book a room and will be able to browse all prior bookings. Administrators have a special interface to create, edit or delete hotels, regions, activities, as well as bookings and users. Consequently, the content of the web application encompasses information on users, bookings, hotels and their special features, rooms, regions as well as activities.

2.1.3.3 Customization Scenarios

TIWAs naturally give space for a series of customization possibilities to better support user experience. For this survey five exemplary customization scenarios likely to be found in real TIWAs have been chosen such that the customization criteria of the catalogue are covered including different context factors.

(1) Customized Activities: Users will have the possibility to view activities that are relevant to them (C.L). The activities shall be filtered according to the user's age, location, the current time, and the weather at the user's location (C.P), (C.CE). From all activities, only those that match these criteria will be displayed. Similar scenarios could filter content according to the user's preferences, only, e.g., list all hotels according to the user's interest.

(2) Special Offers: For some users, there shall be special offers based on their navigational behavior (C.P). Users shall be presented a special offer, if they have visited several pages of a specific region (C.CC). This scenario describes adaptation of the hypertext level (C.L). Alternatively, a user that has already booked three times will get 3% off the price for the next reservation, which represents a content level adaptation (C.L).

(3) Administrator Links: Administrators (C.P) shall be provided with "edit" links for every concept of the content they are allowed to edit (C.L), allowing for an additional and possibly easier way of content management. For normal users or visitors these links will not be available requiring fine-grained hypertext adaptation (C.G).

(4) Multi-Delivery: In order to consider users equipped with small devices (C.P), an appropriate adaptation of the system shall be chosen. If a small-screen device is used, pictures and detailed descriptions shall not be shown. This can be achieved by creating a dedicated hypertext model for each device type which represents a coarse-grained hypertext adaptation (C.G) or by hiding content for devices with small displays, which represents a fine-grained content adaptation (C.L).

(5) Current Season's Style: The *Look & Feel* of the web application shall be different according to the season. Different styles can be chosen during spring, summer, autumn, and winter. The *Look & Feel* might also be subject to customization for supporting users having certain disabilities, e.g., visually impaired users (C.L).

2.2 Comparison of Approaches

In the following the seven selected web modeling approaches will be compared applying the catalogue of criteria presented in Section 2.1.2 and exemplified using the modeling example described above (cf. Section 2.1.3). Each of the following sections is dedicated to one approach which will be ordered along with the categories they belong to, starting with the category of *data-oriented* web modeling approaches.

2.2.1 The Web Modeling Language (WebML), Ceri et al.

MATURITY

WebML[3] [CFB$^+$03] is one of the most elaborated web modeling languages stemming from academia and is supported already over several years by the commercial tool *WebRatio*[4]. Furthermore, WebML and WebRatio have already been successfully employed in several real-world projects

[3] www.webml.org

[4] www.webratio.com

(G.A). WebML is explained by a large number of different modeling examples including various forms of e-Stores selling different kinds of products, a conference management system, an online travel agency, a loan brokering web application, a museum guided tour, a campus tour, and an e-learning application (G.ME). Since its introduction in 1999, WebML has continuously evolved and recently has been extended by additional concepts addressing context-aware [CDMF07], service-enabled [MBC^{+}05], workflow-based [BCFM06], and semantic [BCC^{+}06] web applications as well as rich internet applications [BCFC06] (G.T).

WEB MODELING

The WebML language as presented in [CFB^{+}03] provides modeling concepts for content modeling and hypertext modeling, only. While the content level resembles the well-known ER-model [Che76] (cf. Figure 2.4*(a)*), at the hypertext level a proprietary graphical notation is provided (cf. Figure 2.4*(b)*) (W.L). Available within the WebRatio tool there are additional means for configuring the presentation level (e.g., for defining the web page's style sheet and the positioning of information on a web page) which, however, are not part of the WebML language [CFB^{+}03]. The interface between content and hypertext level is specified graphically by denoting the *entity* of the content level (e.g., *Hotel* in Figure 2.4*(a)*) to be displayed in a so-called *content unit* (e.g., *Hotels* in Figure 2.4*(b)*) as well as in a textual representation to specify additional properties. In Listing 2.1, the textual representation of the *IndexUnit HotelList* of the *Hotels* page in Figure 2.4*(b)* is given.

Listing 2.1: WebML: Textual Representation of the IndexUnit *HotelList*

```
IndexUnit HotelList
(source Hotel;
attributes Name, Picture , Description;
orderby Name)
```

Since specified at the hypertext level, however, the interface cannot be modeled separately (W.I). The approach allows behavioral modeling to a limited extent, only. While UML activity diagrams are used during the requirements specification phase, some behavioral features are represented in the hypertext model by the control-flow-like semantics of WebML's *operation units* (cf. *ModifyUnit* and *ConnectUnit* in Figure 2.5). Recently, the introduction of process modeling concepts into WebML [BCFM06], as well as the introduction of customization modeling concepts [CDMF07], allows further modeling of behavioral aspects at the hypertext level (W.F). The WebML approach comes with its own 7-phase, iterative, and incremental development process based on Boehm's Spiral model [CFB^{+}03] (W.Pr) and comprises all development phases from requirements to implementation (W.Ph).

CUSTOMIZATION MODELING The WebML approach provides two proposals for modeling customization [CDMF07], [CDFM07] for which it is not clear how they are related and how they can be used together, according to available literature. Neither of the proposals provides an explicit context model. Instead, context is modeled within the content model. In the first proposal [CDMF07], however, the modeler is supported in designing the context information with guidelines proposing to imagine the content level as a set of overlapping *sub-schemas* (C.SC). For illustration purposes, these sub-schemas can be indicated in the content model (cf. Figure 2.4*(a)*). The entities of the *Basic User Sub-Schema* are always available in a WebML model and consists of the *User*, *Group*, and *Module* entities. The *Personalization Sub-Schema* associates the *User* entity with other entities, e.g., to denote user preferences. The *Context Model Sub-Schema* contains entities relating the *User* entity with context information, e.g., the user's location, the device used, etc. (C.CE). As a form of static adaptation, WebML *siteviews*, i.e., several hypertext models defined upon the content level, on the one hand, may be used to personalize a web application according to users and user groups and, on the other hand, serve the purpose of expressing alternative forms of content presentation for different devices [CDM03] (C.P). Modeling complex context is not explicitly supported within WebML (C.CC). For modeling adaptations, the idea of a context-aware web page (cf. *Activities* Page in 2.5) and concepts for retrieving context information have been introduced. For example, for retrieving context information from the content level, the *GetDataUnit* concept is employed (cf. *GetRegion* GetDataUnit Figure 2.5), while the *GetClientParUnit* concept is used to obtain context information from the client (cf. *GetLatitude* GetClientParUnit Figure 2.5), e.g., information on the user's location or on wireless connectivity [CDFM07] (C.P). Furthermore two predefined adaptation operations are available, namely *ChangeSiteview* (cf. Figure 2.7*(a)*) and *ChangeStyle* (cf. Figure 2.7*(b)*) (C.O), which allow for a coarse-grained adaptation of the hypertext and the presentation level, respectively (C.L), (C.G). The possibility of changing the navigation flow, e.g., through WebML's *IfUnit* (cf. Figure 2.6*(b)*), enables more fine-grained adaptations. There is no way to explicitly adapt the interface between the content and the hypertext level (C.I), since adaptations are defined for a page rather than for a content unit. It is not possible to extend the predefined set of adaptation operations (C.AE), neither is there a concept for defining complex adaptations (C.CA). In WebML, UWAs are modeled as refinements of the models of a non-ubiquitous web application, i.e., the development process has not yet been adapted to guide developers in considering customization functionality throughout the development lifecylce (C.CP). Since adaptations are modeled as an extension of the hypertext model, WebML does not provide a separation of adaptations from the rest of the web application model (C.SA). In the second proposals for modeling UWAs with WebML, the focus is on personalization purposes and allows defining adaptive behavior of a web application based on event-condition-action (ECA) rules [CDF06]. The approach relies on adaptive pages that specify the action part of the rule and are similar to context-aware pages. Conditions are described on the basis of a so-called *web behavior model* (WBM) script, which is a timed state-transition automaton for representing classes of user behaviors on the web, i.e., navigation patterns. Thus, some form of context chronology is supported (C.C).

MODEL-DRIVEN ENGINEERING

WebML's language concepts are partly defined in terms of XML document type definitions (DTDs) (M.L) and partly hard-coded within the approach's accompanying tool, WebRatio. Consequently, the WebML approach currently cannot profit from the benefits of MDE. Nevertheless, some efforts have recently been made to port WebML to MDE by two proposals, i.e., one for a metamodel based on the Meta Object Facility (MOF) [SWK+07] (see also Chapter 4) and another one for a UML profile [MFV07]. While the WebRatio tool provides code generation support (i.e., J2EE and Jakarta Struts) from the platform-independent WebML models, the WebML approach does not provide for model transformations to different platforms prior to generating code (M.T) As a consequence, the approach does neither provide platform description models for the above mentioned platforms (M.P).

TOOL SUPPORT

WebRatio is a commercial, proprietary tool developed as a standalone application (T.C), (T.O), (T.B). WebRatio 4.3 has been evaluated with an academic license (T.V) which is free of charge (T.C). Currently, WebRatio does not include the concepts for modeling customization, except for requiring the *User*, *Group*, and *Module* entities in every web application model. Still, it is reported that, the concepts proposed in [CDMF07], [CDFM07] have been realized in prototype implementations (T.M). The tool allows extending the WebML language via specialized *content units* and *operation units* (T.O). This way the WebML language and tool support have already been extended e.g. to support service-enabled web applications. WebRatio allows the to generate a web application model into a running application. No additional coding is needed and the web application can be automatically deployed to an integrated Tomcat Servlet Container (T.CG). Moreover, the tool offers a so called *pattern wizard* that allows building automatically common parts of the hypertext model like a login mechanism or content management functionality for selected entities from the content level (T.MG). WebRatio models can be subject to a consistency check that is executed on-demand (T.CC). The process of the method is only partly supported by the tool, e.g., requirements engineering is not considered, and the user can start wherever s/he chooses (T.P). According to the included documentation of WebRatio the tool is also capable of shared editing via a concurrent versions system (CVS) (T.Co).

Modeling Example

In Figure 2.4*(a)* the content level, i.e., an ER-diagram, of the TIWA example is shown for WebML. In WebRatio, the entities *User*, *Group*, and *Module* are predefined and enable basic personalization of the hypertext level to user and user groups (cf. the *Basic Sub-Schema* indicated in Figure 2.4*(a)*). The other entities are representing the application data of the running example: *Hotels* have *Rooms*, which in turn can have multiple *Bookings*, whereby each *Booking* belongs to a *User* (cf. the *Personalization Sub-Schema* indicated in Figure 2.4*(a)*). *Hotels* have additional *Features* such as a swimming pool, an animation team or the like. Every *Hotel* is situated in a *Region* which in turn has neighboring regions. In every *Region* there will be an offer of *Activities*, e.g., in terms of events. Further context information has been incorporated to the content level as follows: The *User* is directly related to the device s/he is using. Additionally, the *User* is also directly associated with the *Region* s/he is currently located as well as indirectly to the *Weather* entity that has been introduced to allow for customization according to the current weather situation (cf. the *Context Model Sub-Schema* indicated in Figure 2.4*(a)*).

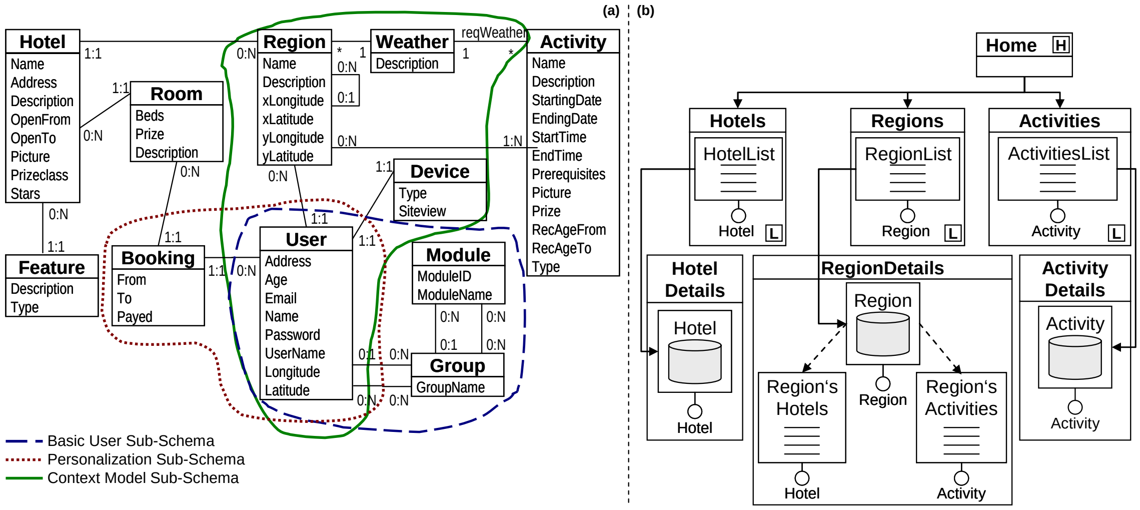


Figure 2.4: WebML: (a) Content Level, (b) Hypertext Level

Part of the hypertext level of the TIWA is shown in Figure 2.4*(b)*. This particular view shows that users can browse hotels, regions, and activities via three dedicated landmark pages (cf. 'L' label). The specific subjects can then be selected via WebML's IndexUnits to be displayed in detail on separate pages, i.e., *HotelDetails*, *RegionDetails*, and *ActivityDetails*. Each of these pages uses a DataUnit for presenting the selected item. In addition, the *RegionDetails* page presents an index of the region's activities and hotels, respectively.

Customization Scenario Customized Activities. The *Activities* page can be made context-aware in order to show only activities of the region the user is currently located in and are employed to filter them according to the user's age and the current date. A GetUnit *GetUser* acquires the current user and two GetClientParUnits provide the user's longitude and latitude (cf. Figure 2.5). The location information then can be stored for the current user in the content model with a ModifyUnit. The location information is also used to select the region the user is currently located with the *GetRegion* GetDataUnit. The ConnectUnit is used to relate the user with the current region, i.e., the user's location. A TimeUnit which has been implemented in WebRatio, can be used to obtain the current date so that only those activities are shown that have not yet started. The current weather situation for the user's region is assumed to be updated in the data source by some external service so that it can be queried by the GetDataUnit *GetWeather*. In order to correctly filter the information on activities for the user, all this context information is then used in a set of so-called SelectorConditions (cf. square brackets) for the *ActivitiesList* IndexUnit in the page *Activities*.

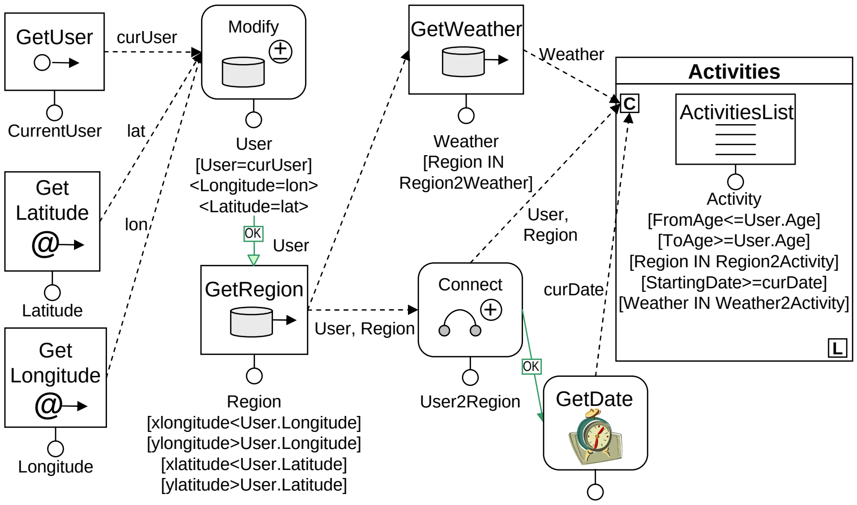


Figure 2.5: WebML: Customized Activities Scenario

Customization Scenario Special Offers. For modeling the *Special Offers* customization scenario the second proposal for modeling UWAs with WebML is used, i.e., WebML's rule-based approach which allows reacting to user navigation. In the following example, the TIWA will monitor the user navigation to find out what region a user is particularly interested in. As soon as this information is available, the user shall be redirected from the *HotelDetails* as well as from the *ActivityDetails* page to a page presenting special offers for hotels and activities. As depicted in Figure 2.6*(a)*, the condition of this rule is specified by a WBM script, i.e., a timed finite state automaton, stating that a user needs to visit three arbitrary pages (denoted with '*'), having in common that each displays information about the same region. In a WBM script, a state (denoted with a circle) represents the user's inspection of a page. The three pages do not necessarily have to be visited in sequence, i.e., the user can navigate to other pages in between. Still, the transition constraint states that after 180 seconds the next page displaying the same region information has to be navigated otherwise the script will be discarded. In case the third page is visited the automaton reaches its accepting state, triggering the action of the rule. In the action part of the rule, the *HotelDetails* and the *ActivityDetails* pages become adaptive pages (A), having a link to the *Special Offers* page, which will be navigated in case the rule's condition is true.

Customization Scenario Administrator Links. In WebML, *siteviews* can be used for modeling different hypertexts for different user groups, e.g., external users and administrators. In the following, to enable editing from the public siteview, fine-grained adaptation can depend on the user's group links shall be allowed in the *Activity Details* page of the public siteview, however. In Figure 2.6*(b)*, depending on the current user's group, the *ActivityDetails* page will present different navigation possibilities. The Alternative concept is employed to indicate that the normal *Activity* DataUnit is displayed in the default case (cf. 'D' label). Alternatively, an *Activity* DataUnit that has a link to an *EditActivity* page can be displayed. An IfUnit is employed to redirect navigation to the respective alternative on the basis of the context retrieved by the *GetUser* GetUnit and the *GetGroup* GetDataUnit.

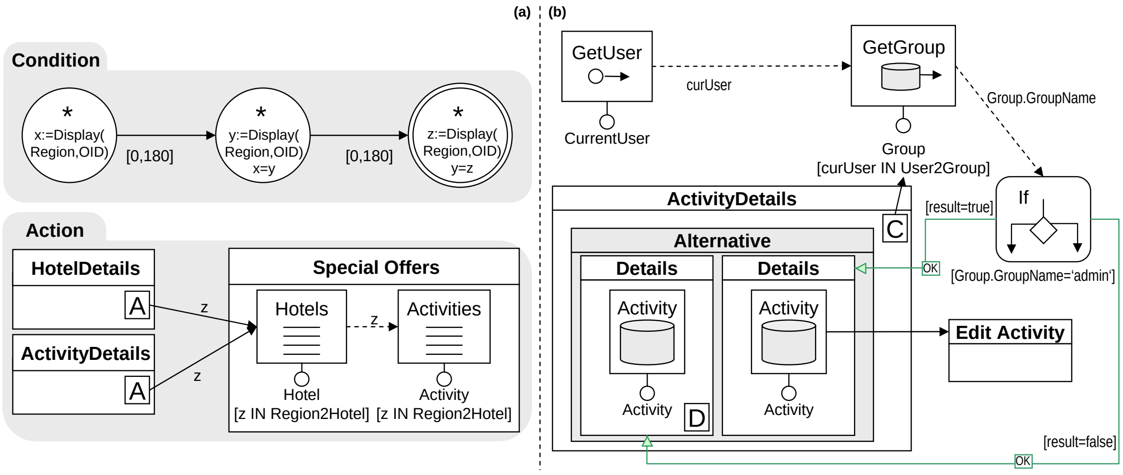


Figure 2.6: WebML: (a) Special Offers Scenario, (b) Administrator Links Scenario

Customization Scenario Multi-Delivery. WebML also suggests modeling dedicated siteviews for special navigation purposes as is the case for adapting the hypertext level according to the user's device. The ChangeSiteViewUnit, used in this scenario, has been recently introduced to the WebML language to allow for changing the hypertext model to better suit the current context of the user. In Figure 2.7*(a)*, the *Public* Siteview has been made context-aware. Thus, if a user requests any page located in this siteview, the GetClientParUnit *GetDevice* will provide the necessary information to find a device specification in the database via the GetDataUnit *GetDeviceSpec*. The user will then be redirected to an appropriate siteview that has been associated to the device in the database by the ChangeSiteviewUnit. In case no such specification can be found, the user will be redirected to a page where the device can be chosen from a predefined list.

Customization Scenario Season's Style. Finally, WebML does also allow customizing the presentation level with the recently introduced ChangeStyleUnit. This coarse-grained adaptation operation changes the *Look & Feel* of the web application by changing the Cascading Style Sheet[5] (CSS) used according to the current context. Figure 2.7*(b)*, shows how the time context is obtained with a TimeUnit *GetDate*. An IfUnit is used to decide if the presentation of the home page shall be based on the summer style or the winter style by using the respective stylesheet.

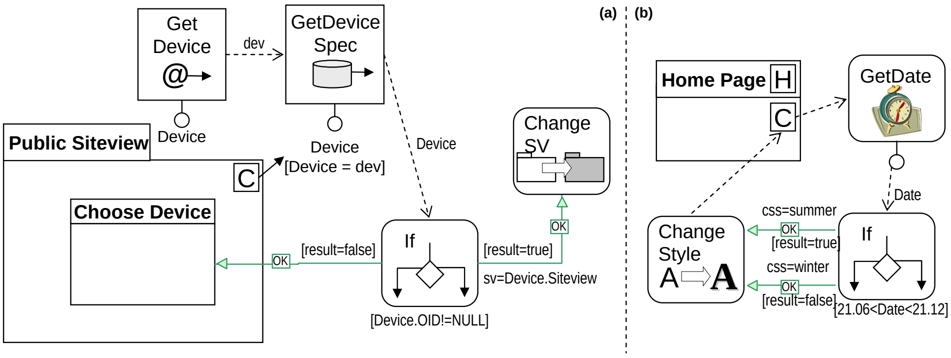


Figure 2.7: WebML: (a) Multi-Delivery Scenario, (b) Season's Style Scenario

2.2.2 The Hera Design Methodology, Houben et al.

MATURITY

The *Hera*[6] approach [FHB06], first introduced in 2000, has been heavily influenced by the hypermedia design method RMM [ISB95] and originally concentrated on web applications that, depending on a user query, automatically generate complete, static hypermedia presentations [BFH02]. Recently, a revision of the Hera approach has been introduced with Hera-S [vdSHBC06], which considers dynamic web applications (G.T). The prevailing modeling example used in almost all publications is a virtual art gallery. Besides, a conference management system, a digital photo library, a movie library, and e-store applications are used as modeling examples (G.ME). While it seems that real-world applications of Hera do not exist yet, four demo web applications have been developed including the museum application where real-world content has been reused (G.A).

WEB MODELING

The Hera approach, in principle proposes models for the content, hypertext, and presentation levels (W.L). Since RMM has been a major influence to Hera [FHV01], the content level (i.e. Hera's *conceptual model* or alternatively *domain model*) first has been based on the ER-model [Che76]. Today, Hera is based on the Resource Description Framework (Schema) - RDF(S) [W3C04c] thus, supporting the engineering of semantic web information systems. It provides a proprietary graphical notation for modeling the content level [FHB06]. As is depicted in Figure 2.8*(a)*, at content level the domain model of Hera is based on *concepts* (e.g. Hotel and Region), attributes (e.g. Name and Picture), *concept relationships*, (e.g. provided_by and provides) and *media types* (e.g. String and Image). The hypertext level (i.e., the *application model*) and the presentation level (i.e., the *presentation model*) still resemble RMM's graphical notation as is shown in Figure 2.8*(b)* and Figure 2.9, respectively. More specifically, the Hera application model is mainly based on *slices*

[5] http://www.w3.org/Style/CSS/
[6] http://wwwis.win.tue.nl/h̃era/

and *slice relationships*, which can be either compositional or navigational. In the upper part of Figure 2.8*(b)*, the *HotelDetail* slice has *Hotel* as the owning concept from the content level (cf. the rounded rectangular) and presents the details of the hotel, including its name, picture, prizeclass, stars, description and address. Furthermore, the set of rooms it contains and the features it provides are part of the slice. From the *HotelDetail* slice, a navigational relationship leads to the *HotelEdit* slice. The presentation level is based on a hierarchy of *regions* allowing to position slices from the hypertext level for their presentation [FFH$^+$04]. A region represents a rectangular part of the display area and has associated with it a layout manager for positioning elements and a specific style. Figure 2.9 describes how the *HotelDetail* slice and its parts are placed within different regions. For example, the *HotelDetail* slice is placed within a Box Layout. The figures show that the interfaces between levels are specified graphically. For example, at hypertext level, a slice always denotes its owning concept from the content level in terms of a rounded rectangular (Figure 2.8*(b)*). Moreover, the interfaces between the levels are as well specified textually on the basis of queries [vdSHBC06]. Still, the interface is always intermingled with the upper level (W.I). The recent introduction of concepts for modeling workflow-based web applications (e.g., a task model and a workflow model) [BFH06] allow for modeling behavioral aspects at the hypertext level (W.F). Hera proposes a development process that includes development phases from requirements engineering to implementation [VH02] (W.Pr), however, this process has not been detailed beyond a set of guidelines. Recent elaborations on the approach focus particularly on design and implementation but omit the requirements engineering phase (W.Ph).

Customization Modeling

In the Hera approach customization is focused on the user and device context, only (C.P). The approach strives for encapsulating context information separately (C.SC): On the one hand, static adaptations are based on context information obtained from a *user profile* [FBHF04], [FFH$^+$04]. In fact, Hera uses a CC/PP vocabulary [W3C04a] to model user preferences and device capabilities. For example, in the profile instance presented Listing 2.2) the hardware platform is characterized by the property *client* which identifies the device used, while the *group* property characterizes the user preferences in terms of providing information on the user's group. On the other hand, dynamic adaptations, that are based on a user model storing the users' navigational behavior, have been proposed but not further explained [FH02]. Moreover, further concepts are needed for supporting extensibility of context explicitly (C.CE), for capturing context history (C.C), and for modeling complex context information (C.CC). Adaptation is realized by means of so-called *appearance conditions* attached to different design artifacts, i.e., to the content, hypertext, and presentation models (C.L), (C.I). If such conditions evaluate to true or false, the presence of their associated artifacts, e.g., content information or links (C.G), is enabled or inhibited, respectively [FBHF04]. In this respect, appearance conditions can be seen as a from of adaptation operation (C.O). As a consequence, the approach does not allow for extensibility of adaptation operations nor for complex adaptations. (C.AE), (C.CA), Since adaptations are only annotations to different design models (C.CP), they cannot be modeled separately either (C.SA). Still, in the Hera-S approach the recent introduction of concepts from the aspect-orientation paradigm is intended to solve this problem [CWH07]. This way, *appearance conditions* can be modeled separately and be "woven" into the models before generating the web application. Another proposal for customization presents how a high-level personalization rule language [GGBH05], can be mapped onto modeling concepts from Hera and OO-H (cf. Section 2.2.6).

Model-Driven Engineering

As already mentioned before, the Hera modeling language is defined in terms of in RDF(S) and CC/PP, respectively (M.L). Currently, the approach focuses on the generation of static hypermedia presentations, i.e., static web sites, in a specific output format, e.g., in terms of HTML and WML. These hypermedia presentations then will be deployable on any web server. The approach suggests the subsequent transformation or rather the integration of the different design model artifacts, i.e., Hera's conceptual model, application model, presentation model, and user profile model. In a last step, the output can be generated (M.T). The approach does not support a separate platform model (M.P).

Tool Support

Hera is supported by the *Hera Presentation Generator* (HPG) tool, which has been evaluated in its 1.3 version (T.V). HPG is a proprietary tool, not extensible, available as freeware (T.B), (T.O), (T.C). Support for a concrete notation for modeling the content and the hypertext as well as presentation level has been realized by three Microsoft Visio[7] templates called the *Conceptual Model Builder*, *Application Model Builder*, and the *Presentation Model Builder*, respectively [FHB06]. A load/export feature provides the RDF(S) serialization of the models for the HPG tool. Apart from that, modeling of the user profile, i.e., the CC/PP code, is supported through appropriate "textual" wizards of HPG (T.M). Since HPG 1.3 generates static hypermedia presentations, it does not allow for modeling dynamic adaptations that consider e.g. the user's inputs or the user's browsing history [FH02]. Today, transforming the models into a suitable static presentation (HTML, WML, etc.) is possible on the basis of XSL transformations using either HPG itself or the external AMACONT engine developed at the Dresden University of Technology [FHHF04] (T.MG), (T.CG). The HPG tool follows the guidelines for developing a web application with Hera by starting with the content level, i.e., Hera's *domain model.* A wizard guides the user through the design and generation process (T.P). In every step the models can be viewed using a text editor or Microsoft Visio, where the model builders enforce some constraints while the user is designing a model in order to produce correct models. Moreover, the models' consistency can also be checked by HPG (T.CC). Up to now, there is no shared editing or versioning support (T.Co). Recently, modeling support for user input and tool support for generating dynamic web applications, which are able to react to user input, has been proposed in [HFBV04] and [FHB06], respectively. This new tool support is currently implemented in Java and provides a runtime environment for the generated applications on the basis of the Java-Servlet platform.

Modeling Example

In the Hera approach, not all customization scenarios can be realized: Since the approach is limited to user and device context information as well as to appearance conditions which do not allow filtering of content, it is possible to present the *Multi-Delivery* and the *Administrator Links* scenarios, only.

Customization Scenario Multi-Delivery. The content level as well as the other levels can be customized with annotations, i.e., *appearance conditions* such as *prf:imageCapable = true* in Figure 2.8*(a)*.

[7] http://office.microsoft.com/en-us/visio/

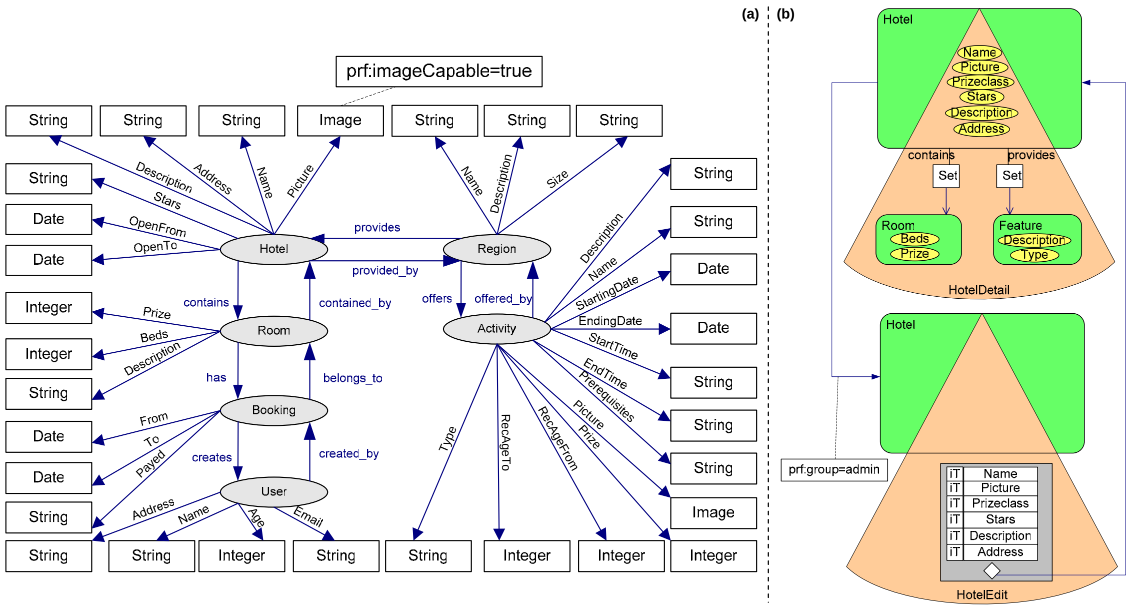


Figure 2.8: Hera: (a) Conceptual Model, (b) Application Model

This appearance condition is specified on the basis of the user profile instance of Listing 2.2 and means that the conceptual model is to provide a picture of the hotel only if the device used is capable of presenting images. Since according to the *client* property in the profile instance, the *imageCapable* property is set to true, the hotel picture will remain in the *domain model* and thus in the generated hypermedia presentation.

Listing 2.2: Hera: User Profile Instance

```
<Description rdf:about= ”Profile” >
  <ccpp:component>
    <HardwarePlatform>
       <imageCapable>true</imageCapable>
       <client>Desktop</client>

    </HardwarePlatform>
  </ccpp:component>
 <ccpp:component>
    <UserPreferences>
       <group>admin</group>
        ...
    </UserPreferences >
  </ccpp:component>
```

With respect to the presentation level, appearance conditions can also be used to choose one of a set alternative regions according to the current device. In Figure 2.9, the set of rooms of the HotelDetail slice are placed at the bottom part of the display (cf. the regions *RegionBottomRooms*). More specifically, two alternatives for presenting the hotel's set of rooms are provided depending on the device used. The appearance conditions associated with the two alternative regions *RegionBottomRooms* state, that in case of a PDA the rooms also shall be displayed according to a Box Layout, while when using a PC users shall be presented with a Table Layout.

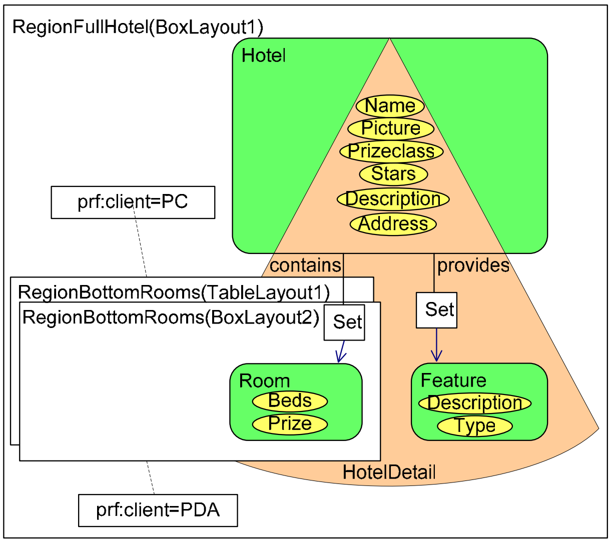


Figure 2.9: Hera: Presentation Model

Customization Scenario Administrator Links. Similarly, appearance conditions can be used to customize the hypertext level, e.g. a link in the Hera application model can be adapted according to the user profile. Figure 2.8*(b)* presents a small excerpt from the application model showing how the *Administrator Links* scenario can be realized. The link from the *HotelDetail* slice to the *HotelEdit* slice is annotated with the appearance condition `prf:group = admin` stating that only users of the group 'admin' will be able to see this link. According to the *group* property in the profile instance presented in Listing 2.2, the user is part of the admin group and thus, the edit link will be available. Another way of hypertext adaptation is the possibility to define multiple application models for the content level. In this respect the application models or rather the RDF(S) specifications just need to be exchanged and the web application can then be re-generated. Please note that appearance conditions also can be associated with a whole slice or one of its shown attributes, e.g., the hotel's description.

Following, Listing 2.3 indicates how such an appearance condition can be modeled separately as is proposed in [CWH07] using the textual Semantics-based Aspect-oriented Adaptation Language (SEAL) of Hera-S . In the example below, the ADVICE specifies that an *appearance condition* needs to be added to the hypertext level, while the POINTCUT specifies the exact place(s). Please note, that in Hera-S the modeling concepts of the language partly have been renamed, e.g., unit corresponds to a slice and the reference to the user profile 'prf' has been changed to 'cm'. Consequently, the POINTCUT specifies the link from the HotelDetail slice to the HotelEdit slice. The advantage of this approach is that within the POINTCUT several links of the hypertext level can be specified in one place and extended with an appearance condition.

Listing 2.3: Hera: Separating Appearance Conditions with Aspects

```
POINTCUT type relationship and from (type unit and hasName
"HotelDetail") and to (type unit and hasName "HotelEdit")
ADVICE ADD CONDITION cm:group = "admin"
```

2.2.3 The Web Site Design Method (WSDM), De Troyer et al.

Overview

The *Web Site Design Method*[8] (WSDM) has been first proposed in 1998 [TL98] and thus, is one of the earliest web modeling approaches. Recently, WSDM is evolving towards the semantic web [CPT06] (G.T). There seem to be no applications of the approach in real-world projects (G.A). Modeling examples used are a conference management system, an e-store, and a department website (G.ME).

Web Modeling. WSDM provides its own five-phase, audience-driven development process (W.Pr) and specifies all its modeling concepts within the *WSDM Ontology*[9] which is based on the Web Ontology Language (OWL) [W3C04b]. The process starts from a *mission statement*, specifying purpose, subject and targeted users, followed by a user-driven requirements engineering and analysis phase, which results in the *audience model*, i.e., a set of *audience classes*, having specific requirements and characteristics. During the design phase, structural modeling of all web application levels is supported (W.L). The content level is represented as the integration of *object chunks* to the so-called *business information* model. Each *object chunk*, represents a tiny conceptual schema, which is derived from the tasks each *audience class* needs to perform. In turn, these tasks are identified in a task modeling activity based on the Concurrent Task Tree (CTT) technique [Pat00], whereby a task is decomposed into elementary tasks arranged in a temporal order (W.F) (cf. Figure 2.10). Then for each elementary task an object chunk is created, which (formally) models the necessary information and functionality needed to complete the task (cf. Figure 2.11). Moreover, an *object chunk* can have associated *object chunk functions*, which allow to model system functionality (e.g., instance creation and select functions) [CPT06], [PCT05] (W.F). Formerly, *object chunks* have been based on the Object Role Modeling (ORM) method[10] [Hal01]. Today, they are specified within the WSDM Ontology, while the graphical notation still follows ORM (cf. 2.11). At the hypertext level, WSDM's *navigational model* (W.L) consists of *navigation tracks*, one for each *audience class*, and models the conceptual structure of the web application in terms of nodes and links (cf. 2.12). A *navigation track* can be considered as a "sub-site" dedicated to the information and navigation needs of the *audience class*. Nodes can comprise several *object chunks* from the content level, which is specified with the OWL object property *hasChunk*. This relationship is also graphically represented in the *navigation track* (cf. rounded rectangles in Figure 2.12) (W.I). At presentation level, the *site structure model* (W.L) maps the concepts modeled at hypertext level onto pages, i.e., the OWL object property *hasNode* decides which nodes and links will be grouped onto web pages. Nodes to be presented on one page are surrounded with a rectangle shape (indicated for the *Show Bookings* node in Figure 2.12) (W.I). In addition, WSDM proposes *page models* defined for each separate page in the *site structure model*, which allows for positioning of page elements. The mapping of *object chunks* onto the actual data source is performed in a *data source mapping* step. Finally, the development process ends with the implementation of the web application (W.Ph).

Customization Modeling

In WSDM's history, two independent proposals for dealing with customization modeling have been published. The first one [CGT05], [CTB03] focuses on modeling adaptive navigation based on rules specified at design time and triggered due to the browsing behavior of users (C.P). The second [TC04], basically extends each of WSDM's development phases (except implementation) in order to model localization. To do so, the concept of "locality" has been introduced describing

[8] http://wsdm.vub.ac.be/Research/publications.php

[9] http://wise.vub.ac.be/ontologies/WSDMOntology.owl

[10] http://www.orm.net

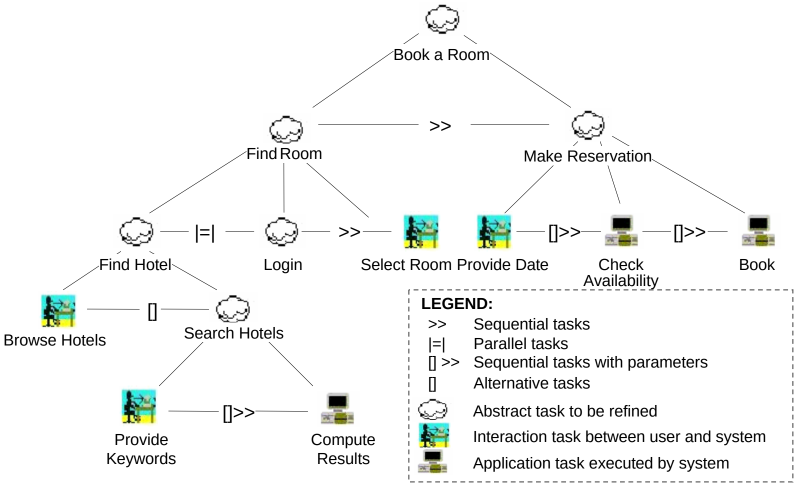


Figure 2.10: WSDM: The Booking Task CTT

a particular place, situation, or location (C.P). In particular, the audience modeling phase has been extended to model localities, i.e., their specifications, characteristics, and their mappings to audience classes. In further design models, localization is considered by annotation-like extensions to the models, which implies no separation of adaptations (C.SA). Context information actually is not modeled at the content level. Consequently, there is no separation of context from the rest of the application (C.SC). Context cannot be extended to support further context properties (C.CE), and neither chronology (C.C) nor complex context can be modeled (C.CC). In [CTB03], the *Adaptation Specification Language* (ASL) has been defined, allowing designers (C.CP) to express rules specifying when and how the hypertext level (C.L), needs to be adapted according to the monitored user behavior. In this respect, the approach offers an implicit context model providing several functions to query data on user behavior, e.g., *numberOfVisits* of a node and *numberOfTraversings* of a link. This way, some form of context chronology can be realized (C.C). The approach offers a fixed set of primitive adaptation operations (C.O), (C.AE) on nodes (i.e., *deleteNode* and *addNode*), connections between nodes and *object chunks* (i.e., *connectChunk* and *disconnectChunk*) (C.I), and links (i.e., *deleteLink* and *addLink*). More complex adaptations based on the primitive ones are amongst others *promoteNode* / *demoteNode*, which moves a node closer to / further away from the root of a web site thus making it easier / harder to find (C.CA). Rules in ASL can be applied to single elements but also to groups of elements (C.G) and represent a mechanism to specify adaptations separately from the rest of the web application (C.SA). The approach suggests that these design time rules can be used to generate rules for an adaptation engine within the web application but does not explain possible effects of navigation model rules on the presentation level.

Model-driven Engineering

As already mentioned before, all modeling concepts of WSDM have been defined within the WSDM Ontology. WSDM models thus, represent instances of this ontology (M.L). The WSDM approach does foresee an implementation phase with a four-step transformation process, respectively a five-step process when considering semantic annotations [CPT06]. In this process, the previously defined models are supposed to be subsequently transformed to the selected output platform, e.g., (X)HTML and WML (PIM2Code). During this process, the different models are integrated into one, thus realizing a PIM2PIM transformation (M.T). The approach, however, does not provide platform description models for the above mentioned platforms (M.P). A prototype for generating code and particularly semantic annotations from WSDM's design model has recently been described in [CPT06]. This prototype currently is able to generated HTML code based on XSLT.

Modeling Example

On the basis of WSDM's current customization mechanisms, it is possible to model the *Customized Activities* scenario, the *Administrator Links* scenario, and a variant of the *Special Offers* scenario. The customization scenarios *Multi-Delivery* and *Season's Style* could possibly be realized by defining different variants of the hypertext and presentation levels for supporting different devices as well as for different seasons. Still, how the different variants are selected in the web application at runtime is assumedly left to implementation issues. Consequently, these customization scenarios have not been realized.

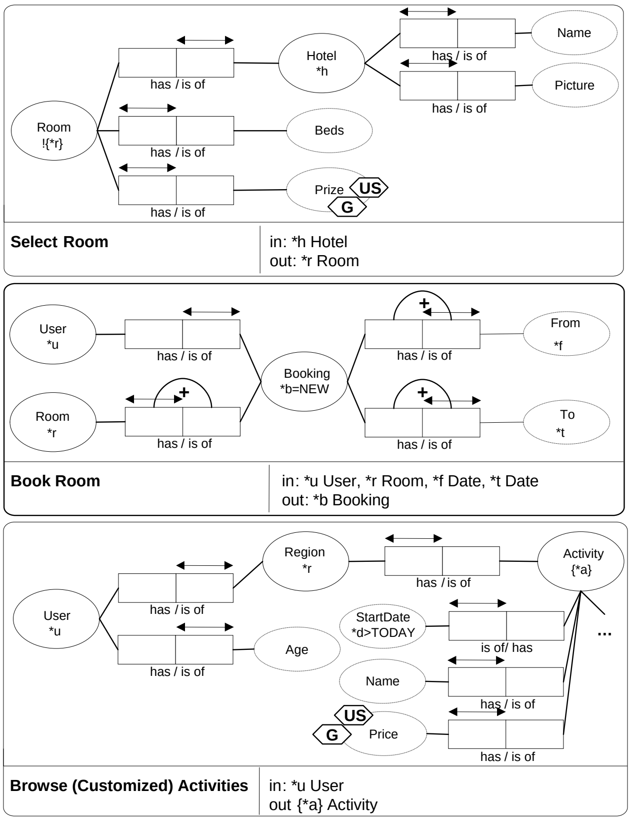


Figure 2.11: WSDM: Object Chunks at Content Level

Customization Scenario Customized Activities. In WSDM, the activities can be filtered according to the current context at the content level. The content level, i.e., the object chunks, is derived from a CTT specifying the details of an audience class's task. The task of browsing activities is rather simple. Consequently, for illustration purposes, an example for the more complex booking task of the user audience class is specified by the CTT in Figure 2.10: To book a room, a room has to be found and after that a reservation has to be made. To find a room, a hotel must be found first. Either before or after that the user has to login and can then select a room that s/he likes, etc.

Some of the object chunks capturing the information and functional requirements of an elementary task are depicted in Figure 2.11. The *Select Room* object chunk shows the information requirements for the *Select Room* task of the CTT shown in Figure 2.10. The input to the object chunk is an instance *h of the hotel object type for which name and picture are shown as the only value types. From the set of rooms of the hotel which is denoted with brackets {*r}, one can be selected as is indicated with '!'. The price value type of the room is tagged with locality labels to indicate information that is dependent on the locality, i.e. the price is different for localities 'G' for Germany and 'US' for United States. The object chunk *Book Room* creates a new instance (annotated by the keyword 'NEW') of booking for a specified room and date, which are inputs from prior tasks, namely *Select Room* and *Provide Date*. The relationships are established with the '+' notation. The object chunk *Browse (Customized) Activities* of Figure 2.11, shall present customization scenario *Customized Activities*. In the example, the location of the user is assumed to be resolved to a region and to be stored in the data source. Thus, having the user *u as input to the object chunk, the region for the user as well as the region's activities can be selected. Furthermore, the activities are selected according to the current date with the keyword 'TODAY'. Please note that, the WSDM specification actually does only allow for testing the equality of values with '==', thus the example is not fully compatible with WSDM. Furthermore, the notation does not allow selecting the activities according to the user's age.

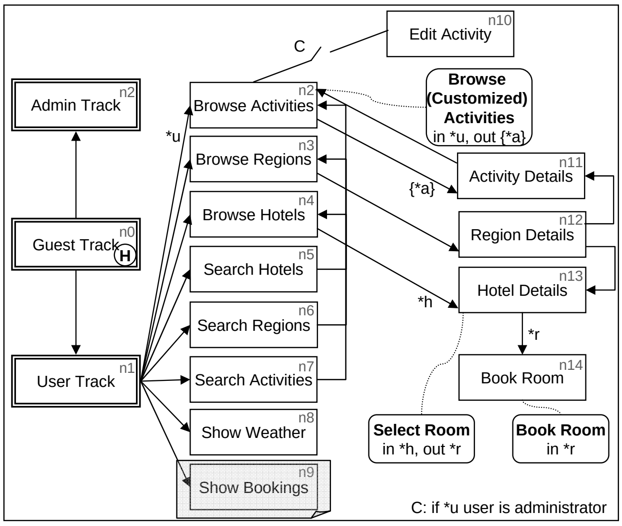


Figure 2.12: WSDM: The Navigation Model and the User Navigation Track

Customization Scenario Administrator Links. In Figure 2.12, the structure of the hypertext level is depicted in terms of the specific tracks for the audience classes. The *User Track* is presented in detail: nodes are connected with the necessary object chunks and links have attached parameters to be transported. Note that, only the object chunks previously presented in Figure 2.11 are depicted. With respect to the customization scenario *Administrator Links*, the WSDM notation allows specifying conditional links (cf. link to *Edit Activity* node), which is only available if the associated condition formulated in natural language turns out to be true (cf. the bottom-right corner in Figure 2.12).

Customization Scenario Special Offers. The *Special Offers* scenario can be realized on the basis of ASL. The users shall be provided with a link to the *Region Details* node, if their browsing behavior indicates a certain interest in the region. The ASL rule of Listing 2.4 specifies to provide a link from the *Activity Details* (n12) to the *Region Details* node (n11) if in more than 90% of the cases, the *Region Details* node is accessed by starting navigation through the hypertext from the *Activity Details* node and if the *Region Details* node is visited in 5% of all node accesses of the web site.

Listing 2.4: WSDM: Special Offers Scenario

```
if numberOfVisits (n12, ((n11,n2),(n2,n3),(n3,n12)))/numberOfVisits (n12) > 0,9 and
numberOfVisits(n12, ((n11,n2),(n2,n3),(n3,n12)))/totalNumberOfVisits > 0.05
then addLink((n11, n12))
```

2.2.4 The Object-Oriented Hypermedia Design Model (OOHDM), Rossi et al.

MATURITY

The *Object-Oriented Hypermedia Design Method* (OOHDM) [RS06] is amongst the earliest approaches to web application modeling and - dating back to 1994 [SR94] - the oldest approach included in this survey (G.T). The approach has been demonstrated by different modeling examples, such as an online catalogue for architecture, an online magazine, a CD store, a university department site, and a conference management system (G.ME). Apparently, two real-world applications of OOHDM exist: first, the portinari project (www.portinari.org.br), a web site presenting information on the painter Candido Portinari [SRB96], and second, an enterprise knowledge portal in the Brazilian subsidiary of the Xerox Corporation [SS02] (G.A).

WEB MODELING

The approach specifies a development process comprising five steps [RS06], namely *requirements gathering* on the basis of user interaction diagrams (UID) a refinement of the well known use cases technique [VSdS00], *conceptual design*, *navigational design*, *abstract interface design*, and *implementation* where the OOHDM-Java2 software architecture is proposed [JSR02] (W.Pr), (W.Ph). OOHDM supports all web application levels. The content level, in terms of the *conceptual model*, is represented as a UML-like class diagram (cf. Figure 2.13). It has to be noted, however, that due to the use of multi-typed attributes, the OOHDM approach is not fully conform to the UML [SR98] (W.L). At hypertext level, the *navigational class schema* (cf. Figure 2.16) is derived from the *conceptual model* and at the same time integrated from the *navigational contexts*[11]. The *navigational contexts* indicate possible navigation sequences to help users complete a task (cf. Figure 2.14). At the end, the *navigational class schema* represents a view on the content level in the form of a proprietary text-based query language - similar to view-definition approaches in object-oriented databases - is used to specify the mapping between content level and hypertext level concepts [SR98] (W.I). In the navigational class schema presented in Figure 2.16, the *Hotel* node encompasses the original *Hotel* class as well as the *Feature* class from the conceptual model. In the node *Person*, the attributes of the *Customer* and *Admin* classes have been integrated. An example of the textual specification for the *Hotel* node is given on the bottom of Figure 2.16. The *navigational class schema* again is represented as a UML class diagram, while the *navigational context schema* uses a proprietary notation

[11] Please note that OOHDM's notion of context is different to the notion of context in the realm of customization used in this thesis.

(W.L). For the presentation level OOHDM originally proposed the concept of *Abstract Data Views* (ADV) [CdL95] which represent abstract interfaces for navigational objects such as nodes, links, and access structures [RSdLC95] (W.L). More specifically, the approach made use of so-called *Configuration Diagrams* for specifying interface objects, their structural relationships, and the relationships among interface objects (ADVs) and navigational objects, e.g. nodes [RSdLC95] (W.I). Furthermore, the approach made also use of *Abstract Data View-Charts* (a statechart derivative) for expressing the behavior of an ADV, thus being the only way of behavioral modeling in OOHDM (W.F). Recently, however, the ADV approach seems to have been abandoned in favor of the *Abstract Widget Ontology (AWO)* [RS06] (W.L). The entire interface is specified by several ontologies using RDF and OWL as a formalism. The AWO proposes a set of concepts whose instances will comprise a given interface: *SimpleActivators* react to external events such as mouse clicks, *ElementExhibitors* exhibit a type of content such as text or images, etc. For lack of an explanation, the interface between hypertext and presentation level is assumed to be specified in natural language, only (W.I).

CUSTOMIZATION MODELING

Customization in OOHDM is considered during the design phase, only [RSG01], [SGR02]. OOHDM supposes that different users might have different tasks, access rights, and information needs. Thus, on the one hand, it is suggested to reuse the *conceptual model* by building different navigational views for different user profiles (C.G). Likewise, it is suggested to build different interfaces for different devices, though this possibility has never been exemplified (C.P). On the other hand, the approach proposes more fine-grained adaptation of content and hypertext levels according to the current user (C.G). In this respect, a ″user″ variable is introduced to be employed in the text-based query language for defining nodes and links from the *navigational class schema*, i.e., constraining them, as well as for defining *context classes* in the *navigational context schema*. This user variable is then ″mapped″ to an appropriate class in the *conceptual model*, e.g., a class ′Customer′. Consequently, the OOHDM approach does not provide for a separate context model (C.SC) and neither suggests ways for supporting further possibly complex context factors as well as context chronology (C.CE), (C.CC), (C.C). With respect to adaptation, the approach does not define adaptation operations (C.O), (C.AE), (C.CA) as already stated before, adaptations are expressed using OOHDM′s query language (C.SA). This language can be used to specify nodes and links (cf. Figure 2.16) as well as context classes (cf. Figure 2.14*(c)*) and thus, supports content adaptation as well as hypertext adaptation (C.L). Since the query language is used for defining the interface between content and hypertext, OOHDM also supports adaptation of the interface (C.I). Though support of presentation level adaptation is claimed [RSG01], [SGR02], a corresponding demonstration in terms of a modeling example seems to be missing (C.L).

MODEL-DRIVEN ENGINEERING

OOHDM is described as a set of models, which are built using object-oriented primitives with syntax close to UML [RSG01]. A language specification of OOHDM in terms of a metamodel, a grammar, or a semantic specification, however, is missing. Nevertheless, as already explained above, the interface model has recently been replaced by the Abstract Widget Ontology making use of RDF and OWL (M.L). There is no way of specifying model transformations in the OOHDM approach (M.T), neither are platform description models available (M.P). Still, OOHDM-Web [SdAPM99] is a web-based environment for OOHDM, which allows to specify an OOHDM design in XML. Given this specification, and a template page, it generates web pages for the CGILua environment [HBI98], thus implementing the specified application.

Modeling Example

Following, the OOHDM solutions to the customization scenarios *Customized Activities*, *Administrator Links*, and *Special Offers* are presented. Although, the approach suggests modeling different hypertext models for enabling device-independence, it is not clear, how the corresponding hypertext can be associated to the current user. Furthermore, the approach offers no mechanism for customizing the presentation level of web applications. Consequently, customization scenarios *Multi-Delivery* and *Season's Style* could not have been realized in OOHDM.

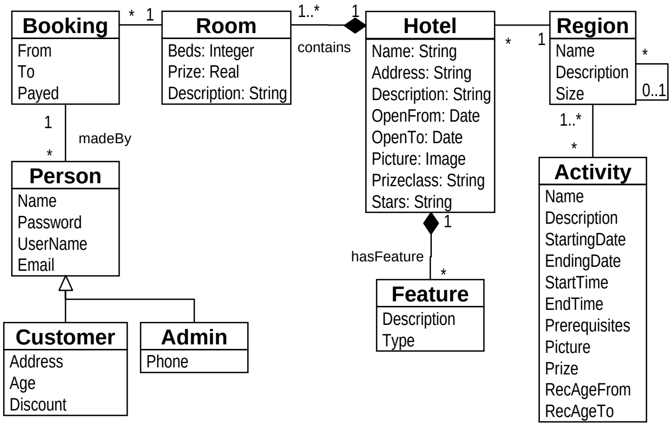
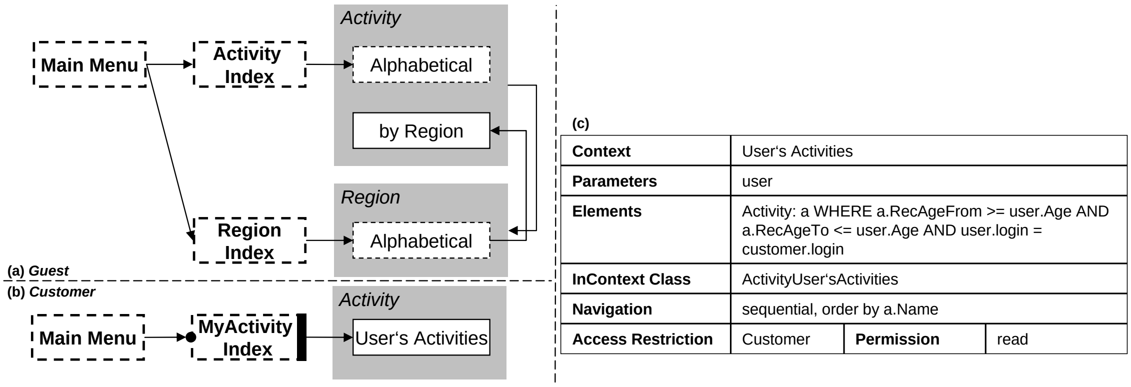


Figure 2.13: OOHDM: The Conceptual Model

The conceptual model of OOHDM is described with a UML class diagram (cf. Figure 2.13). Although possible in OOHDM, in this example we refrain from using multi-typed attributes. Please note the specialization of the *Person* class to the *Customer* and *Admin* classes, which will be used in the following customization scenarios for personalization purposes.

Customization Scenario Customized Activities. Figure 2.14 presents some of the *navigation context schemata* for the different tasks of different user types.



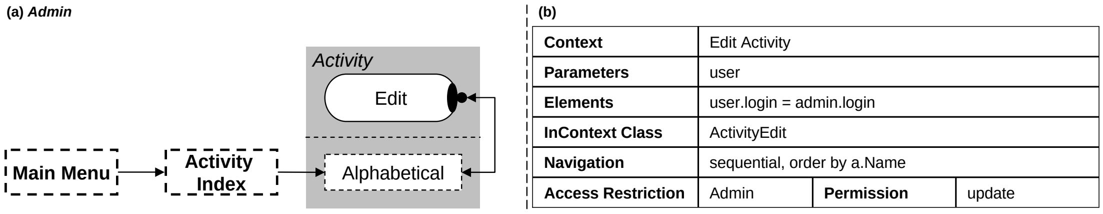
Context	User's Activities		
Parameters	user		
Elements	Activity: a WHERE a.RecAgeFrom >= user.Age AND a.RecAgeTo <= user.Age AND user.login = customer.login		
InContext Class	ActivityUser'sActivities		
Navigation	sequential, order by a.Name		
Access Restriction	Customer	**Permission**	read

Figure 2.14: OOHDM: Customized Activities Scenario

For example, a guest user will be able to navigate from a *Main Menu* to both an *Activity Index* and a *Region Index* (cf. Figure 2.14*(a)*). From these indices, s/he will be able to access the *Activity* and *Region* node (depicted as grey rectangles) in different contexts, e.g., alphabetically or sorted according to some criteria. Navigating from the *Region Index*, all regions will be displayed in alphabetical order. Having selected a region, the user can navigate to the *by Region* context of the *Activity* node and visit the activities of the region. With respect to the customization scenario *Customized Activities*, customers of the web application have an additional index, which requires the user to login (see the black circle for *MyActivityIndex* in Figure 2.14*(b)*). The *Activity* node then can be visited in the *User's Activities* context, which will display only activities that are appropriate for the current user's age. This filter condition as well as the access restriction is defined within the *Elements* section of the context specification card *User's Activities* in Figure 2.14*(c)*, where it is also tested if the user is of type *Customer*. Unfortunately, an explanation of OOHDM's user variable and a possible user model is missing. This example thus has been directly derived from examples given in [RSG01], [SGR02].

Customization Scenario Administrator Links.

OOHDM's solution to the *Administrator Links* scenario is shown in Figure 2.15*(a)*. The administrator will be able to navigate from the context showing the activities in an alphabetic order to the context for editing. Obviously, this time similar access restrictions to this context are needed, which are specified in the context specification card *Edit Activity* in Figure 2.15*(b)*. How a user is identified as administrator, however, cannot be expressed in OOHDM.



Context	Edit Activity		
Parameters	user		
Elements	user.login = admin.login		
InContext Class	ActivityEdit		
Navigation	sequential, order by a.Name		
Access Restriction	Admin	**Permission**	update

Figure 2.15: OOHDM: Administrator Links Scenario

Customization Scenario Special Offers.

For realizing the *Special Offers* scenario, the node *SpecialOffer* is introduced to the navigational class schema in Figure 2.16 and realizes a content adaptation, since the prize of a room is adapted according to the user's discount. Nevertheless, it is not possible to model that this discount shall be given according to the user's booking history and thus it is not possible to fully model customization scenario *Special Offers*. On the bottom of Figure 2.16, the full specification of the *SpecialOffer* node is presented in terms of OOHDM's textual query language. This mechanism can also be used to define several navigational class schemata for different types of users with different access rights. For example, for guest users there will be no need for information on booking or special offers. Furthermore, the query language allows for link personalization. For example, on the bottom of Figure 2.16, the specification of the link *yourBookings* can be found. This link will be displayed on the homepage and will include the bookings of the current user, only. To do so, the user variable is used, which is mapped to the *Customer* class.

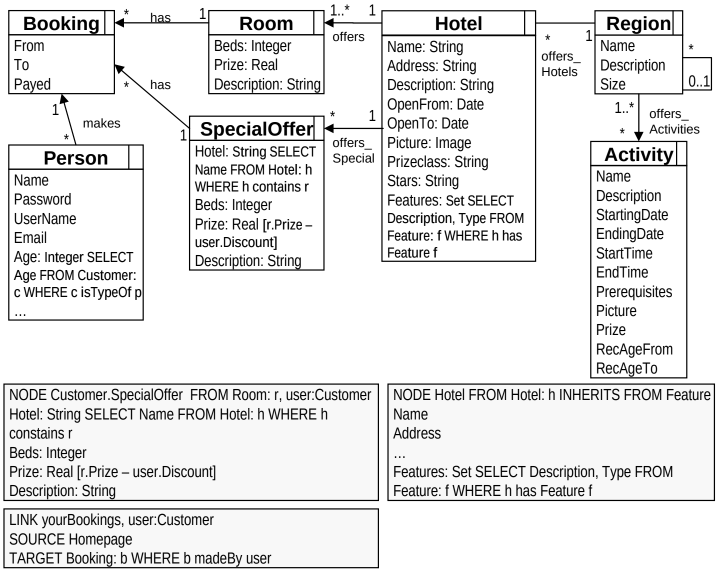


Figure 2.16: OOHDM: Navigational Class Schema

2.2.5 The UML-based Web Engineering Approach (UWE), Koch et al.

Maturity

The *UML-based Web Engineering*[12] (UWE) approach [Koc01], [KK02a] has been continuously developed since 1998/2000. Recent work heads towards model-driven development of web applications [KZC06],[Koc07] based on model transformation techniques and MDA [OMG03] standards (G.T). The approach has been illustrated using several modeling examples, amongst them a music portal, an online library, a conference management system, and an e-store (G.ME). Nevertheless, its application in real-world projects has not yet been reported (G.A).

Web Modeling

The approach supports UML models for content (*conceptual model*), hypertext (*navigation space model* and *navigation structure model*), and presentation levels (*presentation model*) (W.L). Structural modeling is based on class diagrams at all levels, i.e., special stereotypes for the specific web concepts are introduced. While at hypertext level, state chart diagrams are used to model navigation scenarios, at presentation level, sequence diagrams can be employed to depict presentation flows (W.F). The interface between content and hypertext levels is described in terms of OCL. This means each attribute of a ≪navigation class≫, i.e., a hypertext node, in the *navigation space model* is derived from information stored in the *conceptual model* (W.I). The *navigation structure model* is an "extension" of the navigation space model and describes how navigation is supported by access elements such as *menus*, *indices*, *guided tour*, etc. For each *navigational class* of the navigation space model, at the presentation level a separate model is created, describing where and how the navigation primitives will be presented to the user. The UML composition notation for classes is used together with a set of stereotypes for the nested elements [KK02a], e.g., ≪text≫, ≪form≫, ≪button≫, and ≪image≫ [BKM99]. Furthermore, UWE provides natural language guidelines to infer the navigation space model from the conceptual model and to infer the pre-

[12] http://www.pst.ifi.lmu.de/projekte/uwe/

sentation model from the navigation space model. These guidelines also form the basis for model transformation within the tool ArgoUWE. UWE has a well-defined development process based on RUP [Kru00] (W.Pr) and provides explicit support for RUP's 5 phases including for each phase the workflows requirements engineering (use-case diagram), analysis (conceptual model, navigation space model), and design (presentation model and refinement of all models) (W.Ph) [Koc01].

CUSTOMIZATION MODELING

Customization modeling support within UWE is mainly considered during analysis and design phases (C.CP) and has its origins in the Munich Reference Model [KW02]. At content level, Koch et al. distinguish between a *domain model* and a *user model* capturing contextual information about the user. Nevertheless, relationships between classes from both models allow for partial separation of context, only (C.SC). Besides the *user model*, an *adaptation model* is suggested, which is realized by means of a UML communication diagram and describes of a set of condition-action rules (i.e., *construction* rules, *adaptation* rules, and *acquisition* rules), the rules' temporal relationships as well as when they are triggered. Still, the rules themselves are described in natural language, only. More recently, the UWE approach suggests concepts for modeling context which are separated into a *User* package and an *Environment* package in the UWE metamodel presented in [KK03] (C.P). The details of those packages, however, have not been illuminated yet and corresponding modeling examples do not seem to be available either. No indication of either UWE's extensibility with respect to further context properties (C.CE), nor the support of complex context (C.C) and context chronology (C.CC) is given. Furthermore, another proposal for customization modeling at navigation level has been made (C.L), (C.I), which supports link annotation, link hiding, and link generation (C.O) [BKKZ05]. The proposal extends the UWE metamodel with concepts from aspect-oriented modeling (cf. Chapter 3) and adaptations are provided in terms of *aspects*. The *LinkAspect*, *LinkAnnotationAspect*, *LinkTraversalAspect* and *LinkTransformationAspect* are modeled as sub-classes of UML packages. Thus, further adaptations could only be supported by extending the UWE language itself, i.e. through sub-classing within the UWE metamodel. (C.AE). The adaptations can be separate with aspects from the rest of the application (C.SA). The granularity of adaptations - depending on the *aspects' pointcuts* - ranges from micro to macro (C.G). A mechanism for specifying complex adaptations is not available, however (C.CA).

MODEL-DRIVEN ENGINEERING

The UWE language is defined as an extension of the UML metamodel [KK03] and additionally provides a UML profile for interoperability purposes (M.L). The guidelines for generating preliminary models, e.g., the navigation space model, from models created during earlier development phases have been automated within the tool support. Furthermore, transformation of requirements specified in terms of use case diagrams and activity diagrams into UWE models at content and hypertext level based on MOF [OMG02] and QVT [OMG05a] has been recently proposed in [KZC06]. This approach, however, has not yet been automated (M.T). UWEXML represents the proprietary code generation framework prototype and describes PIM2Code transformations based on XSLT to J2EE and Cocoon platforms [KK02b]. Separate platform description models do not exist (M.P).

TOOL SUPPORT

ArgoUWE is the free tool support for UWE (T.C). It is a proprietary extension of the UML modeling tool *ArgoUML*[13] (T.B), (T.O). ArgoUWE 0.16 provides general modeling support (T.V) but customization modeling has not yet been realized within the tool (T.M). During modeling, the tool checks the artifacts for errors as well as offers help to resolve the identified problems with a

[13] http://argouml.tigris.org/

wizard (T.CC). Moreover, the tool allows for semi-automatically generating the navigation space model from the conceptual model, as well as the presentation model from the navigation space model (T.T). With respect to the presentation model, ArgoUWE does not yet support the composition notation of classes but uses composition associations to connect the attributes of the navigation class to the owning presentation class. UWE'S development process, however, is not realized within the tool, since the typical iterations are not supported without loosing information, e.g., changing the conceptual model without loosing information from the hypertext level is not possible (T.P). Code generation (T.CG) is currently not supported, neither is there versioning support to allowing for shared editing (T.Co).

MODELING EXAMPLE

In the following, UWE's customization approach as described in [BKKZ05] is presented. Currently, the approach is restricted to hypertext level adaptations. Thus, it is only possible to model the customization scenario *Administrator Links*.

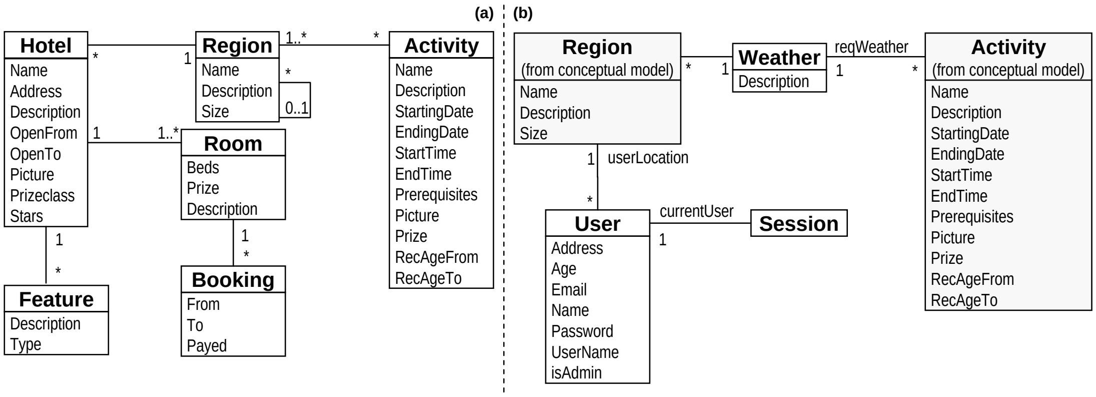


Figure 2.17: UWE: (a) Conceptual Model, (b) User Model

Customization Scenario Administrator Links. The content level is depicted in Figure 2.17 in terms of the conceptual model and the user model. In the user model, context information about the user is captured. Users belonging to the administrator group are indicated with the *isAdmin* attribute set to true. As can be seen in Figure 2.17*(b)*, the user model is connected to the conceptual model via associations to its concepts, i.e., *Region* and *Activity*. It is assumed that the application will be able to detect the user's location and resolve it to a certain region, i.e., the user model will have the necessary data. Furthermore, the user model captures the current weather of a region and activities can be selected according to their required weather situation. Part of the navigation structure diagram is depicted in Figure 2.18*(a)*, which shows that indices of hotels, regions, and activities can be reached from the main menu of the TIWA. From the indices the hotels, regions and activities can be viewed in detail. In particular, the *RegionDetails* navigation class offers to browse the region's hotels and activities via two further dedicated indices *HotelsByRegion* and *ActivitiesByRegion*, respectively. Customization functionality can then be introduced to the hypertext level using aspects. Each aspect has exactly one *pointcut* and one *advice*. According to customization scenario *Administrator Links*, edit links shall be provided from the *ActivityDetails* navigation class to the *EditActivity* navigation class if the user is an administrator.

The ≪link transformation aspect≫ of Figure 2.18*(b)* specifies that some link transformation needs to be performed. More specifically, the ≪pointcut≫ section defines where the adaptation takes place, i.e., the participating navigation classes *ActivityDetails* and *EditActivity* are identified. Within the aspect, any information of the current session can be queried using OCL via 'thisSession', which is supposed to be able to identify the current session the TIWA is responding to. Thus, an additional OCL constraint specifies that the adaptation can only take place for a certain context. It constrains the adaptation to users that are administrators. In the ≪advice≫ section the necessary adaptation is depicted, i.e., a link is introduced from the *ActivityDetails* navigation class to the *EditActivity* navigation class. Additionally, UWE supports *link traversal aspects*, which allow executing an action when a link is traversed (e.g., updating the user model), and *link annotation aspects*, which allow adding information to a link (e.g., how many times it is traversed).

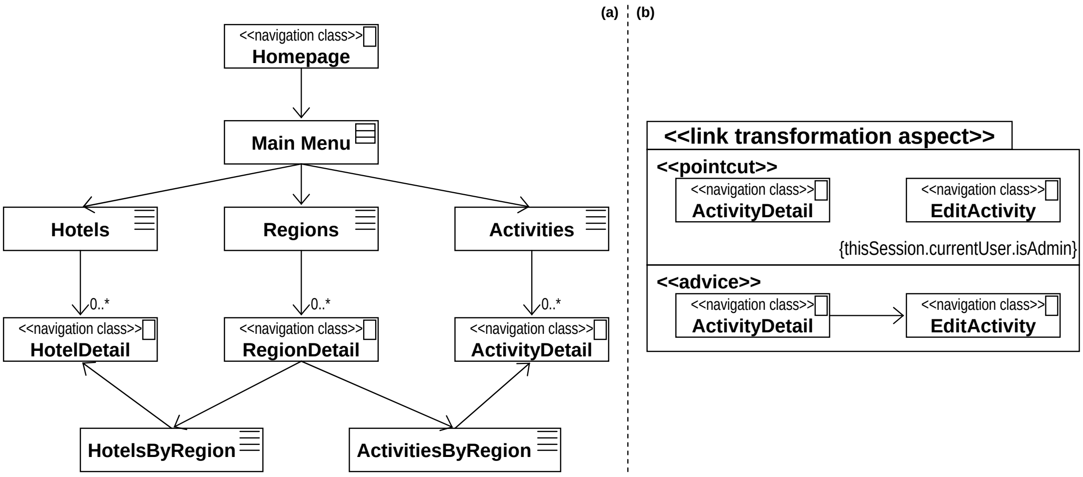


Figure 2.18: UWE: (a) Navigation Structure Model, (b) Administrator Links Scenario

2.2.6 The Object-Oriented Hypermedia Method (OO-H), Gomez et al.

MATURITY

The *Object Oriented Hypermedia* (OO-H) Method [GCP01], [GCP00] first emerged in 2000, formerly as an extension of the OO-Method [PIP$^+$97], a design method for object-oriented systems. The most recent publications deal with personalization of web sites [GCG05], [GCG07] (G.T). The approach is demonstrated using multiple examples including amongst others a conference management system, a library system, a forum application, an e-store application, a hotel reservation system, and an e-mail system. (G.ME). Moreover the OO-H method and its tool support in terms of the *VisualWade* CASE tool[14] have been applied in the development of several real-world applications as is reported in [GBP05] (G.A).

WEB MODELING

The OO-H method is based on UML. It defines its own process for the design phase, while other phases in the development lifecycle are described with guidelines, only (W.Pr) [GCP01].

[14] http://www.visualwade.com

The approach supports all phases from requirements engineering with use-case diagrams to implementation with the methods tool support [CGPP01] (W.Ph). The approach comprises different models for the content, hypertext and presentation level (W.L): Standard UML class diagrams are used for content modeling. With the introduction of UML activity diagrams for modeling processes [KKCM04], the OO-H method also supports behavioral modeling at the content level (W.F). At hypertext level, so called *navigation access diagrams* (NAD) are associated with each user type, capturing the navigation paths and the services (from the content level) the users can activate (cf. Figure 2.20). They consist of *navigation classes* which represent views on the *conceptual classes* from the content level. The interface is specified graphically, i.e., the "label" of a navigation class consists of its name separated with a colon from the name of the underlying conceptual class (cf. Figure 2.20) (W.I). Navigation classes are associated with each other through different types of navigation links, which may have both a *navigation pattern* from the OO-H pattern catalogue [GCP01] (e.g., *Index*, *GuidedTour*) and a set of OCL-like *navigation filters*, associated. Different types of *collections* represent (possibly hierarchical) access structures defined on navigation classes or *navigation targets* and may also have both a navigation pattern and a set of navigation filters associated. Additionally, these concepts can be grouped within navigation targets in order to cover a certain user navigation requirement. *Mapping rules* allow inferring default navigation access diagrams from activity diagrams, which are used for modeling processes [KKCM04]. At presentation level, starting from a NAD, a default *abstract presentation diagram* (APD) [CGP00] can be generated, by using so called *NAD2APD mapping rules* [GCP01] (W.I). An APD can be interpreted as the sitemap of the web application consisting of a set of "abstract pages" associated with links and its modeling concepts are defined in several DTDs. The default APD can be refined through patterns, defined as *transformation rules* [GCP01], which are implemented in Python. Furthermore, the OO-H CASE tool also includes the *composite layout diagram* (CDL), which allows further refinement of the user interface (i.e., the XML specification of the APD) in the manner of a WYSIWYG editor [CGP01].

Customization Modeling

The OO-H method suggests a personalization framework in the form of a UML class diagram [GGC03b], [GCG05] comprising a *user model* and a *personalization model*: The *user model* actually extends the content level (C.SC) and allows capturing context information with respect to user, location, device, time, and network (C.P). Further context properties can be introduced through inheritance from a generic *Context* class [GCG05] (C.CE). The framework currently does not foresee complex contexts or a concrete mechanism for modeling a change in context over time (C.CC) (C.C). The personalization model consists of a set of ECA rules, which realize different personalization strategies: First, *acquisition rules* define how context information is acquired. Second, *personalization rules*, which belong to a certain profile, define the adaptation that is to be made. A profile encompasses a set of personalization rules supporting a group of users with similar needs, e.g., users with small screen devices. And third, *profile rules* associate a user to a profile. Personalization rules are further distinguished into rules manipulating the content, the hypertext, and the presentation level (C.L), (C.I). The existing set of concrete rules (C.O) includes fine-grained adaptations such as *Show* [GGBH05] (used to show attributes in the *Customized Activities* scenario in Listing 2.6 and to show a link in the *Special Offers* scenario in Listing 2.7) but also coarse-grained adaptations (cf. *SetCSSTemplate* [GGC03b] used in the *Season's Style* scenario in Listing 2.13) (C.G) The set of rules can be extended again through inheritance from the generic rule classes (i.e., *ContentRule*, *NavigationRule*, and *PresentationRule*) [GGC03b] (C.AE). Currently, the combination of adaptations to form complex adaptations is not considered (C.CA), however. The ECA rules are specified separate from the other models (C.SA) following the syntax of the EBNF-based *Personalization Rule Modelling Language* (PRML), which can be interpreted by a rule engine at runtime [GGC03a]. The use of this rule language, however, means that any extension of existing adaptation operations has to be done to this language (C.AE). Customization is considered in the design and the implementation phase of the development process (C.CP).

Model-driven Engineering

The OO-H language definition for the content and hypertext level is realized as an extension of UML and OCL [CGP01]. Still, the concepts of the presentation level, i.e., the APD, are defined as a set of DTDs and a separate language for the specification of customization is provided with the EBNF-based PRML (M.L). The OO-H approach provides mapping rules from activity diagrams to NADs and from NADs to APDs. As mentioned before there is tool support for the OO-H method, which implements the NAD to APD transformations (PIM2PIM) as well as modeling PIM2Code transformations (M.T). The approach, however, does not describe platform models (M.P).

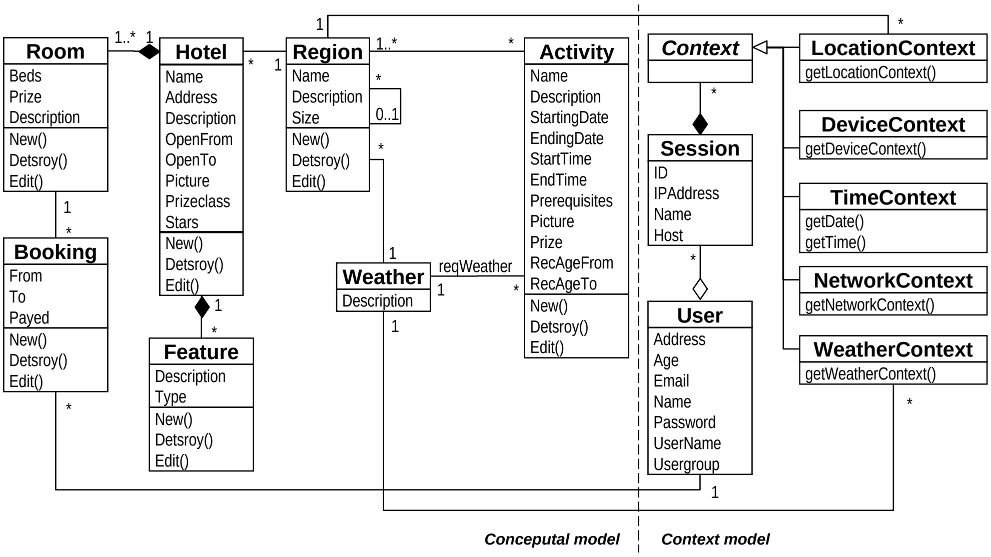


Figure 2.19: OO-H: Conceptual Model and Context model

Tool Support

The current version of the *VisualWade* tool is 1.2 (T.V). VisualWade is a commercial tool, but offers a trial version (T.B), (T.O), (T.C).It allows modeling in general but does not support customization modeling. Apart from VisualWade, the generation of UWAs on the basis of the PRML language is recently provided by a prototype described in [GCG07]. In this work, the OO-H approach has evolved to the distinct *Adaptive* OO-H approach, (T.M). The OO-H method's process is supported only partly within VisualWade, e.g., no use-cases for the requirements definition phase are available. Apart from that, the conceptual model, the NAD as well as the CLD are supported, as mentioned before. The APD actually is not used explicitly but indirectly via the CDL. The user will start from a conceptual model, via the NAD to the CLD, but can go back to previous models for changes. Changing the NAD, however, means loosing the presentation model which has to be generated anew (T.P). VisualWade supports PIM2PIM and PIM2Code transformations (T.MG) as well as provides an OCL-based action language for transformations and code generation capabilities [GBP05]. With respect to code generation, VisualWade currently supports one model compiler that generates code for the PHP target platform. The application can be generated in several independent steps such that the application logic and the database, for example, can be created in random order (T.CG). The modeled artifacts can be statically checked on demand and the user is provided with hints indicating how to solve the detected problems (T.CC). With respect to collaborative development of web applications, VisualWade currently provides no specific support (T.Co).

Modeling Example

OO-H's PRML allows for modeling all customization scenarios. Before presenting the individual customization scenarios in terms of PRML rules, however, the content and hypertext level as well as the context model of the web application are presented.

The content level of the TIWA is extended with the *context model* (cf. Figure 2.19). More specifically, the given context framework has been extended with a *WeatherContext* class in order to be able to consider the weather context factor. Furthermore, the conceptual classes of the content level have been extended with operations for creating, updating, and deleting objects of the specific types.

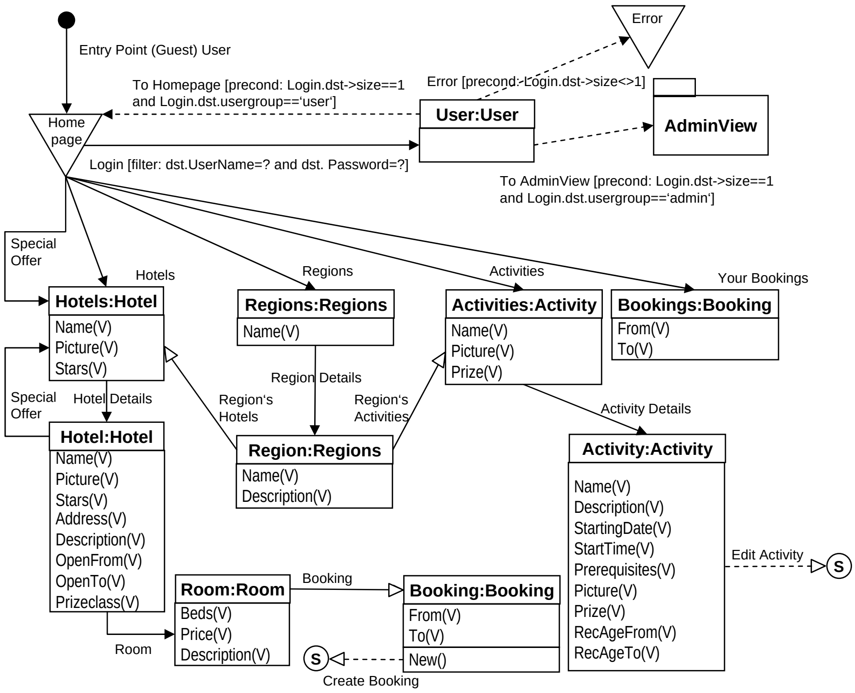
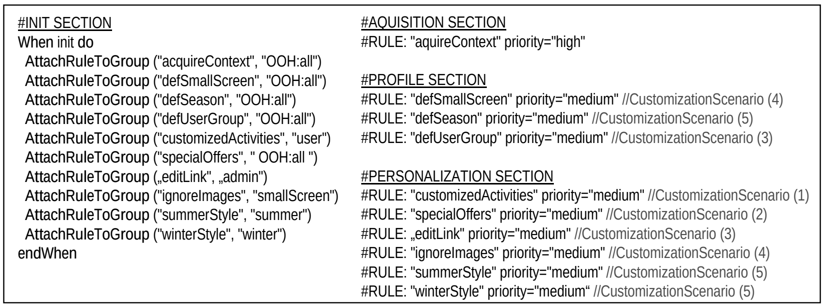


Figure 2.20: OO-H: Navigation Access Diagram

Figure 2.20 presents part of the hypertext level. In this case, one NAD has been designed for guests and registered users of the TIWA. From the homepage user will have to login to the TIWA and will be redirected to the *Homepage* menu if they belong to the user group 'user' (cf. *To Homepage* link). The links supporting the login process have attached OCL-like *preconditions* and *navigation filters*, which will be ignored for the remaining links for readability purposes. A separate view for administrators is indicated with the navigational target *AdminView* but not further de-



```
#INIT SECTION
When init do
  AttachRuleToGroup ("acquireContext", "OOH:all")
  AttachRuleToGroup ("defSmallScreen", "OOH:all")
  AttachRuleToGroup ("defSeason", "OOH:all")
  AttachRuleToGroup ("defUserGroup", "OOH:all")
  AttachRuleToGroup ("customizedActivities", "user")
  AttachRuleToGroup ("specialOffers", " OOH:all ")
  AttachRuleToGroup („editLink", „admin")
  AttachRuleToGroup ("ignoreImages", "smallScreen")
  AttachRuleToGroup ("summerStyle", "summer")
  AttachRuleToGroup ("winterStyle", "winter")
endWhen

#AQUISITION SECTION
#RULE: "aquireContext" priority="high"

#PROFILE SECTION
#RULE: "defSmallScreen" priority="medium" //CustomizationScenario (4)
#RULE: "defSeason" priority="medium" //CustomizationScenario (5)
#RULE: "defUserGroup" priority="medium" //CustomizationScenario (3)

#PERSONALIZATION SECTION
#RULE: "customizedActivities" priority="medium" //CustomizationScenario (1)
#RULE: "specialOffers" priority="medium" //CustomizationScenario (2)
#RULE: „editLink" priority="medium" //CustomizationScenario (3)
#RULE: "ignoreImages" priority="medium" //CustomizationScenario (4)
#RULE: "summerStyle" priority="medium" //CustomizationScenario (5)
#RULE: "winterStyle" priority="medium“ //CustomizationScenario (5)
```

Figure 2.21: OO-H: Personalization Rules

tailed. If the user belongs to the usergroup 'admin', s/he will be redirected to the *AdminView* (cf. *To AdminView* link). In case the input retrieves no dataset, i.e., no user account can be found, the user is redirected to an error page (cf. *Error* link). From the *Homepage* menu the user can navigate, for example, to the *Hotels* navigation class which will present an index of hotels, as is indicated by the second part of the navigation class label which denotes the corresponding conceptual class from the content level. Likewise, the user can browse regions and activities and visit the navigational classes presenting an object of interest, e.g., the *Hotel* navigation class provides detailed information on one hotel. For those links that are not accessible for guests, e.g., *Create Booking*, *Your Bookings*, and *Edit Activity* special rules need to be designed in PRML such as has been done for the *Edit Activity* link in the customization scenario *Administrator Links*. The *Edit Activity* link and the *Create Booking* link are *service links* that call the Edit() and New() operations of the *Activity* and *Booking* conceptual class when the user books a room or the administrator edits an activity, respectively.

As already stated above all customization scenarios can be realized with PRML. Figure 2.21 outlines the realization of the customization scenarios on the basis of ten PRML rules: one acquisition rule, three profile rules, and six personalization rules. All of them need to be initialized (cf. init section in Figure 2.21) meaning that all rules will be associated to a profile group and thus, to the users of that group. The default group 'OOH:all' applies to all users.

In the acquisition section of Figure 2.21, when a new session starts the *aquireContext* rule will collect the necessary context information to be available in other rules, i.e., the device used to access the TIWA as well as the current date:

Listing 2.5: OO-H: Context Aquisition Rule

```
#RULE: "aquireContext" priority="high"
When SessionStart do
  deviceContext=UM.DeviceContext.getDeviceContext()
  date=UM.TimeContext.getDate()
endWhen
```

Also at session start, the user will be attached to one or more profile groups (cf. profile section in Figure 2.21). In the personalization section, the actual rules for adapting content, hypertext, and presentation of the TIWA are specified. Following, the profile rules as well as the personalization rules will be explained for the individual customization scenarios.

Customization Scenario Customized Activities. Rule *customizedActivities* (cf. Listing 2.6) is applied if the user navigates from the *Homepage* to the *Activities* node. Depending on the user's age, location, the current date and time, as well as the current weather, the user will be presented a customized selection of activities. In this respect, the adaptation functionality is encapsulated in *Show*.

Listing 2.6: OO-H: Customized Activities Scenario

```
#RULE: "customizedActivities" priority="medium"
When Navigation(Activities(Activity)) do
  If (UM.User.Age >= NM.Activity.RecAgeFrom and
     UM.User.Age <= NM.Activity.RecAgeTo and
     UM.LocationContext.getLocationContext = NM.Activity.region and
     UM.WeatherContext.getWeatherContext = NM.Activity.reqWeather and
      date < NM.Activity.StartingDate and
     UM.TimeContext.getTime() < NM.Activity.StartTime)
  then
     Show(Activity.Name, Activity.Description, Activity.Prize)
  endIf
endWhen
```

Customization Scenario Special Offers. If the user views 3 hotels of the same region for at least 60 second (cf. LoadSet event [GG06]), the user will be presented the link *Special Offers* to the *Hotels* node. In this node, special offers for the specific region will be presented (cf. Listing lst:ch2:OOHSpecialOffers). This is specified with rule *specialOffers*:

Listing 2.7: OO-H: Special Offers Scenario

```
#RULE: "specialOffers" priority="medium"
When LoadSet[Hotel(NM.Region reg, 3, 60)] do
  Show(Special Offers)
endWhen
```

Customization Scenario Administrator Links. The profile rule *defUserGroup* (cf. Listing 2.8) identifies the user type of the current user, i.e., if the user has logged in s/he will be associated to either the *user* or the *admin* profile group defined in the init section in Figure 2.21 with *AttachUserToPGroup* [GCG05]. The OO-H approach assumes that the current user group can be queried from the user model [GG06].

Listing 2.8: OO-H: Administrator Links Scenario (1)

```
#RULE: "defUserGroup" priority="medium"
When SessionStart do
  If (UM.User.Usergroup="admin" then AttachUserToPGroup("admin")
  endIf
  If (UM.User.Usergroup="user" then AttachUserToPGroup("user")
  endIf
endWhen
```

The *editLink* personalization rule specified in Listing 2.9 is associated to the *admin* profile group (cf. init section Figure 2.21) and ensures that the service link *Edit Activity* will be available to administrators but not to registered users or guests of the TIWA:

Listing 2.9: OO-H: Administrator Links Scenario (2)

```
#RULE: "editLink" priority="medium"
When LoadElement(Activity) do
  Show(Edit Activity)
endWhen
```

Customization Scenario Multi-Delivery. The *defSmallScreen* profile rule in Listing 2.10 associates the user with the *smallScreen* group defined in the init section in Figure 2.21, if the device used to access the TIWA is either a PDA or a mobile phone (cf. *deviceContext* defined in the *contextAquisition* rule of Listing 2.5).

Listing 2.10: OO-H: Multi-Delivery Scenario (1)

```
#RULE: "defSmallScreen" priority="medium"
When SessionStart do
  If (deviceContext="PDA" or deviceContext="Mobile") then AttachUserToPGroup("smallScreen")
  endIf
endWhen
```

Multi-delivery can be realized also at a fine-grained level. If a PDA or a mobile phone is used to access the TIWA, pictures shall be omitted. The rule *ignoreImages* (cf. Listing 2.11)applies to the *Hotel* and *Activity* nodes. As is defined by the above profile rule, it is attached to *smallScreen* group.

Listing 2.11: OO-H: Multi-Delivery Scenario (2)

```
#RULE: "ignoreImages" priority="medium"
When LoadSet[Hotel (NM.Hotel h)|| Activity (NM.Activity a)]
do
  h.picture.Visible=false
  a.picture.Visible=false
endWhen
```

Customization Scenario Season's Style. Finally, the *summerStyle* and *winterStyle* personalization rules change the CSS used to the ones specified. The rules are attached to the *summer* and *winter* profile groups, respectively (cf. init section Figure 2.21) and will cause the presentation to be adapted depending on the current season. According to the *defSeason* profile rule, the user will either be attached to the *summer* or the *winter* group depending on the current date:

Listing 2.12: OO-H: Season's Style Scenario (1)

```
#RULE: "defSeason" priority="medium"
When SessionStart do
  If (date > 21.06. and date < 21.12) then AttachUserToPGroup("summer")
  endIf
  If (date > 21.12. and date < 21.06) then AttachUserToPGroup("winter")
  endIf
endWhen
```

Depending on the group the user is associated with, one of the *summerStyle* and *winterStyle* personalization rules will be triggered and define an appropriate CSS style sheet to be used.

Listing 2.13: OO-H: Season's Style Scenario (2)

```
#RULE: "summerStyle" priority="medium"
When SessionStart  do
  SetCSSTemplate("summer.css")
endWhen

#RULE: "winterStyle" priority="medium"
When SessionStart  do
  SetCSSTemplate("winter.css")
endWhen
```

2.2.7 The Object-Oriented Web Solution Approach (OOWS), Pastor et al.

Maturity

Object Oriented Web Solutions (OOWS) [PFPA06] is an approach that was first proposed in 2000 [PAF00] by extending OO-Method [PIP$^+$97] to support web modeling. Unlike the OO-H approach, OOWS is still based on the OO-Method. Recently, first results of the implementation of tool support for the OOWS approach have been published (G.T). The approach is presented with modeling examples like a university department site, a ticket ordering system, a book store, a travel agency system, and a university research group management system (G.ME). The university department site has been realized and can be visited at *www.dsic.upv.es* (G.A).

Web Modeling

The OOWS method supports all software development phases including a so-called *solution development* phase in which models are translated into code (W.Ph). Developers are provided with appropriate modeling means in each of OOWS's proprietary 4-step development process (W.Pr). For the requirements engineering phase use-case diagrams, scenarios, and task descriptions are used [VFP05], [TPRV05]. Task descriptions comprise task taxonomies, whereby each elementary task is described with an UML activity diagram, as well as information templates which are based on Class-Responsibility-Collaboration (CRC) cards. Thereafter a *conceptual schema* is built in the analysis phase. Since the approach is based on OO-Method, only the models of the hypertext and presentation level are special to OOWS. In the design phase different kinds of models are used (cf. [FPAP03]) (W.Ph): On the content level the information is captured using the *structural model*, i.e., a UML-like class diagram, just like in OO-Method. In addition, UML state and sequence diagrams can be used to describe behavioral aspects and represent the *dynamic model* in OOWS (W.L). A *functional model* describes service effects in a textual formal specification language which is based on the OASIS formal language[15] as is described in [PFPA06]. A *navigational model* comprises *navigational maps* which are used to define the global and structural aspects of the navigation, i.e., how so-called *navigation contexts* are related via navigational links with each other. Apart from that, the navigation contexts can be modeled using the *navigation context diagram* which allows designing the white-box view of a *navigation context* (cf. Figure 2.24*(b)*). The *navigational contexts* comprise *navigational classes* which are views on the content level. The interface between content and hypertext level is graphically specified. The hypertext level and the presentation level are intermingled though: A presentation model only enriches the ”in-the-small” model of the navigational context with presentational patterns concerning aspects such as scrolling or ordering of information (W.I). As already mentioned, only the content and the hypertext level can be enriched with behavioral diagrams while on all three levels structural modeling is supported (W.F).

Customization Modeling

The only context properties addressed in OOWS literature seems to be the user property. In [AFGP02], more possible context properties are mentioned but they would have to be introduced by means of the standard OOWS models (C.P), (C.CE). Apart from that, no mechanisms for chronology (C.C) or complex context (C.CC) are available and since the context properties have to be included in the content level, there is neither separation of context (C.SC). Nevertheless, OOWS allows the definition of adaptations very early in the design process, namely during requirements specification [RVP06], i.e., within activity diagrams, until the design phase, i.e., within *navigational contexts* (C.CP). In particular, the different user types can be incorporated into the content model by merging the base content model with a *user stereotype model*. Furthermore, when specifying

[15] http://www.oasis-open.org/specs/index.php

tasks by means of activity diagrams, the modeler can define adaptation rules such as link or content hiding as well as sorting of information according to previous user behavior (C.O). These adaptation operations are based on OCL conditions and are limited to fine-grained adaptations such as accessibility, filter, or sorting conditions (C.AE), (C.G). Complex adaptations cannot be realized (C.CA). The adaptation operations can be used to influence the hypertext level, only (C.L), (C.I). Moreover, customization in the OOWS approach requires the adaptations to be integrated into the models, allowing no separation of adaptation (C.SA).

Model-driven Engineering

Recently, the OOWS language has been defined as a MOF-based metamodel and tool support on the basis of the Eclipse Graphical Modeling Framework (GMF)[16] is under development [VVFP07]. Also a definition of the language in OWL has been proposed [TFPP04] (M.L). OOWS supports an MDA-based approach including the transformation of platform-independent models. The possibility to generate code for the front-end of a web application is also currently under development within the above mentioned tool support (M.T). In order to generate the back-end of a web application the commercial tool OlivaNova is used. In the future, the integration of the code generation for the back-end and the front-end is planned. A platform description model does not seem to be used, however (M.P).

Modeling Example

In the following, three customization scenarios are presented, namely *Customized Activities*, *Administrator Links*, and *Special Offers*. Concerning the customization scenarios *Multi-Delivery* and *Season's Style*, it is not possible to cope with them in OOWS due to the fact that only the user context can be exploited for adaptation purposes.

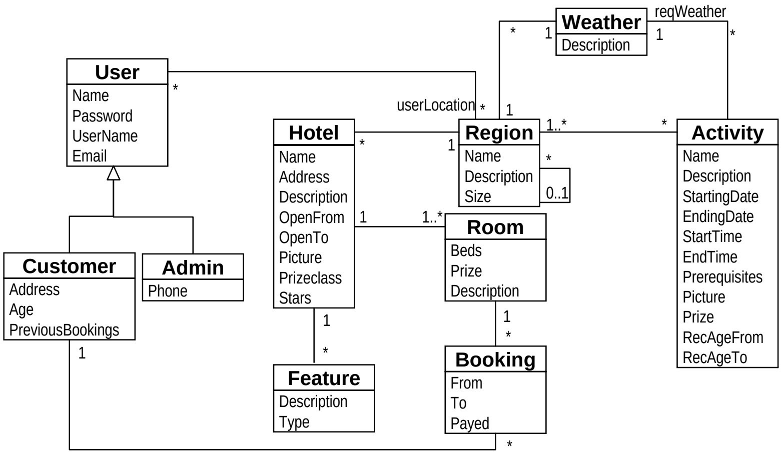


Figure 2.22: OOWS: The Structural Model

Figure 2.22 shows the example's underlying class diagram, in which the different user stereotypes (i.e., *Customer* and *Admin* classes) are already integrated with the structural model of the TIWA. The *User* class and its sub-classes represent a hierarchy of possible user stereotypes of the TIWA. Following, for each scenario, the diagrams which are most relevant for customization in the OOWS approach are shown. These are, on the one hand, the activity diagrams from the re-

[16] www.eclipse.org/gmf

quirements phase and, on the other hand, the resulting navigational models, in particular the navigational contexts, to present how the adaptation rules are incorporated into the hypertext.

Customization Scenario Customized Activities. This scenario is about adaptations filtering information from the content level meaning, that a user currently logged in, should only see activities which are appropriate for her/him. The scenario is explained starting from the requirements specification, in particular from the activity diagram shown in Figure 2.23*(a)*. In the first action the user has to select one region from an index of regions. After selection, it must be verified by the system if the user is logged in or if s/he is an anonymous user. If the user is logged in (cf. action *2:Activity*), the following precondition can ensure that only appropriate activities are shown:

```
self.recAgeFrom > #user#.age and self.recAgeTo < #user#.age
```

Using the keyword `self`, it is possible to access objects which should be displayed in this UML action (in this case, objects of the *Activity* class). With an OCL-like point-notation one can navigate through the structural model and retrieve attribute values of them to be compared with values from the current context. One can access attribute values of the user with the #user# variable representing the current context. If the user is not logged in (cf. action *3:Activity*), all activities of the selected region are shown, i.e., no personalized content is provided to the user. Finally, the user can select one of the activities to navigate to a detailed view of the selected activity.

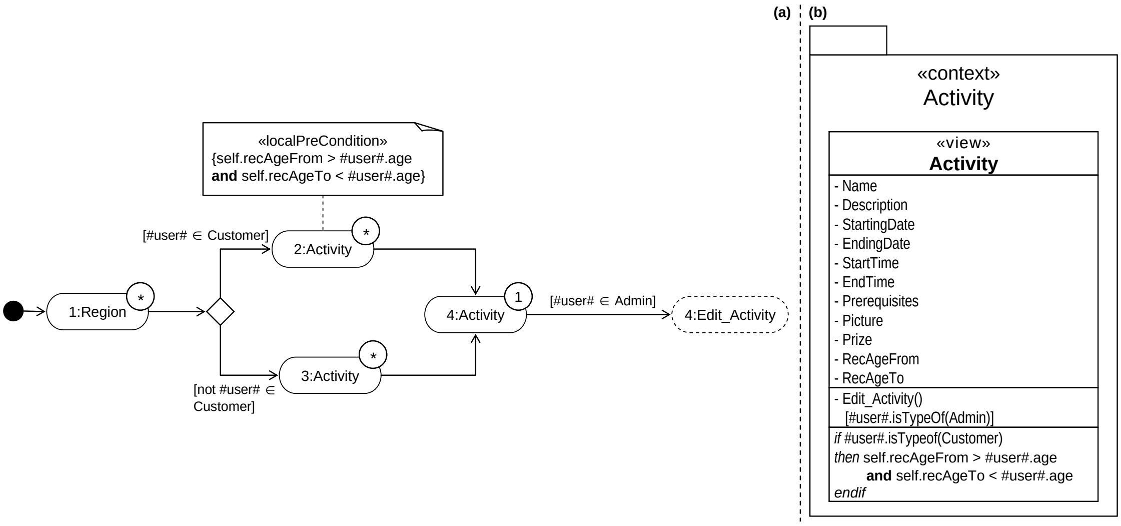


Figure 2.23: OOWS: (a) Activity Diagram Show Activities, (b) Navigation Context Show Activities

In Figure 2.23*(b)*, the resulting navigational context can be seen. In particular, the third compartment of the *Activity* view is important for the required adaptation. The filter for presenting the user with activities according to his/her age is only applied if the user is logged in (cf. `if #user# isTypeOf(Customer)`), otherwise no filtering is done.

Customization Scenario Administrator Links. For editing activities, administrators need to log in to the web application and navigate to the list of activities for a certain region as is specified in Figure 2.23*(a)*. Having selected on activity, administrators are presented with a special link allowing them to edit the activity, as opposed to other normal users. This is ensured by the following condition associated with the control flow of the activity diagram:

```
#user# ∈ Admin
```

Furthermore, in the third compartment of the *Activity* view in Figure 2.23*(b)*, the *Edit_Activity()* operation is shown which corresponds to the previous condition and is only executable if the user is logged in as administrator.

Customization Scenario Special Offers. For the OOWS approach, this scenario is explained with special offers in the context of booking a hotel room. The activity diagram for this scenario is shown in Figure 2.24*(a)*.

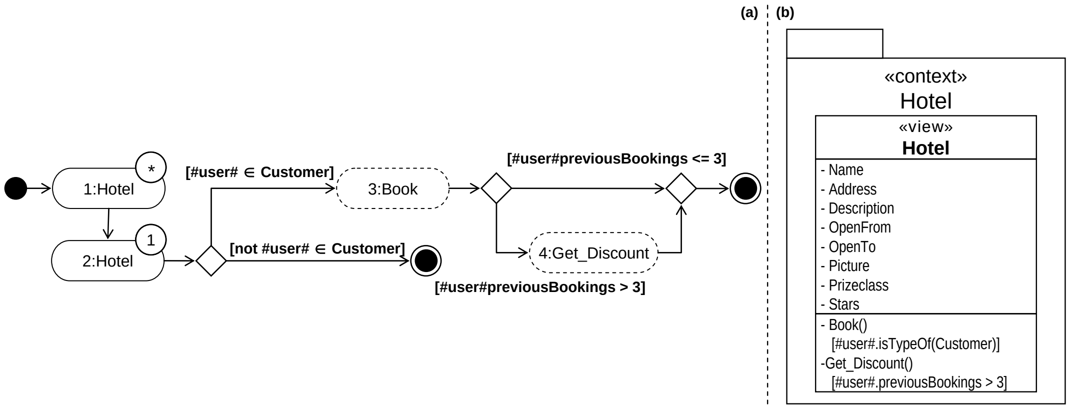


Figure 2.24: OOWS: (a) Activity Diagram Book Hotel, (b) Navigation Context Book Hotel

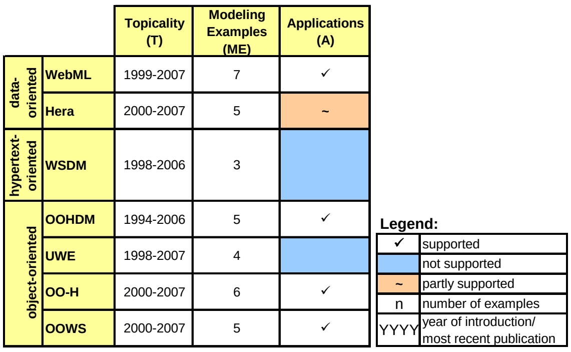
It is depicted that, in the first step, a hotel has to be selected from an index of hotels (cf. action *1:Hotel*). The index is denoted with the '*' label associated to the action, while the '1' label indicates one instance to be displayed. Subsequently, the user can book a hotel, whereby the adaptation comes into play (cf. action *3:Book*). If the user has made more then three bookings in the past s/he gets a discount provided by the action *4:Get_Discount*. However, if the condition is not fulfilled, no discount is allowed. The required adaptation rule to realize this functionality is incorporated into the activity diagram by annotating the transitions with conditions. For example, with the #user# variable it is possible to retrieve the actual user object and its attribute values - in this case #user#.previousBookings retrieves the value of the *previousBookings* attribute of the class *User* or rather of *Customer*. In Figure 2.24*(b)*, the corresponding *navigation context* is illustrated for action *2:Hotel* of the activity diagram shown in Figure 2.24*(a)*. Each navigation context must have exactly one so-called *manager class* which defines a view on the content level, i.e. the *Hotel* class in this case. Concerning the adaptation rule in the activity diagram, the resulting specification is an operation *Get_Discount()* which is only executable under the condition which was specified in the activity diagram, namely the user must have made more than three bookings in the past.

2.3 Lessons Learned

In this section, the experiences acquired during the evaluation of the selected approaches as well as during modeling of the running example shall be summarized. The results of the evaluation have revealed interesting peculiarities of current web modeling approaches. The following subsections sum up the findings of the evaluation from Section 2.2 and in particular illustrate the results at a glance with tables according to the five categories of criteria from the criteria catalogue as well as point out what needs to be done in terms of further development of web modeling approaches.

2.3.1 Maturity

Small Set of Similar Modeling Examples. A first overview on the web modeling field showed that the individual approaches often made use of similar modeling examples. Since stemming from academia, it is not surprising that a conference management system as well as some kind of department web site have been used particularly often to demonstrate a web modeling approach in terms of modeling examples. Beyond this, different kinds of e-stores, e.g., selling books and CD's, as well as art gallery web applications have been used several times. The current set of modeling examples used in the web modeling field, however, raises some important questions: Are those examples complex enough to show the approaches' applicability? Do they cover all different kinds of web applications, e.g., ubiquitous web applications, workflow-based web applications? Consequently, what the web modeling field needs is a set of generally acknowledged modeling examples in a public catalogue. Moreover, reference implementations for those examples can serve as "testbeds", i.e., to show if it is possible to produce such web applications with existing web modeling approaches.



		Topicality (T)	Modeling Examples (ME)	Applications (A)
data-oriented	WebML	1999-2007	7	✓
	Hera	2000-2007	5	~
hypertext-oriented	WSDM	1998-2006	3	🟦
object-oriented	OOHDM	1994-2006	5	✓
	UWE	1998-2007	4	🟦
	OO-H	2000-2007	6	✓
	OOWS	2000-2007	5	✓

Legend:

✓	supported
🟦	not supported
~	partly supported
n	number of examples
YYYY	year of introduction/ most recent publication

Table 2.1: Maturity

Rare Application in Real-World Projects. Web modeling approaches have already a 10 year old history. Nevertheless, their application in real-world projects in particular in the context of a commercial setting is still rare. According to the survey of Barry et al. [BL03], in practice the awareness of academic methods is still rather low and consequently, they are rarely used by practitioners. In this respect, there is an urgent need for more reference applications which are built with academic methods in order to prove their maturity. First, possible application areas are academic web sites such as a department web site. In a second step, cooperation with industry is necessary. Furthermore, besides existing scientific publications, web modeling approaches will need to be presented in a way that is more suitable for practitioners.

Approaches are Continuously Evolving. All of the surveyed web modeling approaches have been presented in numerous publications including refereed papers, articles, books as well as manuals. Over the time, each of the approaches has been subject to extensions, e.g., for supporting business process modeling and customization modeling, as well as to major evolutions, e.g., the introduction of ontologies as a new formalism to specify models and the use of standards enabling model-driven web engineering. From a practitioner's point of view, these developments might communicate that current web modeling methodologies are not yet mature enough to be used in practice. For example, some approaches provide alternative proposals for modeling customization for which it is not clear how they are related to each other and if they can be used in parallel. The more, a distinction between documentation of the web modeling approaches specifically dedicated to practitioners and ongoing scientific work is needed. In this respect, a comprehensive description of the already matured parts of a web modeling approach in terms of a manual for designers need to be provided.

2.3.2 Web Modeling

UML for Content Modeling - Proprietary Solutions for Hypertext Modeling. UML is quite popular for modeling the content level as well as the hypertext level of a web application. While UWE and OO-H are based on UML 1.x, OOHDM uses a UML-like notation. At hypertext level, however, UWE and OOHDM are the only approaches that continue using (stereotyped) UML class diagrams. Indeed, at hypertext level, one can find very different languages and notations for each individual approach.

Presentation Level Seldomly Addressed. When it comes to the supported levels of web applications it becomes obvious that the presentation level is often only a marginal concern [WS00]. Although nearly every method does provide presentation level support, it is typically omitted in modeling examples. Interestingly, WebML is the only method not providing a presentation model. Still, presentation issues are dealt within WebML's tool support, where information from the hypertext level can be placed on a grid and style information can be associated. With respect to the other approaches, nodes of the hypertext level often are mapped one-to-one to pages from the presentation level

		Levels											Features (F)			Phases (Ph)				Development Process (Pr)			
		Supported Levels (L)			Interfaces (I)																		
					C⇔H				H⇔P														
		C	H	P	graphical	text-based	natural language	separated	graphical	text-based	natural language	separated	C	H	P	R	A	D	I	Origin	Steps	Artifacts	Actors
data-oriented	**WebML**	ER	own	🟦	✓	✓	🟦	🟦	n/a				s	s/b	s	✓	✓	✓	✓	Boehm's Spiral model	7	✓	✓
	Hera	RDFS	own	own	✓	✓	🟦	🟦	✓	✓	🟦	🟦	s	s/b	s	🟦	🟦	✓	✓	own	~6	✓	🟦
hypertext-oriented	**WSDM**	ORM	own	own	✓	🟦	🟦	🟦	✓	🟦	🟦	🟦	s/b	s	s	✓	✓	✓	✓	own	5	✓	🟦
object-oriented	**OOHDM**	UML-like	own, UML-like	own	🟦	✓	🟦	🟦	✓	🟦	🟦	🟦	s	s	s/b	✓	✓	✓	✓	own	5	✓	🟦
	UWE	UML CD	UML CD & OCL, UML SM	UML CD UML SD	🟦	✓	🟦	🟦	✓	🟦	🟦	🟦	s	s/b	s/b	✓	✓	✓	✓	RUP	5	✓	✓
	OO-H	UML CD	UML CD & OCL	own	✓	✓	🟦	🟦	🟦	🟦	✓	n/a	s/b	s	s	✓	✓	✓	✓	own	1	✓	🟦
	OOWS	CD, SD, SM	own	own	✓	🟦	🟦	🟦	n/a				s/b	s/b	s	✓	✓	✓	✓	own	4	✓	✓

Legend:

✓	supported
🟦	not supported
n/a	not applicable
s/b	structure / behavior
n	number of phases
RUP	Rational Unified Process
ER	Entity Relationship Diagram
UML	Unified Modeling Language
CD	Class Diagram
SD	Sequence Diagram
SM	State Machine Diagram
OCL	Object Constraint Language
ORM	Object Role Modeling
RDFS	Resource Description Framework Schema

Table 2.2: Web Modeling

Behavioral Modeling Not Comprehensively Considered. Behavioral modeling is typically not supported comprehensively for all levels. There is, however, a tendency for using behavioral diagrams, including use cases, activity diagrams, concurrent task trees, in the requirements engineering and analysis phases, as well as for describing scenarios of user behavior with e.g. sequence diagrams. Some form of behavioral modeling is also introduced by approaches providing support for business process modeling or workflow-based web applications.

Strong Processes. Almost all of the surveyed approaches support the developer with appropriate guidance to developing a web application from requirements engineering to implementation on the basis of their modeling techniques, i.e., the necessary steps, artifacts to be produced within each step, and actors are explained. While most of the approaches propose their own development process, WebML and UWE do base their process on existing work, i.e. Boehm's Spiral model and RUP, respectively. Interestingly enough, all approaches start modeling the web application's content level. Still, the applicability and usability of these processes need to be investigated in real-world projects.

2.3.3 Customization Modeling

Set of Context Properties Limited and Not-Extensible. All web modeling approaches do support customization with respect to the user context property thus laying the path to personalization, while other context properties are often not taken into account. The investigation revealed that context properties typically are considered in isolation and that complex adaptations regarding several context properties are rare. Furthermore, the extension of the supported set of context properties is typically not discussed in current web modeling approaches. In this respect, the context model of OO-H represents the only exception considering user, location, device, time, and network context as well as allowing for their extension. Furthermore, except for the WebML and OO-H approaches, there is no support for explicit modeling concepts capturing context information. Typically, it is assumed that context information is updated by some external service.

Context Chronology and Complex Context not Addressed. Currently, none of the investigated web modeling approaches considers complex contexts as well as basing adaptations on historical context information. Consequently, adaptations have to be specified as a reaction to the sum of simple contexts. Concerning context chronology, some approaches allow for adaptations according to the user's navigation behavior. This historical information about the user is often implicitly available in terms of predicates of a rule language such as in WebML and OO-H. Still, other historical context information, such as the user's location over time, cannot be stored.

Set of Adaptation Operations Limited and Not-Extensible. The set of adaptation operations that some web modeling approaches provide is limited to operations that can be performed upon typical concepts of a web application, including add/remove a link, change the style, add/remove a node, sort some information. Currently, there seems to be no web modeling approach that supports operations on media types such as ”resize image” or ”shorten text”.

Complex Adaptation Operations not Considered. Complex adaptations currently have been realized in the WSDM approach only. The *promoteNode* and *demoteNode* operations have been built upon primitive ones such as add/remove link. Nevertheless, none of the approach provides the necessary modeling means that allow specifying complex adaptation operations on the basis of primitive ones.

Limited Support for Content, Presentation, and Interface Adaptations. Adaptations at the hypertext level are considered within all of the investigated approaches. With respect to content and presentation level only half of the approaches provide necessary adaptation operations, while interface adaptation is supported by WSDM and OOHDM, only. More interestingly, approaches supporting presentation adaptations rather operate at a coarse-grained level, e.g., by providing adaptation operations that change the complete style of the web applications presentation. In contrast, the Hera approach allows defining alternative layout managers for parts of the presentation model, thus realizing adaptation at a more fine-grained level.

		Context									Adaptation													
		Properties (P)					Extensibility (CE)	Chronology (C)	Complex Context (CC)	Separation of Context (SC)	Operations (O)	Extensibility (AE)	Complex Adaptation (CA)	Levels (L)			Interfaces (I)		Granularity (G)	Separation of Adaptation (SA)	Phases (CP)			
		User	Location	Device	Time	Network								C	H	P	C⇔H	H⇔P			R	A	D	I
data-oriented	WebML	✓	✓	✓	🟦	✓	~	~	🟦	~	✓	🟦	🟦	🟦	✓	✓	✓	🟦	mi/ma	🟦	🟦	🟦	✓	🟦
	Hera	✓	🟦	✓	🟦	🟦	🟦	🟦	🟦	~	🟦	🟦	🟦	✓	✓	✓	🟦	🟦	mi	✓	🟦	🟦	✓	✓
hypertext-oriented	WSDM	✓	✓	🟦	✓	🟦	🟦	~	🟦	🟦	✓	🟦	✓	🟦	✓	🟦	✓	🟦	mi/ma	🟦	✓	✓	✓	✓
object-oriented	OOHDM	✓	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	✓	✓	🟦	✓	🟦	mi/ma	🟦	🟦	🟦	✓	🟦
	UWE	✓	🟦	🟦	🟦	🟦	🟦	🟦	🟦	~	✓	🟦	🟦	🟦	✓	🟦	🟦	🟦	mi/ma	✓	🟦	✓	✓	🟦
	OO-H	✓	✓	✓	✓	✓	✓	~	🟦	~	✓	~	🟦	✓	✓	✓	🟦	🟦	mi/ma	✓	🟦	🟦	✓	✓
	OOWS	✓	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	✓	🟦	🟦	🟦	✓	🟦	🟦	🟦	mi	🟦	✓	✓	✓	🟦

Legend:

✓	supported
🟦	not supported
~	partly supported
mi	micro
ma	macro

Table 2.3: Customization Modeling

Customization not Comprehensively Considered in all Development Phases. Customization modeling is predominantly considered during design, the exceptions being the OOHDM and the WSDM approaches. It seems, that customization is treated as a separate step in the design phase in which an existing web application is extended with customization issues. Instead, customization needs to be considered during all phases in the software development lifecycle. Consequently, web modeling approaches will need to adapt their current development processes in order to appropriately include customization concerns in all phases.

Disregarded Crosscutting Nature of Customization. Current web modeling languages do not allow for modeling customization separate from the rest of a web application model. More specifically, this is due to the missing separation of context and adaptation.

Considering separation of context, in general, all web modeling approaches do acknowledge the need for defining context information in a separated model. Still, guidelines supporting a developer in constructing a context model are often not available. Some guidance can be found in the WebML approach as well as in the OO-H framework for modeling context information. Nevertheless, none of the approaches is able to achieve full separation of context from the rest of the web application models, i.e., typically, concepts from the context model are connected to concepts from the content model in some way or the other, e.g., via associations. As another example, it is often not decidable if a certain concept shall be modeled within the content or the context model, such as the user concept of which some attributes might contribute to the core functionality of the web application and some others contribute to customization.

With respect to separation of adaptation, developers typically are required to define adaptations as annotations to existing models. In the UWE, OO-H, and Hera approaches, adaptations at the hypertext level can be captured separately from the rest of the web application models, however. Thus, customization functionality is not intermingled with the rest of the web application. Nevertheless, there is a need for comprehensively capturing customization from all levels of a web application. Furthermore, it is important to know where adaptations do take effect in models, meaning a modeling language needs to specify these subjects of adaptations. While this is supported by UWE's and Hera's aspect-oriented approaches to modeling customization for the hypertext level, in OO-H the same information is captured within the Event and Condition parts of its rule language. Concluding, aspect-orientation represents an suitable mechanism for separately capturing customization but up to now has not been used comprehensively for all levels in a web application model. For example, at the content level aspect-orientation can be used to capture the parts (e.g. attributes) of the user concept the represent context information within a separate aspect, while the application-specific parts remain in the content model.

2.3.4 Model-driven Engineering Criteria

From Notations to Languages. Most web modeling approaches have emerged rather focusing on notations than on using standards for specifying their language. Today, with the rise of model-driven engineering, the semantic web and the general need to produce a running system from the web application models, more and more web modeling approaches do provide formal specifications of their languages in terms of either metamodels, UML-profiles, or ontologies.

Model Transformations not Based on MDE Standards. Model transformations are supported by almost all approaches in one way or another. Still, only the UWE web modeling language has recently been extended to better support model transformations in the sense of MDE, i.e., through providing QVT transformations.

Lack of Platform Description Models. None of the approaches does provide platform description models and consequently no model transformations from platform-independent to platform-specific models are supported.

		Language Definition (M.L)			Model Transformation Type (M.T)				Platform Description Model (M.P)
		Metamodel	Grammar	Semantic Description	PIM2PIM	PIM2PSM	PIM2Code	PSM2Code	
data-oriented	WebML	🟦	DTD	🟦	🟦	🟦	Struts	🟦	🟦
	Hera	🟦	🟦	RDFS CC/PP	🟦	🟦	HTML, WML	🟦	🟦
hypertext-oriented	WSDM	🟦	🟦	OWL	✓	🟦	XSLT	🟦	🟦
object-oriented	OOHDM	~	🟦	RDFS, OWL	🟦	🟦	🟦	🟦	🟦
	UWE	MOF	🟦	🟦	✓	🟦	J2EE/ Cocoon	🟦	🟦
	OO-H	MOF	DTD, BNF	🟦	✓	🟦	PHP	🟦	🟦
	OOWS	MOF	🟦	OWL	✓	🟦	J2EE/ .Net	🟦	🟦

Legend:

✓	supported
🟦	not supported

Table 2.4: Model-driven Engineering

2.3.5 Tool Support

Lack of (Extensible) Tool Support. One of the most problematic issues in web modeling is the lack of tool support for the individual approaches. From the set of surveyed approaches only four provide tool support that has been made available to the public community. Without proper tool support, however, web modeling methods will not gain acceptance in practice but possibly will remain a playground for academic ideas. Out of the four tools, WebRatio and VisualWade represent the only commercial tools and thus have left the status of a prototype implementation. Furthermore, WebML offers an academic license program which has already been installed at several universities. None of the available tools is offered under an open-source license, which would attract developers of open-source web frameworks and technologies. Although ArgoUWE, the tool accompanying the UWE method, has been built upon the open-source tool ArgoUML, the extensions made are not open-source. WebRatio is the only tool offering some built-in extension mechanism for the WebML language, i.e., a plug-in mechanism for so-called custom units.

Customization Modeling not Supported by Tools. Currently, all tools provide support for basic web modeling (with some deviations from the original notation) and thus allow for modeling support. Still, modeling support for dealing with customization modeling is only provided by Hera's tool which allows modeling context information within user profiles used to statically generate adapted hypermedia presentations. For the customization modeling extensions of WebML and OO-H, prototype implementations have been reported on in literature but are not made available yet.

					Modeling Support (T.M)					Process Support (T.P)		
	Version (T.V)	Standalone Application (T.B)	Open Tool (T.O)	Costs (T.C)	Web Modeling Support	Customization Modeling	Model Pre-Generation (T.MG)	Consistency Check (T.CC)	Code Generation (T.CG)	Process of Method	Process Type	Collaboration (T.Co)
WebML: ***WebRatio***	4.3 (acad.)	✓	✓	com	✓	🟦	✓	od	✓	~	phases	✓
Hera: ***Hera Presentation Generator***	1.3	✓	🟦	free	✓	~	🟦	wm/ od	✓	✓	wizard	🟦
UWE: ***ArgoUWE***	0.16	🟦	🟦	free	✓	🟦	✓	wm	🟦	~	step-wise	🟦
OO-H: ***VisualWade***	1.2 rev 163	✓	🟦	com	✓	🟦	✓	od	✓	~	step-wise	🟦

Legend:

✓	supported
🟦	not supported
~	partly supported
n/a	not applicable

free	freeware
com	commercial
od	on demand
wm	while modeling

Table 2.5: Tool Support

Support for Model Transformations Provided. Although not based on MDE standards, all tools offer some form of model transformation, e.g., for generating a hypertext model from the content model, or a presentation model from a hypertext model (ArgoUWE, WebRatio, and VisualWade), or for integrating different models as a prerequisite for code generation (HPG).

Code Generation for One Platform Only. Except for the ArgoUWE tool, all approaches provide code generation support. Still, code generation is limited to specific platforms, only. While VisualWade generates PHP code, WebRatio targets J2EE platforms by producing code for the Struts framework. WebRatio is shipped with a Tomcat Servlet Container and provides for simple deployment of the web application. The HPG of Hera, however, does support generating static hypermedia presentations in several formats, including HTML as well as WML. As a consequence, the employment of web modeling tool also determines the runtime platform.

Limited Process Support. Remarkably, only the HPG tool of Hera in combination with its Model Builders implements the process as described by the supported method. All other approaches have only partial tool support for their defined processes. Typically, the tools do not offer means for supporting developers in producing artifacts (e.g., use cases) from earlier development phases but start with designing the content model. WebRatio allows going back and forth between different models as well as making several changes to either of them without loss of previously modeled artifacts. Instead, ArgoUWE as well as OO-H require the user to follow the web applications levels dimension in first designing the content, hypertext, and finally the presentation level. This is of course a limitation with respect to the iterative development as required for web application development projects.

Lack of Collaboration in Tools. Collaborative work on web application development currently not supported, should be addressed by tools, since web applications are not built single-handed but by a project team. WebRatio, however, represents an exception already supporting CVS.

2.4 Related Surveys

In an effort to shed light on the different approaches to web application development, some surveys have already been published. Following, these surveys are distinguished according to their specific goals and foci into closely related work representing *customization modeling surveys* and more widely related work representing *web modeling surveys* focusing on general web modeling criteria, development processes, requirements engineering, and support for modeling rich internet applications.

2.4.1 Customization Modeling Surveys

Barna et al. [BFHV03] provide a comparison of four approaches, amongst them Hera, OOHDM, and UWE also investigated in the present survey. The approaches are investigated according to their specific design models for content, hypertext, and presentation levels as well as their support for customization design, though the focus is rather on personalization. The discussion is supported using a simple running example of a virtual art gallery.

In a further evaluation but already some time ago, Kappel et al. [KPR+01] compare the customization modeling capabilities of OOHDM and WebML with respect to supported context, granularity of adaptations, and the degree of customizability.

Similar to the above mentioned evaluations, this survey's focus is on investigating the support of modeling customization in current web modeling languages but in contrast, specifically considers also the model-driven development of UWAs including tool support. Besides this difference in goals, this survey is also different in terms of comprehensiveness by surveying seven recent web modeling approaches which provide means for customization modeling and if available their tool support. The evaluation is based on a well-defined as well as fine-grained catalogue of more than 30 criteria, which allows a detailed investigation of each approach with respect to general web modeling characteristics, customization modeling characteristics, as well as model-driven engineering and tool support. In contrast, related works provide less than 5 criteria or no explicit description in terms of a catalogue at all. Furthermore, the evaluation is supported with a running example consisting of five different customization scenarios which is used to better explain the general modeling concepts of each approach and in particular the provided means for customization modeling.

2.4.2 Web Modeling Surveys

General Web Modeling. In Schwinger et al. [SK06], an introduction into modeling web applications is given, including a brief overview of eleven web modeling approaches based on a set of twelve criteria, amongst them one criterion evaluating support for customization modeling as well as code generation.

In contrast, this survey focuses on web modeling approaches providing support for modeling customization. It also differs in that this survey applies a more detailed criteria catalogue as well as a running example while it includes the relevant approaches also surveyed in [SK06].

Development Processes. Nora Koch [Koc99] has evaluated eleven approaches, amongst those only OOHDM and WSDM that are still evolving. That survey specifically focuses on the approaches' development process, supported development phases, modeling techniques and notations used, as well as tool support. In the end, the characteristics of the Rational Unified Process (RUP) [Kru00] are presented as well as a discussion on how some approaches support parts of RUP. Customization, however, is not a focus as in the survey of this thesis.

In Woukeu et al. [WCWH03] eight approaches are investigated, having in common with this survey the WebML, OOHDM, and WSDM web modeling approaches. The evaluation is focused on the supported development process, i.e., their phases, as well as the modeling techniques used. Furthermore, each approach's concepts from the hypertext level are listed, e.g., 'navigation classes' in OOHDM as well as 'pages' and 'content units' in WebML. Finally, each approach is evaluated if it allows to model read-only or read-write web applications so that the survey differs considerably in terms of focus.

Additionally, aiming at a fine grained set of criteria, the criteria used in the works of Nora Koch and Woukeu et al. have been adopted in the catalogue of criteria proposed in this survey (cf. Section 2.1.2). Furthermore, where appropriate they have been endowed with a clear definition including a measurement scale or they have been refined. Such a refinement generally means the decomposition of a criterion into several criteria.

Requirements Engineering. In Escalona et al. [EK04], the scope is requirements engineering for web applications. A comparison of ten web modeling approaches is provided including WSDM, OOHMD, UWE, and WebML, which are also investigated within this survey. In particular, the types of requirements, the activities and techniques employed during requirements elicitation, specification and validation, and the methodologies's focus on the requirements process, techniques, or artifacts have been evaluated.

In contrast to the work of Escalona et al, this survey is rather concerned with the design level means of today's web modeling languages. Nevertheless, the surveyed approaches are investigated with respect to their support of a requirements engineering phase and more particularly, if customization modeling is already considered during requirements engineering. Consequently, this evaluation is complementary to the one of Escalona et al.

Support for Rich Internet Applications. Preciado et al. [PTSC05] compare fifteen representatives from web modeling, multimedia and hypermedia methodologies according to their applicability to model rich internet applications. Again, five of the approaches are also evaluated within this survey, namely, UWE, OO-H, WebML, WSDM, and OOHMD. The set of ten evaluation criteria include multimedia modeling, personalization modeling, and tool support, and are evaluated to a weighted measurement scale consisting of four degrees of coverage.

Again this survey's focus is different to the one of Preciado et al. which investigate their selection of approaches concerning their applicability to model RIAs. Customization modeling, however, is only a marginal concern supported in the work of Preciado et al. with one criterion, only, which in this survey is evaluated in much more detail.

2.5 Summary

This chapter has presented the state-of-the-art in model-driven development of ubiquitous web applications. More specifically, an in-depth comparison of seven web modeling approaches currently supporting the development of ubiquitous web applications has been provided. An evaluation framework has been designed on the basis of a detailed and well-defined catalogue of evaluation criteria. Moreover, the actual evaluation by means of this criteria catalogue is supported by a modeling example, i.e., a tourism information web application, used to provide an initial insight into each approaches' concepts for modeling customization as well as to facilitate their comparability. More specifically, a set of five customization scenarios has been defined, to be tested with each web modeling approach. The per-approach evaluation is furthermore complemented with an extensive report on lessons learned, summarizing the approaches' strengths and shortcomings. In this respect, limitations of current web modeling languages with respect to the model-driven development of ubiquitous web applications are the lack of a proper MDE foundation in terms of metamodels as well as missing tools allowing to model customization. Furthermore, the proposed customization mechanisms are often limited, since they neither cover all relevant context factors in an explicit, self-contained, and extensible way, e.g., within a dedicated context model, nor allow for a wide spectrum of extensible adaptation operations. Furthermore, the provided customization mechanisms frequently do not allow dealing with all different parts of a web application in terms of its content, hypertext, and presentation levels as well as their structural and behavioral features. Finally, current web modeling approaches insufficiently consider the crosscutting nature of customization by not providing the necessary means to comprehensively capture customization separately from all levels of a web application model. Nevertheless, the evaluation has revealed that, the WebML approach provides one of the more powerful customization mechanisms, besides the OO-H approach, but fails to tackle the crosscutting nature of customization. Consequently, on the basis of this evaluation, the WebML approach has been chosen to be bridged to the aspect-orientation paradigm in the context of this thesis.

3 State-of-the-art in Aspect-oriented Modeling

Contents

The concept of Separation of Concerns (SoC) can be traced back to Dijkstra [Dij76] and Parnas [Par72]. Its key idea is the identification of different concerns in software development and their separation by encapsulating them in appropriate modules or parts of the software. Aspect-Oriented Software Development (AOSD), formerly also called Advanced Separation of Concerns (ASoC), adopts this idea and further aims at providing new ways of modularization in order to separate crosscutting concerns from traditional units of decomposition during software development. In particular, AOSD represents the convergence of different ASoC approaches, such as Adaptive Programming (AP) [Lie96], Composition Filters (CF) [ABV92], Subject-Oriented Programming (SOP) [HO93], Multi-Dimensional Separation of Concerns (MDSoC) [TOHS99], and Aspect-Oriented Programming (AOP) [KLM^{+}97]. Nevertheless, it is a fairly young but rapidly advancing research field. From a software development point of view, aspect-orientation has originally emerged at the programming level with AspectJ[1] being the most prominent protagonist. Meanwhile, the application of the aspect-oriented paradigm is no longer restricted to the programming level but more and more stretches over phases prior to the implementation phase of the software development life cycle such as requirements engineering, analysis, and design. This development is also driven by the simultaneous rise of Model-Driven Engineering (MDE) which employs models as the primary artifact in software development [Sch06b]. As a result, there has already been a considerable number of aspect-oriented modeling (AOM) languages proposed in literature, whereof only a few in the meanwhile have come of age. Each of those AOM approaches has different origins, e.g., AOP and SOP, and pursues different goals. This entails not only the problem of different terminologies but also leads to a broad variety of aspect-oriented concepts, including different composition mechanisms used [KL06], as well as diverse notations.

On the basis of previous work [SSK^{+}07], this chapter provides an introduction into AOM as well as a detailed discussion of the state-of-the art in the domain. Section 3.1 presents the *Conceptual Reference Model* (CRM) for AOM in order to centrally capture the basic concepts of AOM and there interrelationships in terms of a UML class diagram. On this basis, an evaluation framework

[1] http://www.eclipse.org/aspectj/

is set up by deriving a detailed and well-defined catalogue of evaluation criteria in Section 3.2. Section 3.3 is dedicated to the actual evaluation of eight selected AOM approaches by means of this criteria catalogue. To better illustrate the notational peculiarities of the AOM approaches, this evaluation is accompanied with a running example. The lessons learned are presented, in Section 3.4. In Section 3.5, the contributions of this survey with respect to other existing surveys are discussed, before the chapter is closed with a brief summary in Section 3.6.

3.1 The Conceptual Reference Model for Aspect-Oriented Modeling

The major difficulty in comparing AOM approaches is the lack of a common understanding for the basic ingredients of aspect-oriented modeling. This is on the one hand due to different concepts introduced by related AOSD approaches (e.g., AP, CF, SOP, and MDSoC) and on the other hand due to the very specific meaning of AOP level concepts, particularly those coined by AspectJ [SKK04]. An example for the first issue is the concept of "aspect", where similar though different concepts have been introduced by related AOSD approaches, e.g., "hyperslice" in Hyper/J, "filter" in CF, and "adaptive method" in Demeter/DJ [vFGCdL03]. An example for the second issue are AspectJ's join points which are defined as "points in the execution of the program" including field accesses, method, and constructor calls [Tea05]. This definition is not comprehensive enough for the modeling level, however. First, modeling languages unify specific programming language concepts into more abstract modeling elements to be able to serve several different programming languages. And second, modeling languages typically are richer in terms of concepts, i.e., modeling elements, that could serve as join points. This is also due to different views available at modeling level, e.g., structural views and behavioral views.

In the light of different terminologies and a broad variety of aspect-oriented concepts, for an evaluation of AOM approaches it is essential to first establish such a common understanding by means of a so-called *Conceptual Reference Model* for AOM. The CRM enables to explain the basic ingredients of aspect-oriented modeling and their interrelationships both in terms of a graphical representation as a UML class diagram and in terms of a glossary comprising a textual definition for each concept introduced in the class diagram. The conceptual reference model, for which a previous version has already been proposed in [SSK^{+}06], represents an intermediate step and forms the basis for setting up an evaluation framework, i.e., inferring concrete criteria as it is done in Section 3.2.

In AOSD literature, one can already find some proposals for reconciliating the currently prevailing diversity in the understanding of concepts from the aspect-orientation paradigm each pursuing a specific focus [KL06], [vdBCC05], [vFGCdL03]. In this thesis, they have been used as a basis for establishing the CRM for AOM. Their particular influence on the CRM is discussed as follows:

- Considering the broader research area of ASoC, one can distinguish between four composition mechanisms, namely, *pointcut-advice*, *open class*, *compositor*, and *traversal* [MK03]. In the CRX model of Kojarski et al. [KL06] pointcut-advice, open class, and compositor mechanisms are supported, only, because the traversal mechanism does not necessarily share properties with the other mechanisms that can be reasonably generalized [KL06]. For the CRM, in this thesis, the authors' idea of first abstracting over the three aspect-oriented composition mechanisms, which later allows to compare AOM languages at a higher level, is adopted. Beyond, the CRM shall capture in detail the corresponding AOM language concepts for each composition mechanism separately. In Section 3.2, this allows to set up a fine-grained set of criteria for each composition mechanism and consequently allows AOM languages realizing the same composition mechanism(s) to be compared in greater detail.

- In van den Berg et al. [vdBCC05] an attempt towards establishing a common set of concepts for AOSD has been made. The proposed definitions are intended to be appropriate for all phases in the software development life cycle. The AOSD Ontology of van den Berg et al. discusses high level concepts such as ”concern” and ”composition”, which allow abstracting over different composition mechanisms such as proposed by Kojarski et al. When looking at the concepts describing the specifics of the different composition mechanisms, however, one can see that the focus is rather on the pointcut-advice and open class mechanisms. Concepts for supporting the compositor mechanism such as ”merge” and ”match method” (cf. Section 3.1.3) are not discussed in the proposed glossary. Beyond, a visualization of the glossary and the concepts’ interrelationships in terms of a conceptual model is missing. The CRM is based on the AOSD Ontology in that the proposed definitions of concepts are adopted, i.e., if such definitions are available.

- The ”theory of aspects” of Chavez et al. [vFGCdL03] describes a ”conceptual framework for AOP” in terms of Entity-Relationship diagrams and a textual description of each entity. The framework, however, explicitly supports aspect-oriented approaches that follow the pointcut-advice and open class mechanisms, only. The framework, e.g., explicitly demands the aspect-base dichotomy, meaning the clear distinction between ”aspects” and ”base”. Consequently the ”theory of aspects” does not describe concepts supporting the compositor mechanism. Nevertheless, the Entity-Relationship diagrams have served as an input for designing the CRM. The definitions of concepts proposed in the AOSD Ontology [vdBCC05] have been preferred over those of Chavez et al. [vFGCdL03], however, since the AOSD Ontology’s terminology is closer to the original terminology of the pointcut-advice mechanism (e.g., ”enhancement” in [vFGCdL03] corresponds to ”advice” in [vdBCC05]).

For those concepts where no definition is available in the discussed literature, a bottom-up approach is followed, taking into consideration the surveyed approaches.

In the following, the concepts of the CRM are described along with its four major building blocks as depicted in Figure 3.1. The *ConcernComposition* package provides a high level view on the concepts of AOM abstracting over different composition mechanism, while the *Language* package describes the means underlying the specification of concerns. The specific composition mechanisms are specialized in separate packages, i.e., the pointcut-advice and open class mechanisms are specialized into the *AsymmetricConcernComposition* package, and the compositor mechanism is specialized in the *SymmetricConcernComposition* package. The concepts’ descriptions possibly contain a reference to the source definition and an optional discussion in case the definition of a concept has been refined.

3.1.1 ConcernComposition

The *ConcernComposition* package abstracts over the different composition mechanisms. It deals first, with the modularization and thus with the separation of a system’s concerns into appropriate units and second, with their interrelationships, and consequently their composition by means of appropriate rules.

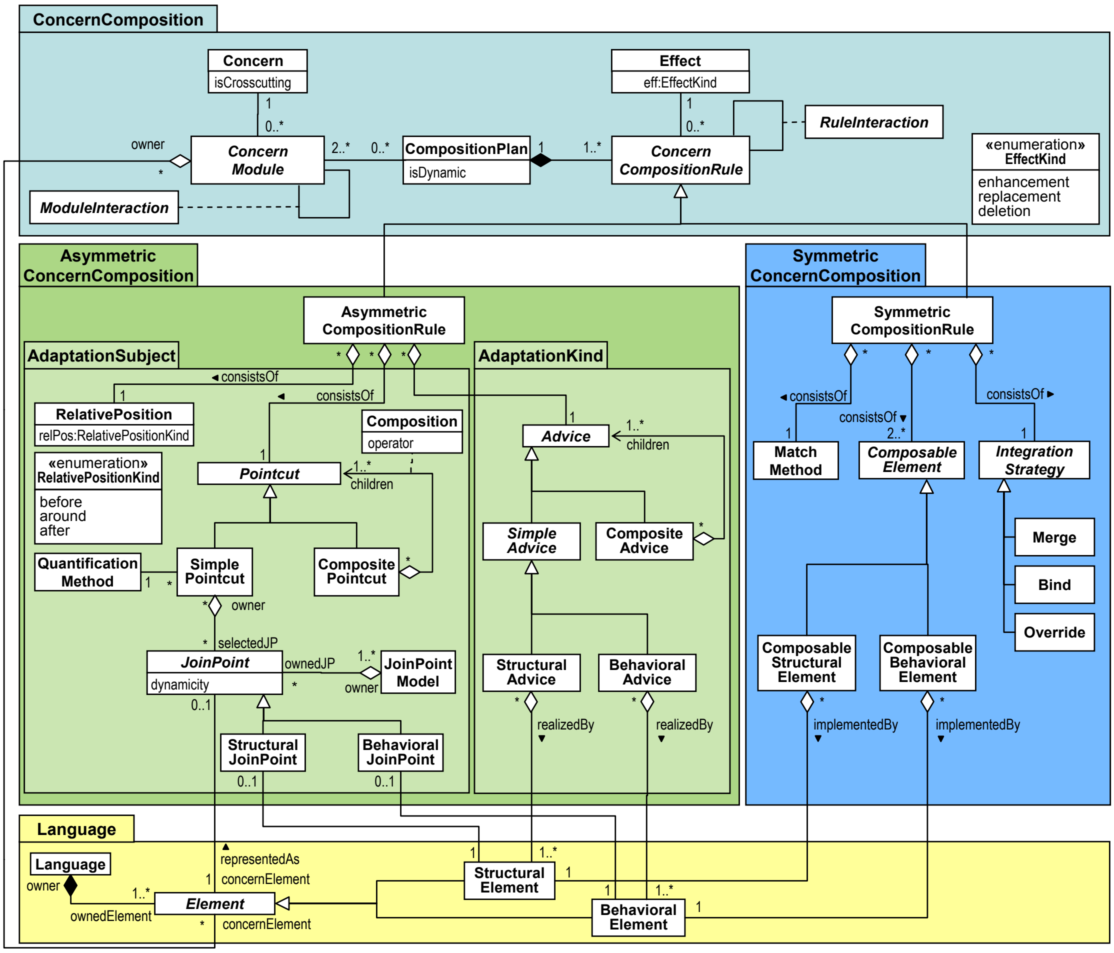


Figure 3.1: The Conceptual Reference Model for Aspect-Oriented Modeling

Concern. Along with [vdBCC05] a *concern* is defined as an interest which pertains to the system's development, its operation or any other matters that are critical or otherwise important to one or more stakeholders. A concern is called a *crosscutting concern* if it cannot be modularly represented within a language's decomposition technique, e.g., classes and methods in the object-oriented paradigm (cf. Figure 3.1 attribute *isCrosscutting*). In AOSD literature, this restriction is called the tyranny of the dominant decomposition [TOHS99]. The elements of a crosscutting concern are then said to be scattered over other concerns and tangled within other concerns of a specific system [vdBCC05]. In AOSD, logging is often seen as the prime example for a crosscutting concern.

ConcernModule. One concern typically is realized by one or more *concern modules*. The term concern module is reused from Kojarski et al. [KL06] and encompasses a set of *concern elements* that together realize a concern or part of a concern (cf. role *concernElement* in Figure 3.1). Thus, it forms a representation of concerns in a formalized language (e.g., a package in UML or in Java). Some approaches have introduced the concept "aspect" [vdBCC05] for modularizing otherwise crosscutting concerns, while existing units of modularization formalizing non-crosscutting concerns have been called ″base″ [HOT02]. This distinction has been used to categorize aspect-oriented approaches into asymmetric approaches to concern composition that support this aspect-base dichotomy and symmetric ones that do not [HOT02]. Today, this distinction has begun to diminish and is being replaced by the more general understanding that the difference between concern modules is in how they are used during composition (cf. *ConcernCompositionRule*) [KL06]. In this thesis, this view is adopted and consequently in the CRM only the *concern module* concept, which subsumes the notions of aspect and base, is considered.

CompositionPlan. The integration of concern modules is specified by a *composition plan* [KL06], which consists of a set of rules. The weaving plan concept of Kojarski et al. [KL06] has been renamed in favor of the more general term composition, which yields the integration of multiple modular artifacts into a coherent whole [vdBCC05]. The ″execution″ of a composition plan results in a composed model of the overall system. During this process one distinguishes two phases, namely detection and composition. While detection is necessary to identify the concern elements that have to be integrated in the composed model, composition means the actual integration of them. For the purposes of this survey, furthermore a distinction between two ways of composing concern modules is made, namely static and dynamic (cf. attribute *isDynamic*). Thereby static indicates that the composed model is produced and thus is available to the modeler at design time analogously to compile-time weaving at programming level. Dynamic composition integrates the concern modules virtually during run-time, i.e., while executing the models. At the modeling level this requires the run-time semantics of the language's metamodel to be specified [FRGG04] (which, considering, e.g., UML, is only the case for parts of the language like state machines). This is similar to a run-time weaving that happens at programming level.

ConcernCompositionRule. The composition plan consists of a set of *concern composition rules* whereby one rule defines in detail how the various concern elements are to be composed. The general concept of concern composition rule is specialized into sub-classes according to the composition mechanism used. Following Kojarski et al. [KL06], the CRM foresees three composition mechanisms. Two asymmetric composition mechanisms exist in the form of pointcut-advice for introducing aspectual behavioral (e.g., intercepting method calls) and open class for introducing aspectual structure (e.g., introducing additional attributes to a class) [KL06]. At the modeling level, in any case augmentations or constraints need to be introduced with respect to model elements, whether they are behavioral elements or structural elements. Consequently, the *asymmetric composition rule* serves to realize both composition mechanisms. The compositor mechanism is provided by the sub-class *SymmetricCompositionRule*.

Module Interaction. Concern modules might be defined in a way such that they interact with each other. Kienzle et al. [KYX03] present a classification of interaction into ″orthogonal″ concern modules, ″uni-directional preserving″ concern modules based on other concern modules without modifying them, and ″uni-directional modifying″ concern modules that change other concern modules. Another classification of Sanen et al. [STW^{+}06] distinguishes between ″mutual exclusive″ concern modules, concern modules ″depending″ on each other, concern modules positively ″reinforcing″ each other, and concern modules "conflicting" with each other. Accordingly, the abstract class *ModuleInteraction* can be specialized to represent the specific interaction types. In case of a conflict, additional resolution strategies may need to be employed.

RuleInteraction. Analogously to module interaction, also concern composition rules may interact with each other. Again a *rule interaction* can be refined accordingly to support different kinds of rule interactions. For example, concern composition rules on the one hand may reinforce each other but on the other hand may also conflict with each other. Consequently, conflict resolution strategies need to be employed. In the context of UML, e.g., a relative or absolute ordering of rules could by realized with dependencies.

Effect. The *effect* specified with the concern composition rule describes what effect the integration of concern elements have. A concern composition rule may have an *enhancement* effect, a *replacement* effect, or a *deletion* effect (cf. *EffectKind* in Figure 3.1). This distinction resembles a differentiation proposed by Hanenberg [Han05] in terms of constructive (cf. enhancement), and destructive (cf. replacement and deletion) effects. There exists, however, an inherent relationship between the effect and the respective concern elements used in a particular rule. For example in case of a pointcut-advice rule, the relative positions *before*, and *after* may lead to an enhancement, whereas in case of *around* the effect may resemble an enhancement, a replacement (i.e., deleting the join point with the advice), or a deletion (i.e., deleting the join point with an empty advice).

3.1.2 AsymmetricConcernComposition

In the *asymmetric concern composition* the concern composition rule is specialized for covering the pointcut-advice and open class composition mechanisms [KL06]. The package is organized into two sub-packages, namely *AspectualSubject* and *AspectualKind*, due to the two distinct roles concern elements play in asymmetric composition.

AsymmetricCompositionRule. *Asymmetric composition rules* are part of a particular composition plan and provide support for the pointcut-advice and the open class composition mechanisms. An asymmetric composition rule consists of a pointcut (cf. *Pointcut*) together with an optional relative position (cf. *RelativePosition*) describing where to augment or constrain other concern modules as well as the advice (cf. *Advice*) describing how to augment or constrain other concern modules. The *consists-of* relationships have been modeled using weak aggregations, since advice, pointcut, and relative position might be reused in other asymmetric composition rules as well.

3.1.2.1 AspectualSubject

The *aspectual subject* describes the concepts required for identifying where to augment or constrain other concern modules.

JoinPoint. According to [FECA05] a join point is a well-defined place in the structure or execution flow of a program where additional behavior can be attached. In contrast, at modeling level the *join point* represents a well-defined place in a model represented by a concern module, which specifies where an advice (cf. *Advice*) can be introduced. Thus, a join point represents a concern element, i.e., an identifiable element of the language used to capture a concern. It has to be noted that in recent works [KL06], [MK03] the notion of join point has changed. It has been described as being a concept of the result domain, meaning it represents the composed element through which two or more concerns may be integrated. Nevertheless, the original concept of join point is essential to the pointcut-advice composition mechanism and may also be used in the open class composition mechanism [KL06]. Consequently, the CRM adheres to the original notion of join point for describing the concepts participating in asymmetric concern composition.

According to Hanenberg [Han05], join points can be distinguished along two orthogonal dimensions, namely abstraction and dynamicity. In this thesis, this categorization is applied to the modeling level while adhering to UML terminology [OMG05d] through the use of the term ”feature” instead of ”abstraction”. Consequently, join points can be structural (cf. *StructuralJoinPoint*) or behavioral (cf. *BehavioralJoinPoint*), while at the same time, they are also modeling level representations of static or dynamic elements (cf. attribute *isDynamic*) in a software system. While *static* join points are elements of a language that can be identified based on information available at design time (e.g., class and method call), *dynamic* join points are elements of a language that cannot be identified before run-time (e.g., object and method execution).

StructuralJoinPoint. *Structural join points* represent structural elements of a language where an advice can be introduced. In addition, structural join points can be either *static* or *dynamic* (cf. *isDynamic* attribute). Exemplifying those two categories by means of UML modeling elements, *structural-static join points* would be classes and *structural-dynamic join points* would be objects.

BehavioralJoinPoint. Analogous, *behavioral join points* represent behavioral elements of a language where an advice can be introduced. Additionally, a distinction is made between *behavioral-static join points* (e.g., activity) and *behavioral-dynamic join points* (e.g., method execution). Admittedly, not all languages may offer elements which allow for dynamic join points as is the case with UML together with OCL.

JoinPointModel. The *join point model* defines the kinds of join points available [vdBCC05]. It comprises all elements of a certain language where it is allowed to introduce an advice (cf. *Advice*), i.e., where the *representedAs* association connects the element with *JoinPoint*. For example, some approaches might want to restrict their join point model to a specific set of language elements, e.g., classifiers in UML.

Pointcut. A *pointcut* describes a set of join points [vdBCC05], i.e., the concern elements selected for the purpose of introducing certain augmentations or constraints (cf. *Advice*). The selection of join points can be done by means of quantified statements over concern modules and their concern elements (cf. *SimplePointcut* and *QuantificationMethod*). A pointcut specification is implemented by either a *SimplePointcut* or a *CompositePointcut*.

SimplePointcut. A *simple pointcut* represents a set of join points of a certain kind (e.g., structural-static), which are selected according to a certain quantification method (cf. *QuantificationMethod*). It thus, represents a means for selecting several concern elements as join points. For this survey, the combination of simple pointcut and quantification method correspond to the definition of pointcut in [vdBCC05].

CompositePointcut. For reuse purposes, pointcuts can be composed of other pointcuts by means of logical *Operators*, e.g., AND, OR, NOT, to form *composite pointcuts*. Thereby, all children of a composite pointcut, i.e., all selected join points, refer to the same join point model.

QuantificationMethod. The *quantification method* concept describes a mechanism, e.g., a predicate for selecting from the potential join points of the join point model those that should be available for introducing an advice (cf. *Advice*). The quantification method corresponds to what is termed a pointcut designator in AspectJ, i.e., its quantification mechanism according to [FECA05].

RelativePosition. A *relative position* may provide further information as to where aspectual features (cf. *Advice*) are to be introduced. It represents some kind of location specification. This additional information is necessary in some cases when selecting join points by pointcuts only is not enough. Such aspectual features can be introduced for example *before* or *after* a certain join point. Still, in some other cases such as for the open class composition mechanism a relative positioning is not necessary, e.g., when a new attribute is introduced into a class the order of the attributes is insignificant (cf. multiplicity 0..1). While the relative position typically is specified with the advice such as in AspectJ, in the CRM it is modeled separately from the advice. The ″wrapping″ technique presented in [FECA05] corresponds to the definition of relative position but in contrast is described for behavioral join points only.

3.1.2.2 AspectualKind.

The *AspectualKind* package comprises the concepts necessary to describe how to augment or constrain other concern modules.

Advice. An *advice* specifies how to augment or constrain other concerns at join points matched by a pointcut [vdBCC05]. An advice is realized by either a structural advice (cf. *StructuralAdvice*), a behavioral advice (cf.*BehavioralAdvice*), or both, i.e., by a composite advice (cf. *CompositeAdvice*). Historically, structural advice has been called ″introduction″, while behavioral advice has been termed ″advice″. Recently, the advice concept is more and more used as an inclusive term for both and consequently has been employed herein.

StructuralAdvice. A *structural advice* comprises a language's structural elements for advising other concerns. For example, adding a new attribute to a class's structure represents a structural advice.

BehavioralAdvice. Likewise, a *behavioral advice* comprises a language's behavioral elements for advising other concerns. In the context of UML, adding a method call, i.e., a message in a sequence diagram represents a behavioral advice.

CompositeAdvice. For reuse purposes, an advice can be composed of a coherent set of both, structural and/or behavioral advice, to form a *composite advice*, i.e., the composite needs to be free of conflicts. For example, an attribute and an operation represent two simple advice. If composed, the composite advice includes the attribute as well as the operation. In this respect, the advice concept extends the general understanding of the advice concept described in [vdBCC05].

3.1.3 SymmetricConcernComposition

In the *symmetric concern composition* the concern composition rule is specialized according the compositor composition mechanism [KL06].

SymmetricCompositionRule. A *symmetric composition rule* comprises first, a specification of the elements to be composed (cf. *ComposableElement*), second, the match method to apply upon them describing which elements to compose (cf. *MatchMethod*), and third, the integration strategy to be applied describing how to proceed on those matched elements (cf. *IntegrationStrategy*). For example in the context of UML such a symmetric composition rule could specify that classes of two packages having identical names shall be matched and their class bodies shall be combined, similarly to the UML ”package-merge” operator. Again, for reuse purposes the *consists-of* relationships have been modeled using weak aggregations.

ComposableElement. *Composable elements* of a symmetric composition rule refer to the elements allowed to be composed [Cla02]. Composable elements can be made up by any element of the underlying language. Therefore, a distinction is made also between composable structural elements (cf. *ComposableStructuralElement*) and composable behavioral elements (cf. *ComposableBehavioralElement*). In the course of a symmetric composition rule more than two of such elements can be integrated.

ComposableStructuralElement. A *composable structural element* comprises a language’s structural elements (cf. *StructualElement*) and can be composed with other composable elements identified in a symmetric composition rule. Examples for composable structural elements with respect to UML are Components, Classes, but also more fine-grained concepts such as Properties.

ComposableBehavioralElement. Likewise, a *composable behavioral element* comprises a language’s behavioral elements (cf. *BehavioralElement*) and can be composed with other composable elements identified in a symmetric composition rule. With respect to UML, examples for composable behavioral elements are Activities and Actions as well as State machines and States.

MatchMethod. The *match method* applied in the detection phase of a composition identifies which concrete elements to match given as input the composable elements for the composition. It supports the specification of match criteria for composable elements and their components, e.g., a class’s attributes. Examples for match methods found in literature [Cla02], [RGR$^+$06] comprise match-by-name, match-by-signature, no-match.

IntegrationStrategy. The *integration strategy* details how to proceed during composition with the matched elements. The general concept of integration strategy is specialized into the subclasses *merge*, *bind*, and *override* [Cla02], [RGR$^+$06].

Merge. With the *merge* integration strategy two or more corresponding composable elements are merged. This set of corresponding composable elements has been identified by the applied match method.

Override. In contrast to the merge integration strategy, for applying the *override* integration strategy the overriding as well as the overridden elements have to be specified from the set of corresponding composable elements identified by the applied match method.

Bind. The *bind* integration strategy typically represents a strategy where some composable elements are treated as template parameters that need to be bound to concrete values, i.e., other composable elements. It is applied in the context of parameterizable concern modules which are often used to realize crosscutting concerns.

3.1.4 Language

Finally, the concepts which are part of the *Language* package describe the means underlying the specification of concerns.

Language. Concern modules are formalized using the language *elements* of a certain *language*, i.e., a modeling language like UML. Depending on the composition mechanism used, some aspect-oriented approaches have distinguished between different languages for formalizing crosscutting and non-crosscutting concerns [KL06].

Element. A language comprises a set of *elements*, like e.g., class, relation, package which allow the modeler to express certain concepts. Typically a language's *elements* can be distinguished into structural (cf. *StructuralElement*) and behavioral elements (cf. *BehavioralElement*). Depending on the composition mechanism, the elements of a language are used differently. With respect to asymmetric approaches, elements serve two distinct purposes. First, they may represent join points and thus in the role of join points specify where to introduce an advice. Second, elements of a language are used for formulating the advice itself. In the case of symmetric approaches such a distinction is not made.

StructuralElement. *Structural elements* of a language are used to specify a system's structure. Natural examples for such elements in the case of UML are classes, packages, and components.

BehavioralElement. Likewise to structural elements, *behavioral elements* of a language are used to specify a system's behavior. Behavior is expressed in UML through behavioral elements like actions, states, and messages.

3.2 Evaluation Set-Up

3.2.1 Selection of Approaches

There already exists a considerable amount of proposals for AOM languages each of them having different origins and pursuing different goals dealing with the unique characteristics of aspect-orientation. Only few of them have come of age and have been presented at acknowledged conferences and journals, however. Since aspect-orientation is often considered an extension to object-orientation, it seems almost natural to use and/or extend the standard for object-oriented modeling, i.e., the Unified Modeling Language (UML), for AOM. To the best of our knowledge, there are only a few AOM proposals that do not base their concepts on UML [SR05], [SVWJ05] compared to the amount of approaches that do. Thus, this survey focuses on UML-based approaches to aspect-oriented modeling, only.

In literature, fourteen such well-published, UML-based, design-level AOM approaches have been identified, namely: [CB05], [CM06], [CvdBE07], [EAB05], [FPT07], [Gru00], [HJPP02], [JN05], [KK06], [KHJ06], [PSD+05], [RGR+06], [SHU02a], and [vFGC04]. In this survey, the results of evaluating a representative set of eight AOM approaches are presented, including in the set first of all those two approaches that have not been investigated in existing surveys, namely: [CvdBE07] and [KHJ06]. As indicated before, the rationale behind choosing the remaining six [CB05], [EAB05], [JN05], [PSD+05], [RGR+06], [SHU02a] out of the identified, is to assort a representative mix of approaches. In this respect, the goal has been to maintain the ratio between approaches based on metamodel extensions and those relying on UML Profiles as well as the ratio between symmetric and asymmetric approaches.

3.2.2 Catalogue of Evaluation Criteria

In the following, a catalogue of criteria for a structured evaluation of AOM approaches is proposed. The focus in designing this catalogue of criteria was to provide a fine-grained catalogue of criteria which constitutes the prerequisite for an in-depth evaluation of existing approaches and thus allows to compare different AOM approaches in greater detail than in previous surveys [BBR+05], [CRS+05], [dbTB+06], [RTT04]. The criteria of the evaluation framework have been derived in a top-down manner from the CRM (cf. Section 3.1) as well as in a bottom-up manner considering related AOM surveys:

Deriving Criteria from the Conceptual Reference Model. The CRM presented in the previous section sketches the concepts that have been identified to be important for the AOM domain. This has been done both at an abstract level, i.e., abstracting from different composition mechanisms, and at a detailed level, i.e., looking at the specific characteristics of each composition mechanism. Corresponding criteria in the catalogue operationalize the CRM with respect to allowing a comparison of approaches. In particular, for each concept of the CRM one or more criteria have been derived. This implies that either a concept of the CRM maps onto one-to-many criteria in the catalogue or one-to-many concepts of the CRM map onto one criterion in the catalogue. A concept that is represented as an abstract class, however, does not necessarily need a corresponding criterion in the catalogue, since it is implicitly evaluated by its sub-concepts and their criteria.

Collecting Criteria from other Surveys. Following a bottom-up approach, the goal was to complement the set of criteria by those used in related AOM surveys [BBR+05], [CRS+05], [dbTB+06], [RTT04]. More specifically, criteria definitions found in other surveys have been adopted or refined. In this respect, a refinement for instance has been the provision of a measurement scale, e.g., the UML version used for the *Language* criterion (cf. Section 3.2.2.1), or the decomposition of a criterion into several sub-criteria, e.g., the composability criterion of [CRS+05] has been refined for each composition mechanism in this survey.

Excluding Non-Measurable Criteria. From the catalogue of criteria a few criteria proposed in related surveys have been explicitly excluded, since they cannot be measured without user studies or extensive case studies. These include the following criteria of the survey of Blair et al. [BBR+05], i.e., reusability, comprehensibility, flexibility, ease of learning/use, parallel development, as well as change propagation, which corresponds to the evolvability criterion of Chitchyan et al. [CRS+05].

Establishing a Schema for Criteria Definition. Furthermore, the goal was to avoid blurred criteria by working out, as far as possible, unambiguous definitions and the criteria's values that are also measurable. Thus, each criterion is described by a set of properties:

1. a *name* along with an *abbreviation* allowing to reference the criteria during evaluation of the approaches in Section 3.3,

2. a *reference to the source* in case a criterion has been adopted or refined from another survey as

well as an explanation of how such a refinement has been accomplished,

3. a *definition* specifying the criterion as unambiguously as possible along with an optional *discussion* on difficulties in defining the criterion,

4. an appropriate *means of measurement*, such as a list of possible values or a measurement scale, including *not applicable* as a default value for each criterion.

Categorizing the Selected Criteria. The criteria of the catalogue have been grouped into six categories (see Figure 3.2) with four out of them being specifically inferred from corresponding parts in the conceptual reference model (cf. Section 3.1) and the general categories *Maturity* and *Tool Support* providing mainly descriptive criteria.

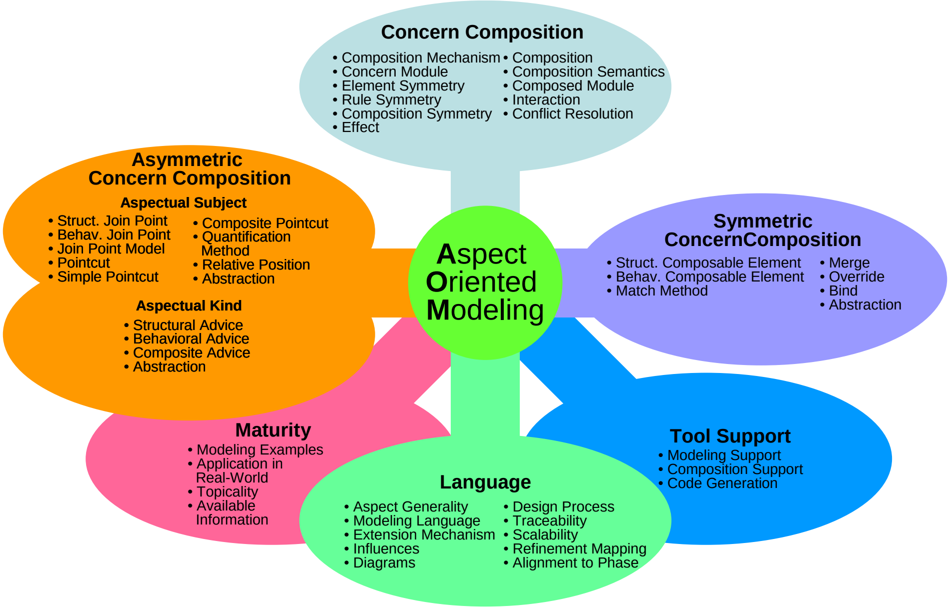


Figure 3.2: Categorization of Criteria

The *Language* category provides criteria for evaluating some basic characteristics of AOM languages (e.g., the modeling language, the extension mechanism used, and traceability). Beyond, it also provides a criterion for checking the availability of a design process. In the *ConcernComposition* category, the representation of the *concern module* concept and the composition mechanisms used is considered amongst others. With respect to symmetric concern composition in the *SymmetricConcernComposition* category, the kind of *composable elements* and provided *integration strategies* are investigated. In contrast, the *AsymmetricConcernComposition* category subsumes criteria for the *join point* and its sub-concepts (cf. *AspectualSubject* sub-category) as well as criteria evaluating the modeling support of *advice* (cf. *AspectualKind* sub-category). The *Maturity* of an approach is discussed along the criteria of provided modeling examples, real-world applications, and available information. And finally, in the *Tool Support* category the availability of tools for modeling and composing concern modules as well es for code generation is evaluated. Since a thorough evaluation of *Tool Support* for AOM would go beyond the scope of this survey, tool support is evaluated on the basis of the available literature, only. Following, each categories' criteria are presented.

3.2.2.1 Language

This category contains general criteria describing the modeling language and design process. A separate criteria for evaluating the *element* concept described in the CRM (cf. Section 3.1) is not considered, since it is implicitly evaluated with several other criteria that investigate the corresponding CRM's AO concepts with respect to their modeling representation.

Aspect Generality (M.AG) Besides being a general-purpose modeling language with respect to the application domain, an AOM approach also may be general-purpose with respect to aspects. The following two forms of *aspect generality* can be distinguished: A *general-purpose* AOM language supports modeling of all kinds of aspects, whereas an *aspect-specific* modeling language considers one specific aspect, only. Theoretically, there could be modeling languages that support two, three or more specific aspects. Still, these are not considered to be aspect-specific, since in that case, the definition for general-purpose modeling languages gets blurred. The aspect generality criterion has been adopted from the ”purpose” criterion in *Reina et al.* [RTT04]. In this survey, the focus is on general-purpose AOM languages, thus, also the aspect generality criterion is used for selection purposes, only.

Modeling Language (M.L) With respect to the modeling language used, UML-based AOM approaches are considered, only. Therefore, a distinction between the underlying UML version, i.e., version *1.x*[2], and version *2.0* [OMG05d] is made.

Extension Mechanism (M.E) Although UML is very expressive, its modeling mechanisms do not provide for aspect-oriented concepts. Thus, AOM proposals tend to use one out of two UML extension mechanisms to cater for the necessary modeling mechanisms. First, by what is called *heavy-weight extension*, the UML metamodel itself is extended through inheritance and redefinition of metamodel elements. Second, UML profiles, grouping user-defined extensions to metamodel elements in terms of stereotypes [RJB05], represent UML's built-in *light-weight extension mechanism*, which permits only extensions that do not change the metamodel. This way a new dialect of UML can be defined in order to better support specific platforms or domains [OMG05d]. The light-weight extension mechanism fosters tool interoperability [RJB05], since they are designed in a way that tools can store and manipulate the extensions without understanding their full semantics. This criterion has been inspired by *Chitchyan et al.* [CRS$^+$05], where this kind of information has been provided but an explicit criterion has not been defined therein.

Influences (M.I) Originally, the intention was to use ”platform dependency” as a criterion for this catalogue. Still, in literature, no clear definitions of platform or platform (in)dependence, e.g., in the context of OMG's Model Driven Architecture (MDA) [OMG03], have been available. For example, there may be many abstraction levels between MDA's Platform Independent Models (PIM) and Platform Specific Models (PSM). Consequently, what defines platform and platform-independence is a matter of objectives and has to be determined in the context of one's own work. In this survey, a common definition of platform for the evaluated approaches is not attempted. Instead, the ”inspired by” criterion of *Reina et al.* [RTT04] is resumed, according to which many of the AOM approaches have been inspired by concepts expressed in a specific aspect-oriented programming language. In contrast to [RTT04], this criterion is not restricted to AOP platforms but *lists research areas* (e.g., SOP,

[2] http://www.omg.org/technology/documents/vault.htm#modeling

MDSoC, and CF) *and platforms* in general that have "influenced" a particular approach. In addition, platforms are also listed if models can be mapped onto them, provided that proof is given through a mapping definition or at least appropriate examples.

Diagrams (M.D) The emphasis in modeling concern modules can be its structure and/or its behavior. In this respect, the kinds of supported structural and/or behavioral diagrams to specify aspect-orientation are evaluated. Hence, this property *lists all UML diagram types and possibly proprietary diagram types* that have been used to support on the one hand *structural* and on the other hand *behavioral* modeling of concern modules. This criterion also has been inspired by *Chitchyan et al.* [CRS$^+$05], where this kind of information has been provided but an explicit criterion has not been defined.

Design Process (M.DP) A design process describes a well-defined, step-wise approach to modeling. This criterion has been adopted from *Op de beeck et al.* [dbTB$^+$06] and evaluates if the surveyed AOM approach provides *explicit* support for a design process or if some *implicit* design process support is available, e.g., in terms of guidelines, only.

Traceability (M.T) The traceability criterion is defined as a property of a relationship between two models where one model is a refinement of another, and has been adopted from the work of *Chitchyan et al.* [CRS$^+$05]. More specifically, the criterion distinguishes between external and internal traceability. The external traceability measure focuses at aspect-oriented design models in relation to the full software development life cycle, i.e., requirements (R), analysis (A), design (D), and implementation (I). Possible values are combinations such as $R \rightarrow D \rightarrow I$, which means traceability from a requirements specification over design to the implementation level. The internal traceability measure deals with traceability between models belonging to one phase in the software development life cycle. In this survey, AOM approaches are investigated if during design more abstract design models are refined into more detailed design models. This sub-criterion evaluates to *supported* or *not supported*, respectively.

Scalability (M.S) Scalability, which is defined as the ability to cope with small as well as large modeling projects, is investigated with respect to first, which *high-level modeling elements* of an approach support scalability, e.g., UML packages, and/or *high-level diagram types*, and second, if scalability has been *proven or not proven* in real-world projects or by modeling examples that go beyond the composition of two concern modules. This definition of scalability has been refined from *Chitchyan et al.* [CRS$^+$05] with respect to its measurement scales.

Refinement Mapping (M.R) The refinement mapping criterion is adopted from *Op de beeck et al.* [dbTB$^+$06]. It describes how the refinement of an initial abstract design model into a more detailed one is achieved. One can distinguish the *extending* step-wise refinement from the *creating* step-wise refinement. The difference between these two possibilities is that for the latter a new instance of the model is created with every step in the refinement process.

Alignment to Phase (M.A) Design is just a phase embedded in the overall software development life cycle. An AOM approach therefore may be more aligned to certain phases in the software development than to others. Ideally, an approach is balanced between the abstraction available from the requirements phase and the abstraction needed for the implementation phase. An AOM approach can thus be aligned to *requirements* and/or the *implementation* phases but also to none of the phases. This criterion has been adopted from *Op de beeck et al.* [dbTB+06].

3.2.2.2 ConcernComposition

This category considers criteria derived from the corresponding package in the CRM, amongst others, the representation of the *concern module* concept, the composition mechanisms used as well as the approaches symmetry.

Composition Mechanism (CC.M) The concepts described in the CRM support the pointcut-advice (*PA*), open class (*OC*), and compositor (*CMP*) composition mechanisms. This criterion therefore allows to evaluate which of the three composition mechanism is realized by the AOM approaches. It is also possible to support more than one composition mechanism.

Concern Module (CC.CM) This criterion investigates the concern modules's representation in the modeling language in terms of a UML *meta-class* or a *stereotype* definition and, if provided, the notational element used.

Element Symmetry (CC.ES) Two possible ways of concern decomposition can be distinguished, namely, *symmetric* and *asymmetric* concern decomposition. In the asymmetric paradigm one distinguishes between concern modules of different structure, i.e., between ”aspects” and ”base”. As an example some AOM approaches, introduce a new stereotype ≪aspect≫ derived from UML meta-class *Class* to distinguish ”aspects” from normal ”base” classes. In the symmetric paradigm no such distinction is made. In fact, the symmetric paradigm treats all concerns, both crosscutting and non-crosscutting, as ”first-class, co-equal building-blocks of identical structure” [HOT02].

Rule Symmetry (CC.RS) The rules for composing concern modules can be specified in a *symmetric* or in an *asymmetric* way [HOT02]. In particular, the symmetry is determined by the placement of the concern composition rules. Rule asymmetry defines the concern composition rules within one of the concern modules that are to be composed (e.g., in AspectJ the rules are captured within the aspect in terms of pointcut-advice combinations), whereas rule symmetry defines them in neither of the concern modules. Please note, that rule symmetry corresponds to relationship symmetry in [HOT02].

Composition Symmetry (CC.CS) This criterion has been adopted from the work of *Op de beeck et al.* [dbTB+06] and investigates which concern modules are allowed to be composed with each other. While in the *asymmetric* case composition happens between ”aspects” and ”bases” only, i.e., ”aspects” are woven into ”bases”, in the *symmetric* case all concern modules can be composed with each other. For those approaches supporting element asymmetry and thus distinguishing between ”aspects” and ”bases”, symmetric composition is only supported if the following combinations are allowed: aspect-base, aspect-aspect, base-base. Approaches supporting element symmetry accordingly also support composition symmetry.

Effect (CC.E) This criterion evaluates if the approaches provide means for modeling the effect of the integration of concern elements via concern composition rules. The criterion's possible values are *supported* or *not supported*.

Composition Semantics (CC.S) The composition semantics criterion has partly been inspired by the survey of *Chitchyan et al.* [CRS+05], though not explicitly defined therein. This criterion evaluates if the composition semantics have been *defined* or *not defined* for both the detection of the elements to be composed as well as for their actual composition into a composed element.

Composition (CC.C) A distinction between composing concern modules *statically* or *dynamically*, i.e., by executing the models, is made. Nonetheless, a specific approach might neither support static nor dynamic composition at modeling level but defer weaving to later phases in the software development process, e.g., by separately generating code from concern modules, which are finally composed by a dedicated mechanism of the underlying AOP language. The advantages of approaches that support model composition are first, at code level non aspect-oriented platforms can be used and second, the composite results can be validated prior to implementation. However, once composed, the concern modules cannot be recovered at later stages thus causing traceability problems.

Composed Module (CC.CP) This criterion evaluates the resulting composed module in terms of its modeling representation. In particular, this criterion distinguishes between composed modules represented with *standard UML* and composed modules represented based on the *extensions made to the UML*. The composed module criterion has been adopted from the ”composability” criterion of *Chitchyan et al.* [CRS+05].

Interaction (CC.I) An AOM approach may offer ways to specify *interactions* between *concern modules* on the one hand but also between *concern composition rules* on the other hand. This criterion evaluates for both concepts, what *kind of interactions* can be modeled and the *modeling representations* thereof, e.g., UML meta-class or stereotype.

Conflict Resolution (CC.CR) In accordance with [BBR+05], conflict resolution may be based on a mechanism to *avoid* conflicts in advance or to *detect* conflicts and then *resolve* them manually. While conflict avoidance might be a possible solution to cope with conflicting aspects, one still might need ways to detect and resolve conflicts that could not be captured by conflict avoidance in advance. In case no conflict resolution has been specified, this criterion evaluates to *not supported*.

3.2.2.3 AsymmetricConcernComposition

This category subsumes criteria for evaluating approaches following an asymmetric way to concern composition which are categorized into two sub-categories *AspectualSubject* and *AspectualKind*.

ASPECTUALSUBJECT. The *AspectualSubject* sub-category provides criteria for evaluating concepts used to describe where to augment or constrain other concern modules, e.g., the *join point* and its sub-concepts.

Structural Join Point (AS.SJP) This criterion evaluates if structural join points are *supported*. More specifically, the focus is on what kind - with respect to dynamicity - of structural join point are considered in the approaches, i.e., structural-static join points like classes or structural-dynamic join points like objects.

Behavioral Join Point (AS.BJP) Likewise, the behavioral join point criterion evaluates if behavioral join points are supported by the surveyed AOM approaches. In this respect, examples for a behavioral-static join point are UML activities and messages. Behavioral-dynamic join points typically depend on certain conditions evaluated at run-time. Specifying such conditions can be done for example with OCL.

Join Point Model (AS.JPM) This criterion distinguishes between two possible ways of specifying a join point model. First, the join point model can be made *explicit* by identifying a language's model elements as join points. This can be achieved for example by enhancing the language's metamodel in a way that certain model elements inherit from a join point meta-class or by at least identifying and declaring the join points of a language in "natural language" such as in [SHU02c] or [Tea05]. Second, the join point model can be defined *implicitly* by the AOM language's join point selection mechanism, thus, comprising all join points that the join point selection mechanism is able to select.

Pointcut (AS.P) Although the pointcut concept is represented as an abstract class in the CRM (cf. Section 3.1), a separate criterion is foreseen for evaluating the commonalities of the concrete pointcut sub-classes. In particular, the criterion evaluates if the pointcut mechanism has been realized based on a *standard* (e.g., AspectJ code, UML, OCL) or on a *proprietary* language.

Simple Pointcut (AS.SP) This criterion evaluates how simple pointcuts are represented by concepts of the modeling language or extensions thereof and particularly distinguishes between *graphical* and *textual* representations of simple pointcuts.

Composite Pointcut (AS.CP) Furthermore, the composite pointcut criterion evaluates if at all and how composite pointcuts are represented in the modeling approach. Again, a distinction is made between *graphical* and *textual* representations of composite pointcuts.

Quantification Method (AS.SM) This criterion evaluates which quantification methods are employed to select join points in a certain approach. The selection of join points can be specified *declaratively*, *imperatively*, or simply by *enumeration*.

Relative Position(AS.RP) This criterion investigates the general support of specifying a relative position with respect to join points and, if provided, lists the different possibilities of relative position specification, i.e., *after*, *before*, and *around*, supported by the approaches.

Abstraction (AS.A) This criterion is refined from the definition given in *Chitchyan et al.* [CRS$^+$05]. In contrast, in the context of asymmetric concern composition, two dimensions of abstraction are considered, namely *abstraction with respect to the aspectual subjects (AS.A)* and *abstraction concerning the aspectual kind (AK.A)* (cf. Section 3.2.2.3). With respect to the aspectual subjects, a *high* level of abstraction means that the join points might not have been identified yet, i.e., the model only specifies the fact that a certain concern module affects others, but not exactly where. On the contrary, modeling languages providing a *low* level of abstraction allow specifying the exact points where advice take effect.

AspectualKind. The *AspectualKind* sub-category subsumes criteria for evaluating concepts used to describe how to augment or constrain other concern modules, e.g., the *advice*, as well as the abstraction level at which modeling of the *advice* is possible.

Structural Advice (AK.SA) This criterion evaluates if AOM approaches provide ways of specifying structural augmentations and/or constraints. Furthermore, the *concepts or extensions* of the modeling language as well as the *notational elements* used for representation are investigated.

Behavioral Advice (AK.BA) Likewise to structural advice, this criterion evaluates if AOM approaches provide ways of specifying behavioral advice and in particular what *concepts or extensions* of the modeling language and what *notational elements* have been used for representation.

Composite Advice (AK.CA) In addition to evaluating structural and behavioral advice support, the focus is on how the approaches provide ways of composing multiple pieces of advice to form a more complex advice in terms of *concepts or extensions* of the modeling language and appropriate *notational elements*.

Abstraction (AK.A) This criterion has been refined from the definition given in [CRS$^+$05] for the asymmetric concern composition. It is decomposed into the criteria *abstraction with respect to the aspectual subjects (AS.A)* (cf. Section 3.2.2.3) and *abstraction concerning the aspectual kind (AK.A)*. Analogously to (AS.A), it specializes the criterion given in [CRS$^+$05] for the aspectual kind and, as already mentioned, contributes to the overall evaluation of the abstraction of an approach. Since models at a high level of abstraction might be incomplete with respect to providing a specification for code generation, a *high* level of abstraction with respect to the aspectual kind means that it might not yet be clear *how* the specific concern module(s) should be advised, i.e., the model only specifies that a certain concern module exists, but not the actual advice it provides. In contrast, *low*-level models of aspectual kind refer to models that provide detailed information on how the concern module's internals (i.e., the actual advice and auxiliary functionality) look like.

3.2.2.4 SymmetricConcernComposition

This category subsumes criteria for evaluating approaches following a symmetric way to concern composition, i.e., the necessary concepts identified in the CRM as well as the level of abstraction at which modeling is supported.

Structural Composable Element (S.SCE) This criterion evaluates if and what structural composable elements are supported by an AOM approach. It *lists the UML meta-classes representing structural concepts* that in a symmetric concern composition approach can be composed.

Behavioral Composable Element (S.BCE) Likewise, the behavioral composable element criterion evaluates if and what behavioral composable elements are supported by an AOM approach. This criterion therefore *lists the UML meta-classes representing behavioral concepts* that in a symmetric concern composition approach can be composed.

Match Method (S.MM) This criterion evaluates which method(s) to identify the matching elements out of the set of composable elements are foreseen by an approach. It distinguishes between three possible methods, namely *match-by-name*, *match-by-signature*, and *no-match*.

Merge (S.M) This criterion investigates if AOM approaches supporting the symmetric concern composition provide ways of defining the specific integration strategy *merge*. In particular, it investigates what *concepts or extensions* of the modeling language as well as what *notational elements* have been used for representation.

Override (S.O) Similarly, this criterion checks if an AOM approach allows for modeling symmetric concern composition rules with an *override* integration strategy. Again, it also investigates what *concepts or extensions* of the modeling language as well as what *notational elements* have been used for representation.

Bind (S.B) Like the previous two, the *bind* criterion evaluates possible *extensions* of the modeling language to support such a binding and provides information on the *notational elements* used.

Abstraction (S.A) Analogous to the abstraction criteria for the asymmetric concern composition, this criterion has been refined from the definition given in [CRS$^+$05]. In this context, however, the level of abstraction is defined with respect to the composable elements used in a symmetric composition rule. A *high* level of abstraction is supported if the symmetric composition rule is used to compose two or more higher-level or composite modeling elements, such as UML packages, of which their internals have not been specified. A *low* level of abstraction is provided, if these composite modeling elements can be detailed, e.g., a class diagram for a UML package, and if symmetric composition rules can also be specified for more fine-grained modeling elements such as UML attributes.

3.2.2.5 Maturity

The criteria in this section intend to evaluate the approaches' maturity in general. It has to be noted that in [BBR$^+$05] the criterion maturity was used to evaluate whether an approach has been used in real world examples, only, whereas in this survey maturity is evaluated with a set of sub-criteria described in the following.

Modeling Examples (Ma.E) Besides evaluating the breadth of modeling examples, it is also interesting to investigate the modeling examples' depth in terms of how many different concern modules are integrated within the examples. Thus, the criterion is supported by two values, namely, the *number of provided modeling examples by each* approach as well as the *maximum number of concern modules* integrated in one example.

Application in Real-World Projects (Ma.A) The successful deployment of the AOM approach in the *design of a real-world application* proves its applicability and consequently indicates a high level of maturity of the modeling concepts. Possible values are *yes*, and *no*.

Topicality (Ma.T) The topicality criterion presents for each approach when the *most recent piece of work* in terms of the year of publication has been published to indicate the approach's topicality and thus, gives an indication whether the approach might still evolve or not. This criterion has been refined from the "year" criterion of *Reina et al.* [RTT04]

Available Information (Ma.I) Another measure of the approaches' maturity is the available *amount of manuals, papers* and *books*. Although, admittedly, the amount of publications does not necessarily correlate with an approach's quality. The values for this criterion provide the *number* of different resources of information.

3.2.2.6 Tool Support

Tool Support improves the adoption of an approach an developer productivity as well as ensures syntactical correctness of the model. While the criterion distinguishes between support for modeling, composition and code generation, the latter are both dependent on modeling support.

Modeling Support (T.M) Modeling support is defined as providing the means to use the modeling language's notation and furthermore of validating the created aspect-oriented models for syntactical and semantical correctness. If the modeling language is realized in terms of a UML profile, modeling support should be portable to any UML modeling tool. This criterion evaluates to *supported*, possibly providing further information on modeling support, or *not supported*.

Composition Support (T.C) This criterion specifies if composition of concern modules is also *supported* or *not supported* by a tool an thus allows to view and/or simulate the composed model.

Code Generation (T.G) In line with the concepts of MDE, code generation facilities should be provided, thus requiring a mapping between the notation and the supported implementation language. This criterion evaluates if code generation, in principle, is possible. Beyond, this criterion also evaluates if there is a more sophisticated mechanism to code generation such as the OMG's MDA [OMG03] (i.e., existence of platform-independent models, platform definition models and their transformation into platform-specific models by using a mapping mechanism). Thus, possible values for this criterion are *supported* or *not supported*. Additional information is provided in case of a more sophisticated code generation mechanism.

3.2.3 Modeling Example: The Observer Pattern Applied to a Library Management System

3.2.3.1 Motivation

As an appropriate running example of a crosscutting concern to be applied to a system, in this evaluation, the well-known observer pattern [GHHV04] is adopted, a prominent example not only in AOSD literature (cf. [CW05], [PZ03], [SHU02a]) but also in software engineering literature. In the running example, the observer pattern is applied as a crosscutting concern to a library management system, of which an overview along with the underlying model is given in the following. It has to be emphasized that on the basis of this rather simple example it is not (and cannot be) the intention to illustrate each and every concept of the approaches but rather to foster their overall understandability and comparability.

3.2.3.2 An Example Library Management System

In Figure 3.3, the *Library* package models the structure for managing books of a library in a library management system based on [CW05]. Of course, it only depicts a small excerpt of such a system, primarily containing those parts of the system that are crosscut by the observer concern.

A *BookManager* manages a list of *Books* (cf. *addBook(Book)* and *removeBook(Book)*) allowing users to search (cf. *searchBook(Book)*) the list and access provided information for each book (e.g., *authors*). A library may offer several copies of each *Book*, i.e., the physical entities (cf. *BookCopy*),

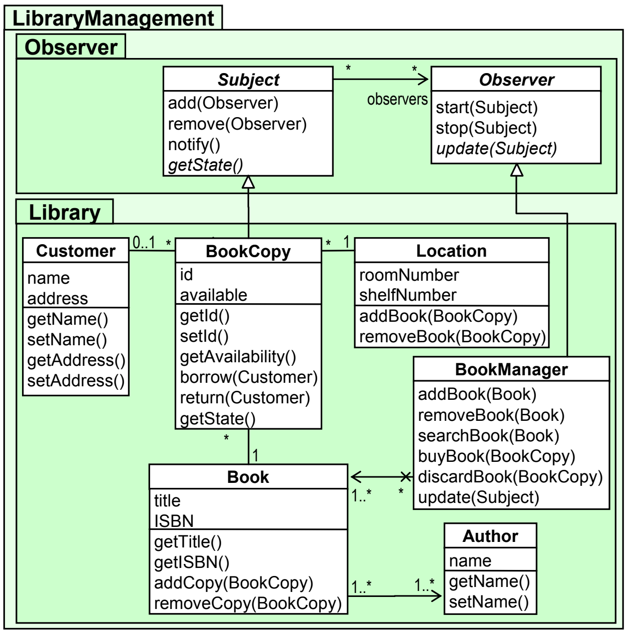


Figure 3.3: The Library Management System with Observer Aspect.

which need to be managed accordingly. *BookCopies* might get lost or be stolen. Still, a *Book* does not have to be removed from the *BookManager*'s list until new *BookCopies* are obtained. The *BookManager* associates *BookCopies* with their *Books* as they are bought (cf. *buyBook(BookCopy)* and *addCopy(BookCopy)*) and likewise, disassociates them as they are discarded (cf. *discardBook(BookCopy)* and *removeCopy(BookCopy)*). *Books*, in particular their copies, have a *Location* on a certain shelf in a certain room of the library. The status of each *BookCopy*, i.e., its availability, should be kept up-to-date. Thus, each time a *BookCopy* is borrowed or returned by a *Customer* (cf. *borrow(Customer)* and *return(Customer)*), the *BookManager* has to be notified. This notification functionality is not provided by the library management system, but is applied using the observer pattern as depicted in Figure 3.3.

3.2.3.3 The Observer Pattern

The observer pattern [GHHV04] as depicted in the *Observer* package in Figure 3.3 defines a one-to-many dependency between objects in a way that whenever a *Subject* (i.e., a *BookCopy*) changes its state, all its dependent *Observers* (i.e., instances of *BookManager*) are notified (cf. *notify()*) by using their provided update interface (cf. *update(Subject)*). While *Observers* can register and unregister with their *Subjects* of interest using the methods *start(Subject)* and *stop(Subject)*, a *Subject* keeps a list of *Observers* (cf. *add(Observer)* and *remove(Observer)*), which are interested in changes of the *Subject*'s state.

In Figure 3.3, thus, the *Subject* and *Observer* roles are adopted by *BookCopy* and *BookManager*, respectively. Applying the observer pattern, however, affects the library management system's modularity. In particular, the abstract methods *getState()* and *update(Subject)* have to be implemented by *BookCopy* and *BookManager*, respectively. Additional code modifications are necessary to call *start(Subject)*/*stop(Subject)* whenever a *BookCopy* is bought/discarded and to call *notify()* whenever a *BookCopy* is borrowed or returned. Therefore, the observer functionality can be regarded as crosscutting concern and, thus, be realized with the concepts of various AOM approaches.

3.2.3.4 Limitations of the Running Example

The observer pattern is a well-known example for a crosscutting concern and actually has been used in three of the surveyed approaches (cf. [Cla02], [FKGS04], [SHU02a]). One might argue that the use of an example which has already been used by some of the analyzed approaches might lead to a bias in the evaluation. Since the running example is used to only visualize the respective approach to the reader and to have a side by side comparison of the approaches, any biased influence on the survey itself is negligible. Still, some approaches do not allow for fully operationalizing the running example, which is due to their particular focus. For instance, the approach of Klein et al. [KHJ06] does not allow to model crosscutting structure, since the approach's focus is rather on a weaving algorithm for (non-)crosscutting behavior. Nevertheless, the application of one running example for all approaches generated some insight into the differences of each individual approach. Of course all AOM approaches should be tested in a real world setting or at least in a non-trivial example, which encompasses more than two concerns as well as all concepts described in the conceptual reference model, e.g., the *AO Challenge* [KG06]. Such an example would allow for testing the approaches' means for capturing interactions and resolving conflicts, which in this survey, can only be described textually. Still, the obvious advantages of a small and easy to understand running example would be lost.

3.3 Comparison of Approaches

This survey is based on a literature study, including modeling examples, provided by the individual AOM approaches. For each surveyed approach, additional information and discussion is provided in the following. The evaluation of each approach follows the order of categories of the criteria catalogue presented in Section 3.2. Moreover, a running example (cf. Section 3.2.3) that is modeled by means of the concepts of each AOM approach is provided. This further enhances the evaluation in that it first, provides an insight into each approach and second, allows to easier compare the modeling means of the approaches.

In the following, the modeling means of each surveyed AOM approach is presented by means of this running example. Basically, the approaches realizing the *pointcut-advice* and *open class* composition mechanisms are presented first and then those realizing the *compositor* composition mechanism are elaborate on. In particular, the first two approaches of Stein et al. (cf. Section 3.3.1), Pawlak et al. (cf. Section 3.3.2), are similar, since they have been specifically designed as modeling languages for two aspect-oriented programming platforms, i.e., AspectJ and the JAC Framework[3] respectively. The commonalities of the third approach of Jacobson et al. (cf. Section 3.3.3) and the approach of Pawlak et al. are that they do not envisage composition of concerns at modeling level but defer composition to the implementation phase. The next two approaches, are both very recent proposals to AOM focusing on composing behavioral diagrams. In this respect, the approach of Klein et al. (cf. Section 3.3.4) presents an algorithm for first, detecting

[3] http://jac.objectweb.org/

the model elements to be composed and second, composing them. The approach of Cottenier et al. (cf. Section 3.3.5) also supports composition of models and, in contrast to all others, already comes with tool support for modeling, composition and code generation. The last group of three approaches supports the *compositor* composition mechanism. While the approach of Aldawud et al. (cf. Section 3.3.6) focuses on the composition of state machines, the approaches of Clarke et al. (cf. Section 3.3.7) and France et al. (cf. Section 3.3.8) originally have considered the composition of class diagrams. Lately, the approach of France et al. also realizes the pointcut-advice composition mechanism through the composition of sequence diagrams. The results of the comparison are discussed and illustrated in Section 3.4 *Lessons Learned* at a glance (cf. Table 3.1 to 3.7).

3.3.1 The Aspect-Oriented Design Model, Stein et al.

LANGUAGE

The *Aspect-Oriented Design Model (AODM)* of Stein et al. [SHU02a], [SHU02b], [SHU02c] has been developed as a design notation for AspectJ (L.I) and thus is aligned to implementation (L.A) as well as allows for external traceability from design to implementation (L.T). Internal traceability is not applicable (L.T), since a refinement of models models is not forseen in *AODM* (L.R). For this approach, both AspectJ and UML have been studied in order to find corresponding parts for AspectJ's concepts in UML and extend it to support AspectJ's concepts if necessary as well as identify where UML's concepts used in *AODM* are more expressive than actually necessary, e.g., the destruction of an instance is not part of AspectJ's join point model [SHU02a]. *AODM* is basically specified using the UML 1.x light-weight extension mechanism (L.L), (L.E), though extensions of the metamodel have also been necessary. For example, the UML extend relationship from which the ≪crosscut≫ stereotype has been derived originally can be specified between use cases, only [SHU02b]. Structural and behavioral modeling is achieved by employing class diagrams, UML 1.x collaborations, and sequence diagrams. In addition, sequence diagrams are used for visualizing join points, e.g., messages, while use case diagrams and collaborations demonstrate AspectJ's composition semantics (L.D). In the *AODM* thus, UML is used such that scalability in terms of high-level modeling elements is not supported and no other proof in terms of non-trivial modeling examples is available (L.S). The approach furthermore does not outline a design process or provide guidelines (L.DP).

CONCERNCOMPOSITION

AODM represents a notation designed for AspectJ and consequently supports the pointcut-advice and open class composition mechanisms (CC.M) as well as follows the asymmetric school of thought (CC.ES), (CC.CS), (CC.RS). A distinct concern module for crosscutting concerns has been introduced in *AODM* and is represented by a stereotype ≪aspect≫ (cf. *SubjectObserver-ProtcolImpl* in Figure 3.4[4]), which is derived from the UML meta-class *Class* (CC.CM). In addition, several meta-attributes capture the peculiarities of AspectJ's aspects, e.g., the instantiation clause. The composition actually is deferred until the implementation phase (CC.C). Nevertheless the composition semantics of AspectJ have been redefined for the modeling level to a limited extent, e.g., in terms of UML use case diagrams and collaborations (CC.S), (CC.CP) [SHU02a]. The only way for modeling interactions (CC.I) is to manually specify the order for composing ≪aspects≫ in terms of a stereotyped dependency relationship between ≪aspects≫, i.e., ≪dominates≫ for conflict resolution (CC.CR) [SHU02b]. A means for explicitly specifying the effects of the concern composition rules or rather of the advice in models, however, is not addressed in *AODM* (CC.E).

[4] Please note that, in AspectJ the Observer functionality is realized using interfaces instead of abstract classes.

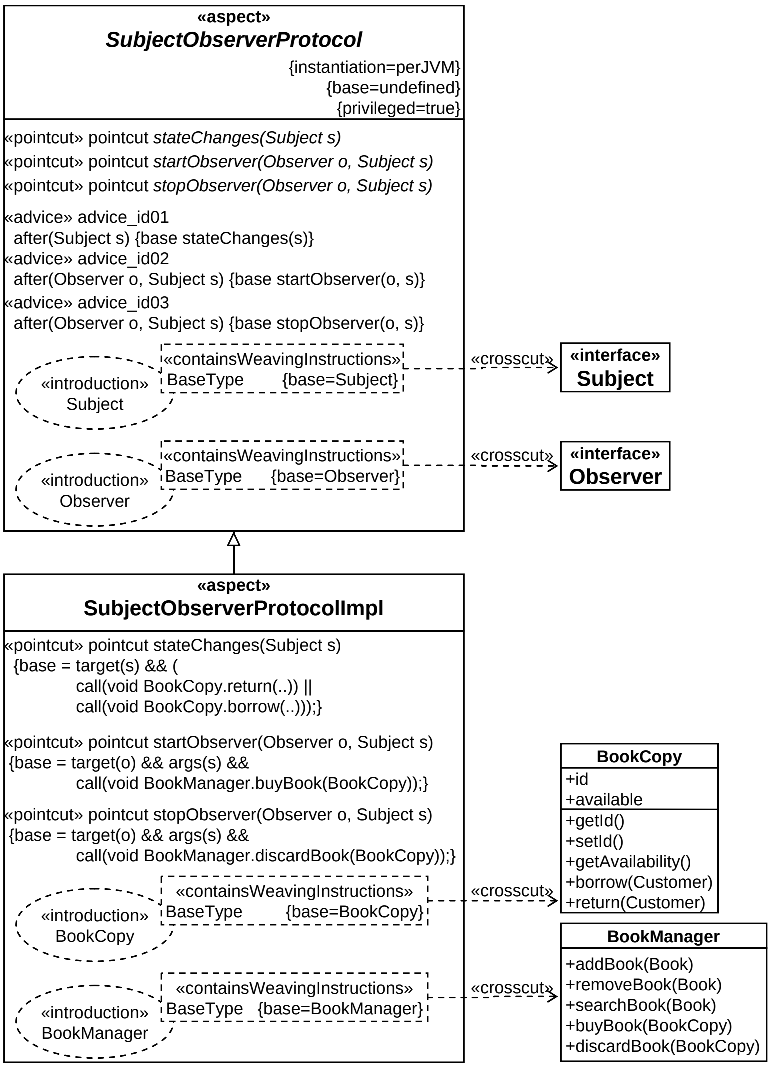

«aspect»
SubjectObserverProtocol
{instantiation=perJVM}
{base=undefined}
{privileged=true}
«pointcut» pointcut stateChanges(Subject s)
«pointcut» pointcut startObserver(Observer o, Subject s)
«pointcut» pointcut stopObserver(Observer o, Subject s)
«advice» advice_id01
after(Subject s) {base stateChanges(s)}
«advice» advice_id02
after(Observer o, Subject s) {base startObserver(o, s)}
«advice» advice_id03
after(Observer o, Subject s) {base stopObserver(o, s)}
«introduction» Subject
«containsWeavingInstructions» BaseType {base=Subject}
«crosscut»
«interface» Subject
«introduction» Observer
«containsWeavingInstructions» BaseType {base=Observer}
«crosscut»
«interface» Observer
«aspect»
SubjectObserverProtocolImpl
«pointcut» pointcut stateChanges(Subject s)
{base = target(s) && (
call(void BookCopy.return(..)) ||
call(void BookCopy.borrow(..)));}
«pointcut» pointcut startObserver(Observer o, Subject s)
{base = target(o) && args(s) &&
call(void BookManager.buyBook(BookCopy));}
«pointcut» pointcut stopObserver(Observer o, Subject s)
{base = target(o) && args(s) &&
call(void BookManager.discardBook(BookCopy));}
«introduction» BookCopy
«containsWeavingInstructions» BaseType {base=BookCopy}
«crosscut»
«introduction» BookManager
«containsWeavingInstructions» BaseType {base=BookManager}
«crosscut»
BookCopy
+id
+available
+getId()
+setId()
+getAvailability()
+borrow(Customer)
+return(Customer)
BookManager
+addBook(Book)
+removeBook(Book)
+searchBook(Book)
+buyBook(BookCopy)
+discardBook(BookCopy)

Figure 3.4: The Observer Aspect, Stein et al.

AsymmetricConcernComposition

AspectualSubject. Though *AODM* has been specifically designed as a modeling language for AspectJ, Stein et al. [SHU02c] extend their notion of a join point model (AS.JPM): UML Classifiers are identified as structural-static hooks (AS.SJP). Besides, UML 1.x Links represent behavioral-static join points. Behavioral-static join points are depicted by highlighted messages in sequence diagrams (see [SHU02a]) (AS.BJP). For those join points where no such messages exist (e.g., field reference, field assignment, initialization, execution) pseudo operations and special stereotypes have been provided. Using a ≪crosscut≫ dependency relationship, the subjects of structural advice are specified at a high level of abstraction (AS.A). The pointcuts in *AODM* are similar to AspectJ's pointcuts (AS.P). Selections of behavioral-static join points and behavioral-dynamic join points (AS.BJP) are represented by ≪pointcut≫ stereotyped UML Operations that are implemented by special ≪ContainsWeavingInstructions≫ stereotyped UML Methods. A meta-attribute ″base″ introduced for this ≪ContainsWeavingInstructions≫ stereotype then holds the pointcut in the form of AspectJ code (AS.P), (AS.QM). This allows first, the specification of composite pointcuts (AS.SP), (AS.CP), and second, the specification of the aspectual subjects at a low level of abstraction (AS.A). In addition, a second stereotype ≪ContainsWeavingInstructions≫ at this time derived from the UML meta-class TemplateParameter[5] is used to specify the pointcuts for structural advice (e.g., ≪introduction≫ *Subject* in Figure 3.4). The new meta-attribute ″base″ introduced for the ≪ContainsWeavingInstructions≫ stereotype specifies the pointcut in the form of AspectJ's type patterns. AspectJ's - and consequently *AODM*'s - means for specifying a pointcut is following a specific conceptual model. Recently, the authors have been working on a more expressive pointcut mechanism supporting different conceptual models [SHU06], which is independent from the *AODM* approach, however. Concerning the declaration of a relative position, *AODM* supports the relative positions before, after, and around for behavioral-dynamic join points, only, and depicts them in an AspectJ-like manner as a keyword in the signature of behavioral advice (AS.RP).

AspectualKind. In a class diagram, behavioral advice are depicted in the operation compartment of a class consisting of the operation's signature as well as a base tag containing the pointcut's signature. Behavioral advice in *AODM* are represented by stereotyped UML Operations, i.e., ≪advice≫. These are implemented by special ≪ContainsWeavingInstructions≫ Methods, which contain the actual behavior in the method's ″body″ meta-attribute and reference a pointcut in the introduced ″base″ meta-attribute (AK.BA). Additionally, behavioral advice are specified in terms of sequence diagrams. Thus, behavioral advice are modeled at a high as well as a low level of abstraction (AK.A) likewise structural advice are modeled at a high and low level of abstraction: Structural advice are realized as parameterized collaboration templates with the stereotype ≪introduction≫. The parameters are of type ≪ContainsWeavingInstructions≫, which specify the subjects of advice in the form of AspectJ's type patterns (AK.SA). The details of the collaboration templates are shown in Figure 3.5. Composite advice, since not a concept available in AspectJ, are not addressed by *AODM* (AK.CA).

MATURITY

AODM has been described in some publications (M.I). While the approach has not been tested in a real-world application (M.A), some modeling examples have been provided, e.g., timing and billing aspects for a system in the area of telecommunication [Tea05] and the realization of the observer pattern (M.E). Today, the authors have moved on and specifically focus on research towards graphical ways to select join points in UML . For this they have introduced join point designation diagrams (JPDD) [SHU06], which basically are UML diagrams (i.e., class and object diagrams, as well as, state charts, sequence, and activity diagrams) enriched with e.g., name and signature patterns, and wildcards. They represent an independent pointcut mechanism that can be applied to any UML-based AOM language, allows to select all kinds of join points (i.e., structural-static, structural-dynamic, behavioral-static, and behavioral-dynamic) as well as supports composite pointcuts (M.T).

TOOL SUPPORT

The approach claims rapid modeling support by a wide variety of CASE tools [SHU02b], which is due to using UML's light-weight extension mechanism. This is, however, questionable, since the authors also extended UML's metamodel (T.M). Both composition support and code generation support are currently not considered (T.C), (T.G).

[5] Stein et al. apparently have used the same name for two different stereotypes.

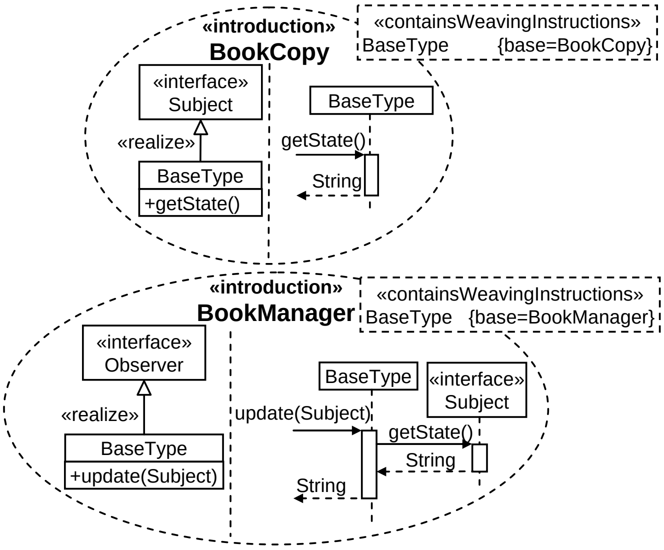


Figure 3.5: Structural Advice, Stein et al.

3.3.2 The JAC Design Notation, Pawlak et al.

LANGUAGE

The *JAC Design Notation* proposed by Pawlak et al. [PDF+02], [PSD+05] is mainly designed for the JAC Framework, i.e., an open source framework which includes a complete IDE with modeling support and serves as a middleware layer for aspectual components (L.I). Thus similar to the AODM approach of Stein et al., the *JAC Design Notation* represents an approach aligned to implementation as well as supporting external traceability from design to implementation (L.A), (L.T). Internal traceability is not applicable, since models typically are not refined in the approach (L.R), (L.T). The approach is based on light-weight UML extensions. Since it has been developed out of a pragmatic need to express crosscutting concerns in the JAC Framework, the authors do not claim full compliance with UML but aim at keeping it intuitive and simple (L.E). The authors provide no information on the UML version used. The extended UML metamodel in [PSD+05], however, indicates the usage of a UML version prior to version 2.0 (L.L). The approach relies on class diagrams, only (L.D). Consequently, scalability is not supported by the *JAC Design Notation* (L.S). Beyond, the approach provides neither a description of a design process nor guidelines (L.DP).

CONCERNCOMPOSITION

The *JAC Design Notation* realizes the pointcut-advice composition mechanism (CC.M). The stereotype ≪aspect≫ which is derived from the UML meta-class Class is used to represent crosscutting concern modules (CC.CM) (cf. *Observer* in Figure 3.6). Consequently, the approach follows the asymmetric approach with respect to elements (CC.ES). Since ≪aspects≫ are composed with normal classes only, the *JAC Design Notation* also supports composition asymmetry (CC.CS). With respect to concern composition rules (CC.RS), the design notation represents a symmetric approach using a UML Association stereotyped with ≪pointcut≫ (cf. AspectualSubject). Composition is not available at modeling level but deferred until implementation (CC.C) and therefore no composed model is available either (CC.CP). Consequently, the composition semantics are those of the JAC framework (CC.S). Both modeling of interactions (CC.I) as well as conflict resolution (CC.CR) are not addressed at all by the approach. There are five stereotypes derived from the UML meta-class Operation (cf. AspectualKind) which specify advice. The specification of effects is partly considered by one of them, i.e., the stereotype ≪replace≫ which provides for either a replacement or a deletion. All other stereotypes are considered to have an enhancement effect (CC.E).

AsymmetricConcernComposition

AspectualSubject. A join point model is explicitly defined in natural language, only [PSD+05] (AS.JPM) and join points are limited to method calls thus supporting behavioral-static join points, only (AS.BJP). Nevertheless, structural-dynamic join points are also supported via the ≪role≫ stereotype (cf. AspectualKind) (AS.SJP). The additional concept of "pointcut relation" corresponds to the asymmetric composition rule concept defined in the CRM (cf. Section 3.1). It is an association stereotyped with ≪pointcut≫. The association has a name and an optional tag to allow for adding extra semantics (cf. *stateChanged* in Figure 3.6). The rule connects the actual pointcut definition with the advice, i.e., the association ends contain information about the pointcut definition and the advice, respectively. Pointcuts are defined using a proprietary, textual language based on regular expressions and/or keywords (AS.P), (AS.QM), e.g., *!BookCopy.MODIFIERS* in Figure 3.6 selects as join points all method invocations (denoted with '!') of methods from class *BookCopy* that modify the object state (AS.BJP). Thus, the notation provides a low level of abstraction, while a high level of abstraction is not addressed (AS.A). The provided pointcut mechanism also allows composing simple pointcuts using operators, e.g., AND, OR, etc. (AS.SP), (AS.CP). Furthermore, the approach introduces the "group" concept supporting the design of distributed applications. ≪group≫ is depicted as a stereotyped class but is derived from UML meta-class ModelElement and subsumes arbitrary and probably distributed classes that might need the same set of advice. It is, thus, part of the pointcut mechanism. For example in the observer aspect, subjects, i.e., arbitrary "base" classes that have to be observed, might be distributed and can be abstracted within a ≪group≫ named Subject. In the library management system such subjects might represent other resources than books such as journals, CDs, etc. The relative position is specified for behavioral advice, only, by three out of the five stereotypes for advice, i.e., ≪before≫, ≪after≫, and ≪around≫ (S.RP).

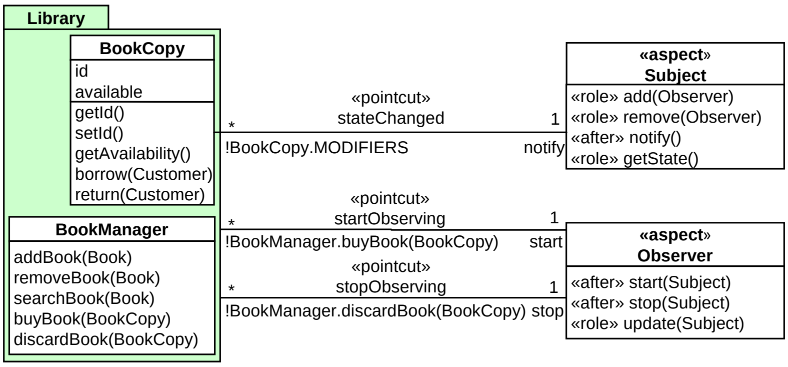


Figure 3.6: The Observer Aspect, Pawlak et al.

AspectualKind. Both behavioral and structural advice are represented as methods of ≪aspect≫ classes. The kind of advice is indicated by the stereotype of the advice operation. The stereotypes ≪before≫, ≪after≫, ≪around≫, and ≪replace≫ indicate behavioral advice, e.g., ≪after≫ *update()* in Figure 3.6 (AK.BA), whereas ≪role≫, i.e., the fifth stereotype for advice, represents a structural one (AK.SA). In the *JAC Design Notation*, structural advice which are implemented by ≪role≫ methods are not really added to the structure of the base class but can be invoked on the objects that are extended by the aspect, e.g., ≪role≫ *addObserver(BookManager)* can be invoked on *BookCopy* (cf. Figure 3.6). Moreover, these methods can access the extended class attributes and the attributes of the ≪aspect≫. Role methods therefore are similar to the "introduction" concept of AspectJ. Composite advice, in principle are possible within the JAC Framework through method composition. The *JAC Design Notation*, however, provides no means for explicitly modeling such composite advice (AK.CA). With respect to abstraction, the notation of Pawlak et al. represents predominantly a low level modeling approach, also with respect to advice, i.e., it shows aspect internals (AK.A).

MATURITY

The *JAC Design Notation* has already been well described (M.T), (M.I) and has been applied to several well-known aspects like caching, authentication, tracing, and session in the context of a simple client-server application but not in combination with each other. These examples generally do not greatly differ from each other and follow the same simple principals but show the applicability of the notation to any aspect in general (M.E). It has been tested in real industrial projects like an online courses intranet site, an incident reporting web site, and a business management intranet tool (M.A).

TOOL SUPPORT

The JAC Framework includes a complete IDE with modeling support. The provided modeling tools allow for designing base and aspect classes as well as their relations using the proposed UML notation (T.M). The IDE also supports code generation (i.e., Java) for the JAC framework (T.G). Weaving is supported at runtime (T.C) but not at design time.

3.3.3 Aspect-Oriented Software Development with Use Cases, Jacobson et al.

LANGUAGE

The approach of Jacobson et al. [JN05] represents a use case driven software development method that has been realized by extending the UML 2.0 metamodel (L.L), (L.E). *Aspect-Oriented Software Development with Use Cases (AOSD/UC)* comes with a systematic process that focuses on the separation of concerns throughout the software development life cycle, i.e., from requirements engineering with use cases down to the implementation phase (L.DP). Since the approach covers the whole software development life cycle, it is aligned to both the requirements and the implementation phase (L.A). Furthermore, the approach fosters external traceability between all phases through explicit ≪trace≫ dependencies between models (L.T). During the whole software development life cycle the approach makes use of different UML diagrams including use case diagrams in the requirements phase as well as class diagrams and communication diagrams in the analysis phase. For the design phase, component diagrams can be refined into class diagrams (L.T), (L.R), while sequence diagrams are used to model behavioral features (L.D). The language extensions reflect the influence by the Hyper/J and AspectJ languages (L.I). Scalability of the approach is supported with high-level modeling elements (i.e., ≪use case slice≫) and has been demonstrated with a non-trivial example (L.S).

ConcernComposition

Concerns are modeled with the ≪use case slice≫ stereotype, which is derived from the UML meta-class Package (CC.CM). At this level, the approach of Jacobson et al. supports element symmetry (CC.ES). Taking a closer look, however, the ≪use case slice≫ - inspired by the Hyper/J language - encapsulates modeling artifacts of one phase in the software development life cycle, i.e., concerns are kept separately until the implementation phase. In this evaluation, the focus is on slices produced during design, where the artifacts include classes, sequence diagrams and ≪aspect≫ classifiers as depicted in Figure 3.7. Consequently, at this level of abstraction the approach follows element asymmetry (CC.ES). Although, at first sight it seems that *AOSD/UC* supports the compositor composition mechanism on the basis of UML package merge, the internals of ≪use case slice≫ are such that they actually support the pointcut-advice and open class mechanisms (CC.M). In fact, composition is deferred until implementation (CC.C) thus, a composed module is not available at modeling level (CC.CP). Furthermore, the composition semantics seem to be those of AspectJ (CC.S), (CC.RS), (CC.CS). Since concerns are modeled as ≪use case slice≫, which represent use cases in the design phase, they inherit the relationships of use cases, i.e., inheritance, ≪extend≫ and ≪include≫ (CC.I). With respect to conflicting interactions, the approach follows a strategy that avoids conflicts through refactoring actions performed on models (CC.CR). The effects of composition are not modeled in *AOSD/UC* (CC.E).

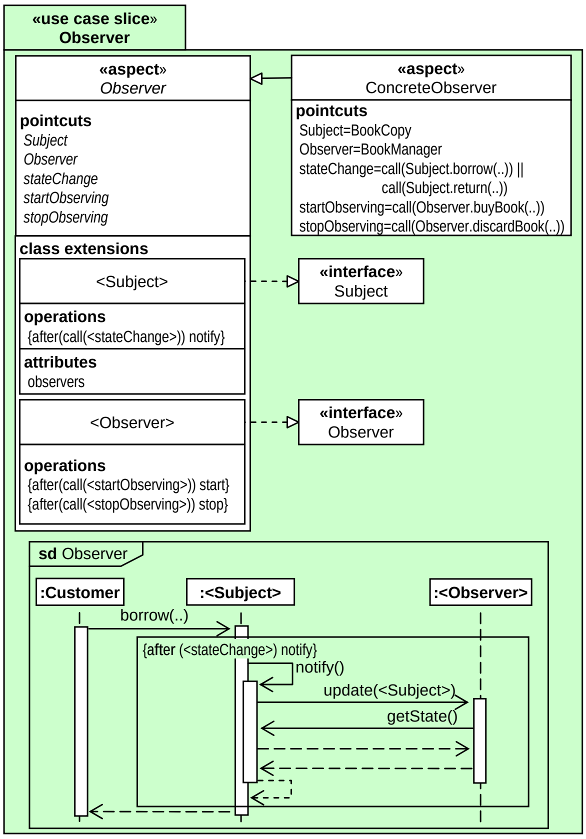


Figure 3.7: The Observer Aspect, Jacobson et al.

ASYMMETRICCONCERNCOMPOSITION

AspectualSubject. The join point model of the approach is similar to that of AspectJ and is implicitly defined by the pointcut mechanism used (AS.JPM). UML Classifiers are used as structural-static join points (AS.SJP), while behavioral-static and behavioral-dynamic join points are identified with the AspectJ pointcut language (AS.BJP), (AS.P). Consequently, the approach supports simple as well as complex pointcuts (AS.SP), (AS.CP). Pointcuts are specified in a separate ”pointcuts” compartment of an ≪aspect≫ classifier. If specified as abstract, pointcuts need to be defined in a concrete aspect such as depicted in Figure 3.7 with ≪aspect≫ *ConcreteAspect*. The pointcut then is specified with AspectJ code allowing for usage of name patterns, type patterns and wildcards (AS.QM), e.g., pointcut *stateChange* represents a complex pointcut quantifying behavioral-static join points and the pointcut *Subject* represents a simple pointcut quantifying a structural-static join point. The aspectual subject is thus modeled at a detailed level but also can be modeled at a higher level of abstraction, i.e., with component interfaces in component diagrams (AS.A). As an alternative, placeholders such as <*Subject*> and <*stateChange*> can be used for parameterizing the use case slice, similarly to the template-based approaches of Clarke et al. (cf. Section 3.3.7) and France et al. (cf. Section 3.3.8). As it is done in AspectJ, the relative position (i.e., *before*, *after*, or *around*) is specified with the advice. In Figure 3.7 the <*Subject*> classifier is extended with an operation which is to be executed *after* the pointcut <*stateChange*> matches a join point (AS.RP).

AspectualKind. Advice are modeled at a low level of abstraction, only (AK.A) and are detailed as ”class extensions” in a separate compartment of the ≪aspect≫ stereotype (cf. Figure 3.7 ≪aspect≫ *Observer*). As an example, for structural advice, the <*Subject*> classifier is extended with an attribute *observers* and an operation declaring the aspectual advice (cf. Figure 3.7). The fact, that the <*Subject*> classifier needs to implement the ≪interface≫ *Subject* is represented with a UML realization dependency (AK.SA). The behavioral advice is further detailed within sequence diagrams. So called ”frames” are used to insert aspectual behavior and are labelled with the signature of corresponding operations, such as {*after (<stateChange>) notify*} in Figure 3.7 (AK.BA). There is no way for modeling composite aspectual features (AK.CA).

MATURITY

The approach of Jacobson et al. has been recently elaborated in detail in three publications [JN05], although some of the ideas can be traced back to earlier works of the authors (M.I), (M.T). A hotel management system has been used as a comprehensive example encompassing several different concerns. The example is used to illustrate each phase in the software development life cycle (M.E). Still, no information of applications in real-world projects could be identified (M.A).

TOOL SUPPORT

The *AOSD/UC* approach does not come with tool support. Since composition is deferred to the implementation phase, composition support within a tool is not the authors' focus (T.C). Nevertheless, modeling the extensions made to the UML metamodel currently are not supported within a tool (T.M), neither is it possible to generate code for a specific AO platform (T.G).

3.3.4 Behavioral Aspect Weaving with the Approach, Klein et al.

LANGUAGE

The approach of Klein et al. [KHJ06] is originally based on Message Sequence Charts (MSC) a scenario language standardized by the ITU [Sec04]. UML 2.0 sequence diagrams have been largely inspired by MSCs. Thus, the approach can be applied to sequence diagrams as is shown in the KerTheme proposal [JKBC06] (L.L). No extensions to the UML sequence diagrams (or MSCs) have been made in this respect (L.E). Klein et al. have designed a ”weaving algorithm” for composing behaviors, i.e., scenarios modeled with UML sequence diagrams (or MSCs) (L.D). The composition is specified at modeling level regardless of any implementation platform (L.I). The approach does neither outline a design process nor guidelines (L.DP), since the goal is rather on complementing existing AOM approaches with a weaving mechanism for aspect behavior. The approach is thus not aligned to other phases in the software development life cycle either (L.A), nor are sequence diagrams further refined in the process of modeling (L.R). Consequently, the approach does neither provide means for supporting traceability (L.T) nor scalability (L.S)

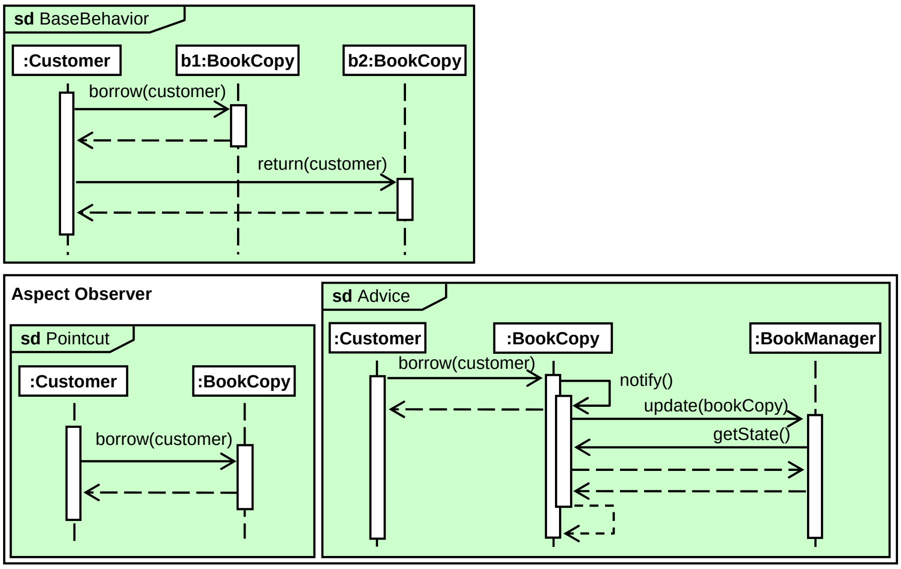


Figure 3.8: The Observer Aspect, Klein et al.

ConcernComposition

The approach of Klein et al. supports the pointcut-advice composition mechanism (CC.M). Modeling all behavior is achieved by means of sequence diagrams (CC.ES). Nevertheless, an aspect consists of two scenarios having distinct roles when used in a composition (CC.CM): one defines a part of behavior, i.e., the pointcut, that should be replaced by another, i.e., the advice (cf. Figure 3.8). This replacement is done every time the behavior defined by the pointcut appears in the semantics of the base scenario. In this respect the approach follows rule asymmetry (CC.RS). Concerning composition symmetry, however, the approach is symmetric, since behavior that once has served as advice or pointcut could serve as base behavior some other time or vice versa (CC.CS). The composition semantics are clearly defined by the two-phase weaving algorithm. In the first phase, join points are detected in the base behavior according to the pattern specified in the pointcut. In the second phase, the base behavior then is composed with the behavior specified in the advice (CC.S). Currently, models are composed statically, the implementation of the algorithm is, however, subject to future work (CC.C) The composition results again in a sequence diagram (CC.CP) as depicted in Figure 3.9. The effect in the approach is always replacement by definition, since the behavior detected via pointcuts is replaced by the advice behavior. Consequently, with the current weaving algorithm there is no need for specifying an effect (CC.E).

A means for specifying interactions (CC.I) and/or handling conflicts currently is not addressed but stated to be subject to future work (CC.CR).

AsymmetricConcernComposition

AspectualSubject. Join points in the approach of Klein et al. are sub-MSCs or a sequence of messages in a sequence diagram (AS.JPM) that match the sub-MSC or sequence diagram defined by the pointcut as is depicted in Figure 3.8 (AS.P), (AS.QM). Consequently, the approach's join point model supports behavioral-static join points, only (AS.BJP), (AS.SJP). The pointcuts are modeled at a detailed level, only (AS.A) and in principle can be composed using the sequential composition operator of basic MSCs [KHJ06], although no explicit concept for composite pointcut is available (AS.SP), (AS.CP). There is no need for specifying a relative position, since the behavior detected via pointcuts is always replaced by the advice behavior and this basically simulates the *around* relative position kind (AS.RP). It has to be noted that the pointcut used in the running example cannot fully illustrate the expressiveness of the weaving algorithm's join point detection mechanism. The algorithm allows to detect much more complex patterns [KHJ06]. For instance, it is easy to express a pointcut as a sequence of messages.

AspectualKind. Like pointcuts, aspectual behavior is modeled as MSCs thus only supporting behavioral advice (AK.BA), (AK.SA). In principle, they also can be composed using the sequential composition operator of basic MSCs, although no explicit concept for composite advice is available (AK.CA). Aspectual features in the approach are modeled at a detailed level, only (AK.A). The advice illustrated in Figure 3.8, shows the observer behavior being inserted after the *BookCopy b1* is borrowed by the *Customer*. As already pointed out above, the behavior specified by the pointcut, i.e., the *borrow(customer)* message, has to be modeled within the advice if it is to appear in the composed behavior (cf. Figure 3.9).

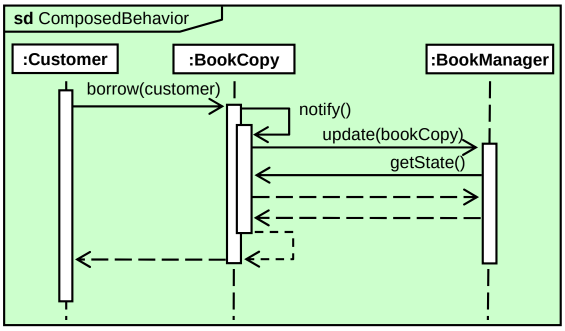


Figure 3.9: The Composed Model, Klein et al.

Maturity

The approach of Klein et al. has been described in several applications (M.I), where similar examples have been used, i.e., a login process. The focus has been on demonstrating the weaving algorithm's join point detection mechanism on the basis of complex behaviors, i.e., pointcuts, to be detected in the base behavior (M.E). In order to allow for testing aspect-oriented models, the approach recently has been combined with the Theme/UML approach of *Clark et al.* [JKBC06] (M.T). So far, the approach has not been employed in a real-world application (M.A).

TOOL SUPPORT

Modeling of scenarios is possible with tools either supporting MSC or UML 2.0 sequence diagrams (T.M). The authors are currently working on an implementation of the weaving algorithm within KerMeta [MFJ05] (T.C). Still, code generation currently seems not to be the authors' focus. Since weaving is supported at modeling level, existing code generation facilities could be reused (T.G).

3.3.5 The Motorola Weavr Approach, Cottenier et al.

LANGUAGE

The *Motorola Weavr* approach [CvdBE07], [CAE07] and tool has been developed in an industrial setting, i.e., the telecom infrastructure software industry. The modeling language of choice in this domain is the Specification and Decription Language (SDL) ITU recommendation [Sec02] of which features such as composite structure diagrams and transition-oriented state machines have been adopted in UML 2.0. The *Motorola Weavr* approach consequently is based on UML 2.0 and a light-weight profile that completes the UML specification towards the SDL and AOM concepts (L.L), (L.E). Besides class diagrams the approach makes heavily use of composite structure diagrams as a refinement of class diagrams (L.R), (L.T). The behavioral features of concerns are modeled using transition-oriented state machines (cf. Figure 3.10 (c), (e), (f)) and the SDL action language as well as sequence diagrams which are used as test cases. A special "deployment diagram" is used to direct the composition of concern modules (L.D), (L.S). Although targeted at telecommunications domain, the approach is platform-independent. Indeed, platform-specific details are encapsulated within several code generators and code optimizers that prohibit round-trip engineering (L.T). Consequently, the approach rather aims at composing concerns at the modeling level than drawing mappings onto platform-specific models (L.I). Nevertheless, the approach is aligned to the implementation phase (L.A). Furthermore, since Motorola uses the approach in production, one can infer that the approach supports scalability, which has already been proven with appropriate modeling examples (L.S). A design process for the approach has not yet been described, however. Likewise no guidelines are given (L.DP).

CONCERNCOMPOSITION

The *Motorola Weavr* approach supports the pointcut-advice composition mechanism (CC.M). Aspects are represented by the stereotype ≪Aspect≫ which is derived from the UML meta-class Class (CC.CM), (CC.ES). The approach puts forward rule asymmetry (CC.RS), since pointcuts and the binding to advices are modeled as parts of the aspect (cf. Figure 3.10 (d)). Furthermore, the approach also supports composition asymmetry, i.e., aspects can be woven into the base but not the other way round (CC.CS). The deployment of aspects to multiple base models as well as aspects can be modeled using the ≪crosscuts≫ stereotype derived from the UML meta-class Dependency. In order to resolve possible conflicts, the approach allows to define precedence relationships between aspects using a stereotype ≪follows≫, also derived from the UML meta-class Dependency. It has to be further noted, that precedence can also be defined at the level of concern composition rules again using the stereotype ≪follows≫ (CC.CR). Beyond, the approach defines two further dependency stereotypes for specifying interactions at concern module level, namely ≪hidden_by≫ and ≪dependent_on≫ [ZCVG06] (CC.I). These relationships are usually depicted in a separate diagram called "deployment diagram" (cf. Figure 3.10 (a)). The approach, however, does not offer a way to specify effects (CC.E). The Motorola Weavr tool supports the static weaving of aspects into base models. The composition semantics consequently is clearly defined.

More specifically, the approach distinguishes between a phase of detection (i.e., ”connector instantiation”) and of composition (i.e., ”connector instance binding”) (CC.S). The approach/tool, however, does not show the results of composition which are internally available in standard UML in a composed model (CC.CP). Instead, the modeler can simulate the composed model and view the specific base or aspect parts during execution (CC.C).

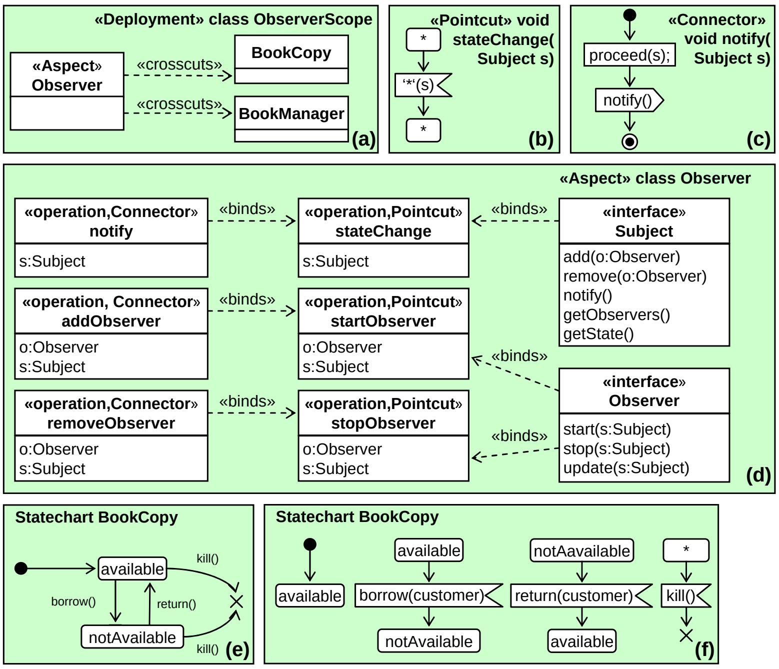


Figure 3.10: The Observer Aspect, Cottenier et al.

AsymmetricConcernComposition

AspectualSubject. The join point model consists of action join points including call expression actions, output actions, create expression actions, and timer (re)set actions as well as transition join points including start transitions, initialization transitions, termination transitions, and triggered transitions (AS.JPM). Consequently, the approach’s join point model supports behavioral-static join points (AS.BJP). Object instances represent structural-dynamic join points (AS.SJP). Pointcuts are defined using a stereotype which is derived from the UML meta-class Operation (AS.P), e.g., the pointcut ≪operation,Pointcut≫ *stateChange* in Figure 3.10 (d). The implementation of a pointcut is modeled as a transition-oriented state machine, which can be specified using wildcards (AS.QM) (cf. Figure 3.10 (b)). Consequently, the approach’s pointcuts are modeled at a high level of abstraction where only the pointcuts’ parameters are known and at a detailed level using state machines (AS.A). Beyond, pointcuts can be composed to form more complex pointcuts by means of AND and OR logical composition operators [CvdBE07] (AS.SP), (AS.CP). The relative position kind cannot be modeled but the approach supports the *around* relative position kind: Join points are always replaced by the advice behavior but can be called using an AspectJ-like *proceed()* action such as is depicted in Figure 3.10 (c) (AS.RP).

AspectualKind. Like pointcuts, behavioral advice are modeled using a stereotype derived from the UML meta-class Operation, e.g., the ≪Connector≫ *notify()* in Figure 3.10(d). A connector corresponds to AspectJ's advice and like pointcuts in the *Motorola Weavr* approach is implemented as a transition-oriented state machine (cf. Figure 3.10 (c))(AK.BA). Thus, advice in the approach are modeled both at a high and a low level of abstraction (AK.A). The connectors are bound to the pointcuts to which they shall be applied using a ≪bind≫ stereotype which is derived from the UML meta-class Dependency (cf. Figure 3.10 (d)). It has to be further noted that precedence can also be defined for connectors again using the stereotype ≪follows≫. Structural advice are modeled via interfaces, only (AK.SA). In the running example, the ≪Aspect≫ Observer introduces two interfaces *Subject* and *Observer*, which are bound to pointcuts of the aspect using the ≪bind≫ dependency (cf. Figure 3.10 (d)). The semantics of this relationship is that the interfaces are bound to the object instances that contain joinpoints for the specified pointcuts [CvdBE07], which is similar to the approach of Pawlak et al. (cf. Section 3.3.2). Consequently, the *Subject* interface is bound to the *stateChange* pointcut, while the *Observer* interface is bound to the other pointcuts. To the best of our knowledge, there exists no means for modeling composite advice (AK.CA).

MATURITY

The approach of Cottenier et al. represents one of the most recent approaches to AOM and has already been illustrated in several publications (M.I), (M.T). Besides a set of simple modeling examples, aspects covering exception handling, recovery, atomicity, and a two-phase commit protocol have been applied to a server-based communication system [CvdBE07] (M.E). The *Motorola Weavr* approach and tool is already being used in production (M.A) and is made available to academia under a free of charge license.

TOOL SUPPORT

The *Motorola Weavr* is designed as an add-in for the Telelogic TAU MDA tool[6] (T.M) and allows composing aspect models with base models as well as verification of the composed model via simulation (T.C). Starting from the composed model, existing code generation facilities such as the Motorola Mousetrap code generator [BLW05] can be used (T.G). The Motorola Weavr tool is already being deployed in production at Motorola, in the network infrastructure business unit [CvdBE07].

3.3.6 The AOSD Profile, Aldawud et al.

LANGUAGE

The *AOSD Profile* (L.E) of Aldawud et al. [AEB03], [EAB05] is based on UML version 1.x (L.L) and is aimed at being independent of any particular AOP language (L.I). While class diagrams are used to express the structural dependencies, state machines model the behavioral dependencies of concerns (L.D). The models are continuously refined from class diagrams to state machines (L.R). In order to do so, a set of guidelines for using the concepts provided by the *AOSD Profile* is offered (L.DP), which allows for external traceability from the requirements phase but not specifically for internal traceability (L.T), (L.A). The specific usage of state machines and their event propagation mechanism indicates that the approach does not support scalability and we are not aware of a modeling example proving the opposite (L.S).

CONCERNCOMPOSITION

In the *AOSD Profile*, crosscutting concerns have a separate representation in the form of the stereotype ≪aspect≫ (CC.CM), (CC.ES), which is derived from the UML meta-class Class (cf. Figure 3.11). Although, it is allowed to relate aspects to other aspects, each aspect has to be

[6] http://www.telelogic.com/products/tau/g2/

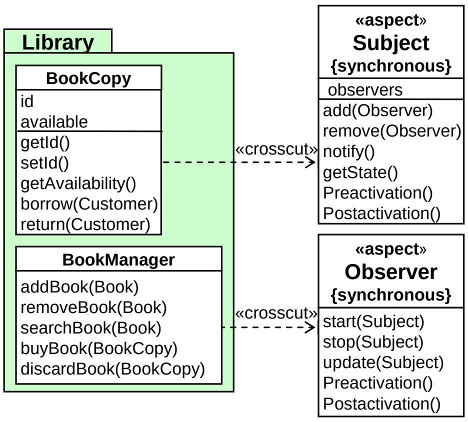


Figure 3.11: The Observer Aspect, Aldawud et al.

woven into at least one base class and, hence, this actually constitutes an asymmetric view of composing concerns (CC.CS). An integrated model view where aspects would already be woven into the base classes is not provided (CC.CP). Composition is rather deferred to implementation [MBAE04] (CC.C). At a more detailed level, one can see that the approach supports the compositor composition mechanism (CC.M): in the *AOSD Profile* approach concurrent state machines are used to model both non-crosscutting and crosscutting behavior in orthogonal regions, meaning element symmetry at the level of state machines (cf. Figure 3.12). More specifically, the composition semantics (CC.S) are ”specified” by the event mechanism used to indicate the flow of crosscutting behavior in state charts (CC.RS). The state machines thus implicitly express the composition semantics. The ≪crosscut≫ dependencies[7] between aspects and base classes as well as aspects and aspects dictate the ordering of events propagated in the orthogonal regions of statecharts (CC.I) (cf. Figure 3.11). Since the state charts allow for specifying the temporal sequence in the control flow of crosscutting behavior, i.e., an ordering of aspects, further conflict resolution is implicitly available (CC.CR). The effect of adaptations, however, cannot be modeled (CC.E).

SYMMETRICCONCERNCOMPOSITION

Composable elements in the approach of Aldawud et al. are elements of UML state machine diagrams, in particular events (S.BCE), whereas structural composable elements are not supported (S.SCE). Events trigger transitions from one state to another. The approach of Aldawud et al. makes use of broadcasting events to cause transitions in orthogonal regions of the same or other state machines, i.e., to activate other concerns. For example, the *observingBookManager* state machine in Figure 3.12 describes the behavior of the *BookManager* class. If a new *BookCopy* is bought (cf. *buyBook()*) the transition from state IDLE to state observing is triggered. This transition, however, triggers the transition from *IDLE* to the *startObservingSubject* state in the *observing* region. For the *observedBookCopy* state machine of the *BookCopy* class, this means a transition from state *notObserved* to state *observed*, given that the *BookCopy* has been in the state *notObserved*. The event mechanism of state machines allow to ”compose” the behavior of different concerns represented in orthogonal regions of state machines following a merge integration strategy (S.M). The corresponding elements, or rather events, are explicitly defined by using naming patterns. The ap-

[7] Please note, that the reading direction of the ≪crosscut≫ dependencies is different to the other approaches, e.g., *BookCopy* is *crosscut by* the aspect *Subject*.

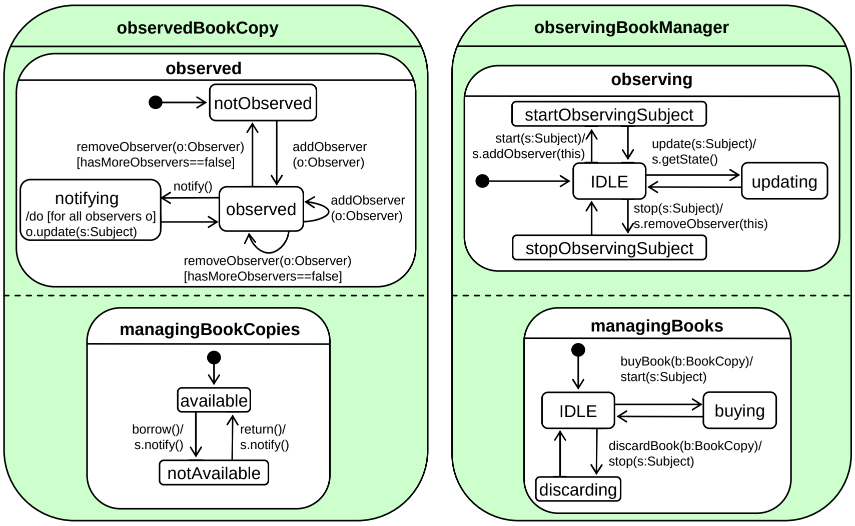


Figure 3.12: The Observer's Crosscutting Behavior, Aldawud et al.

proach thus supports a name-based match method, only (S.MM). With respect to the level of abstraction, the details captured by the state machines suggest a low level but not a high level of abstraction. Indeed, recently the use of the State pattern [GHHV04] has been used to translate the behavior captured within state machines to code [MBAE04] (S.A).

Maturity

Although the approach is described in several recent publications (M.I), (M.T), it is illustrated using a single example, the bounded buffer system, only. Still, it covers various aspects, namely, synchronization, scheduling, and error handling (M.E). A real-world application of the approach, however, is not available (M.A).

Tool Support

Due to using the UML profile extension mechanism, modeling support within the approach is available through existing UML modeling tools (T.M). Nevertheless, neither composition (T.C) nor code generation support have yet been addressed (T.G).

3.3.7 The Theme/UML Approach, Clarke et al.

Language

The *Theme* approach of Clarke et al. [CB05] provides means for AOSD in the analysis phase with *Theme/Doc*, which assists in identifying crosscutting concerns in requirements documents, and in the design phase with *Theme/UML*. In this survey, the focus is on *Theme/UML*, which is used in producing separate design models for each "theme" from the requirements phase (L.T), (L.A), i.e., the encapsulation of a concern representing some kind of functionality in a system [CB05]. *Theme/UML* is based on a heavy-weight extension of the UML metamodel version 1.3 (L.L), (L.E). It is designed as a platform-independent AOM approach, which originated from SOP [CHOT99], and evolved from the composition patterns approach of Clarke [Cla01], [Cla02] as well as provides mappings to AspectJ, AspectWerkz, and Hyper/J (L.I), (L.T), (L.A). Basically, *Theme/UML* poses no restrictions on what UML diagrams might be used for modeling. Nevertheless, particularly package and class diagrams are used for modeling structure and sequence diagrams are used for behavioral modeling (L.D). Theme/UML allows every concern to be refined separately and then to be composed into a new model (L.R). Scalability of the approach is supported by using UML packages for modeling concerns and has been demonstrated with non-trivial examples [CB05] (L.S). Beyond, the authors outline a design process for their modeling approach (L.DP).

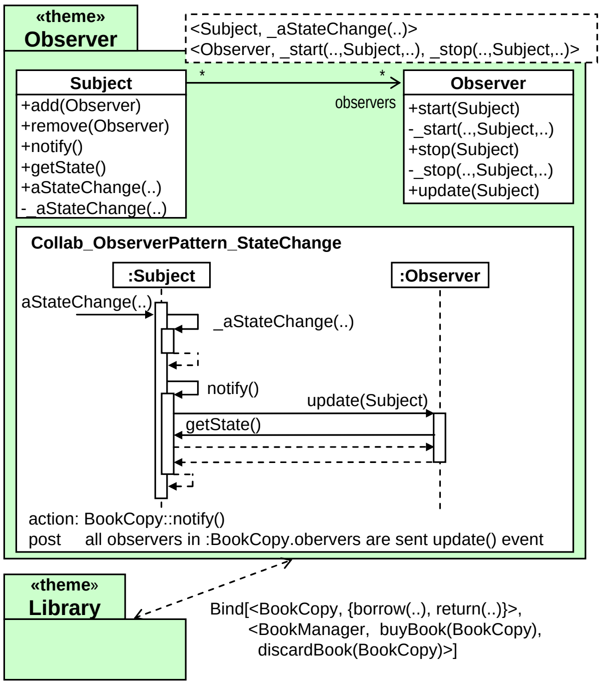


Figure 3.13: The Observer Aspect, Clarke et al.

ConcernComposition

Theme/UML realizes the compositor composition mechanism (CC.M). Concerns are encapsulated in UML packages denoted with a stereotype ≪theme≫ (cf. *Observer* and *Library* in Figure 3.13). Concern modules generally are modeled using standard UML notation. Crosscutting concerns, however, are realized using UML's modified template mechanism, which allows instantiating template parameters more than once, thus supporting multiple bindings (CC.CM), (CC.ES), (CC.CS). The composition semantics are clearly stated in [Cla01] for both how to detect the corresponding elements to be composed as well as for the actual composition itself (CC.S). A set of themes is composed statically into a composed ≪theme≫ (CC.C) as is shown in Figure 3.14, i.e., the *ObserverLibrary theme* is composed of the *Observer* and *Library themes* from Figure 3.13 (CC.CP). Besides relating two or more themes through *Theme/UML*'s ″composition relationships″, i.e., specialization from UML meta-class Relationship (CC.RS), there is no other way to model interactions between concern modules (CC.I). The composition relationships can also be used at a more fine-grained level, e.g., for specifying composition at the level of classes and attributes. Special attachments or ″tags″ to the *Theme/UML* composition relationships represent the conflict resolution mechanism. First, the ″prec″ tags define an ordering for theme precedence, with 1 indicating the highest precedence. Second, in case of a conflict the ″resolve″ tag allows specifying default values for elements of a certain type (e.g., visibility of attributes is private). And third, for a specific conflict the ”resolve” tag allows defining explicit composition output values. *Theme/UML* wants developers to first compose all non-crosscutting themes and then weave crosscutting themes one after the other into the composed model, thus forcing the developer to consider the ordering of crosscutting themes (CC.CR). The approach, however, does not provide modeling means for specifying the effects of composition (CC.E).

SYMMETRICCONCERNCOMPOSITION

Composable elements in Theme/UML are identified with the introduction of the new meta-class ComposableElements [Cla02]. More specifically, the UML meta-classes Attribute, Interaction, Collaboration (S.BCE), Operation, Association, Classifier, and Package (S.SCE) all inherit from the new meta-class and thus are allowed to be composed. For identifying the corresponding elements of two or more themes, the approach allows to tag the composition relationships with the ”match” tag. *Theme/UML*, currently supports two ways of matching composable elements, namely match-by-name and no-match. The latter states that the composeable elements participating in the composition relationship do not match (S.MM). Composition is catered for through three different integration strategies (specialization of UML meta-class Relationship), ”merge” (S.M) and ”override” (S.O), and ”bind” (S.B), which is a specialization of merge and allows composing crosscutting themes with non-crosscutting ones. This binding can be done for several themes. In Figure 3.13, the crosscutting ≪theme≫ *Observer* is composed with the *Library* ≪theme≫ using the bind integration strategy. The template parameters of the crosscutting theme (i.e., classes, operations, and attributes) placed on the theme package template within a dotted box need to be bound to concrete modeling elements of a non-crosscutting theme. The sequence diagram templates in a crosscutting theme (cf. *ObserverPattern_StateChange* in Figure 3.13) allow modeling crosscutting behavior and when it shall be triggered, e.g., within the control flow of other operations. In contrast, the class diagram templates of a crosscutting theme allow modeling crosscutting structure. The concrete binding is specified by the ”bind” tag placed on the composition relationship between the crosscutting theme and other themes. It binds the template parameters to actual classes, operations, and attributes possibly using wildcards. This way, the *Subject* class is bound to *BookCopy*, and the *stateChange()* operation is bound to *borrow()* and *return()* (S.B). Since, *Theme/UML*’s composition relationships can relate composite elements such as package as well as fine-grained ones such as attributes, the approach supports both modeling at a high level of abstraction as well as at a low level (S.A).

MATURITY

The Theme/UML approach represents one of the most mature, and still evolving approaches to AOM (M.I). Lately, first results on the approach’s extension with the join point designation diagrams of Stein et al. [SHU06] has been presented [JC06] as well as an extension with the weaving algorithm of Klein et al. [KHJ06] has been published [JKBC06] (M.T). Theme/UML comes with a plethora of literature and modeling examples such as the synchronization and observer aspects in a digital library example [CW05], the logging aspect in an expression evaluation system example and a course management system example. The crystal game application presented in [CB05] consists of more than 15 concern modules amongst them two crosscutting ones. The composition of some of them is demonstrated. Furthermore, two similarly sized case studies are presented, i.e., phone features and usage licensing (M.E). It is not clear, however, if the approach has been applied in a real world project (M.A).

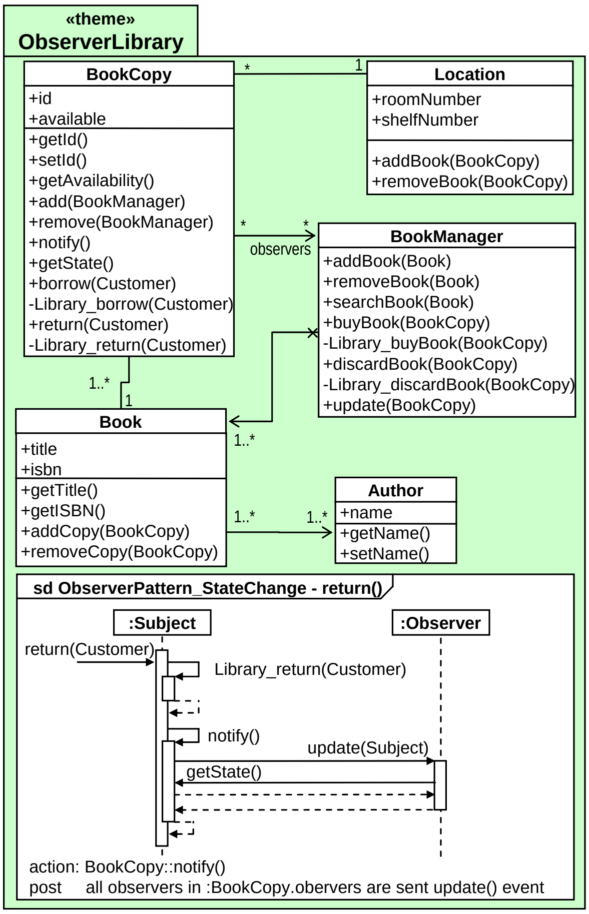


Figure 3.14: The Composed Model, Clarke et al.

TOOL SUPPORT

Besides first proposals in [JKBC06] with respect to composition, no information on a tool for *Theme/UML* supporting either modeling, composition or code generation has been provided (T.M), (T.C), (T.G).

3.3.8 Aspect-Oriented Architecture Models, France et al.

LANGUAGE

The *Aspect-Oriented Architecture Models (AAM)* approach of France et al. [FRGG04], [RGR+06] is based on UML 2.0 (L.L). The language is designed as a platform-independent approach with no particular platform in mind (L.I). Concerns are modeled using template diagrams, i.e., package diagram templates, class diagram templates and communication diagram templates [FRGG04] as well as recently sequence diagram templates [RSFG06], [SSR+05] (L.D). With respect to using UML templates, the approach is similar to *Theme/UML* (cf. Section 3.3.7). For readability purposes, however, the authors prefer to provide a notation different to standard UML templates and in contrast denote template model elements using '|'. This notation is based on the Role-Based

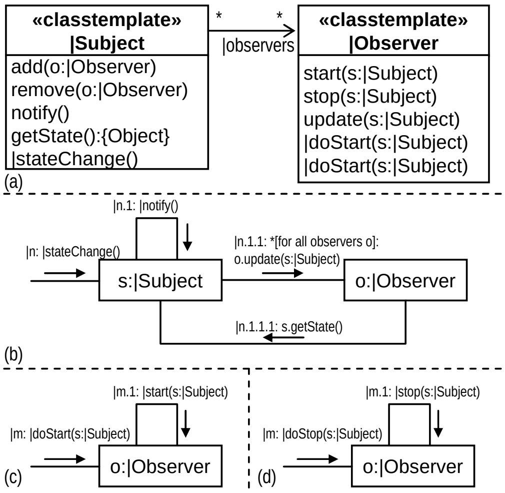


Figure 3.15: The Observer Aspect Model, France et al.

Metamodeling Language [FKGS04], [KFG04], which is a UML-based pattern language designed as an extension to the UML (L.E). The use of packages for capturing concerns caters for scalability, although this has not yet been demonstrated within an example encompassing several concerns (L.S). The approach is not specifically aligned to the requirements or implementation phases (L.A) and does not support external traceability (L.T). Nevertheless, similar to *Theme/UML*, the different models are continuously refined and at some point composed (L.T), (L.R). A design process is briefly outlined in terms of guidelines (L.DP).

ConcernComposition

The approach of France et al. originally is based on the compositor composition mechanism similar to the *Theme/UML* approach. Recently, specific attention has been paid, however, to the composition of sequence diagrams [RSFG06], [SSR^{+}05], which realizes the pointcut-advice composition mechanism (CC.M). France et al. support element symmetry in that all concerns are modeled as UML packages (CC.CM), (CC.ES). The authors distinguish, however, between "primary models" and "aspect models", which model crosscutting concerns. Aspect models are based on template diagrams, which are described by parameterized packages. These packages include class diagram templates as in Figure 3.15 (a), communication diagram templates as in Figure 3.15 (b)-(d), and recently sequence diagram templates (cf. Figure 3.18 (a)). A textual "binding" to a certain application instantiates a "context-specific" aspect model from the UML template. In the context of the library management system, the following binding instantiates the aspect model for the observer pattern from Figure 3.15 and results in the context-specific aspect shown in Figure 3.16:

```
(|Subject,BookCopy);           (|Observer, BookManager);
(|stateChange(),borrowCopy()); (|doStart(s:|Subject),buyBook());
(|stateChange(),returnCopy()); (|doStop(s:|Subject),discardBook());
(|observers, bookManagers);
```

The context-specific aspect models are finally used for composition with the base model, suggesting rule and composition symmetry (CC.RS), (CC.CS). However, the composition of sequence diagrams is somewhat different. The locations of where to introduce a behavioral advice defined within an aspect sequence diagram template (cf. Figure 3.18 (a)) are specified using "tags" in the primary model (cf. Figure 3.18 (b)). The aspect sequence diagram template can be composed with the primary model, only (CC.CS), and the rule information is placed within the primary model (CC.RS), meaning asymmetric composition and asymmetric placement of rules. Recently, the approaches composition semantics (CC.S) in terms of a composition metamodel have been operationalized in KerMeta, a metamodeling language that extends the Essential Meta-Object Facility (EMOF 2.0) with an action language. Thereby, the semantics of detection have also been operationalized (i.e., the getMatchingElements() operation), which allows for detecting (syntactical) conflicts (CC.CR). The composition is done statically yielding standard UML diagrams, i.e., class diagrams, communication diagrams and sequence diagrams (CC.CP). The composed model is shown in Figure 3.17 and Figure 3.18 (c), respectively (CC.CP) in terms of standard UML. Since KerMeta allows specifying operations with its action language, dynamic composition of models is subject to future work (CC.C). The approach also proposes so called "composition directives" which are intended to refine the concern composition rules used to compose models. The use of so called "model composition directives", allows specifying the order in which aspect models are composed with the primary model. These "precedes" and "follows" model composition directives are depicted as stereotyped UML dependencies between aspect models and represent a conflict resolution mechanism. Other forms of interactions between modules, however, cannot be modeled (CC.I). So called "element composition directives" amongst others allow to add, remove, and replace model elements. The element composition directives, consequently, also serve as a conflict resolution mechanism (CC.CR). The approach does not describe ways to specify effects (CC.E).

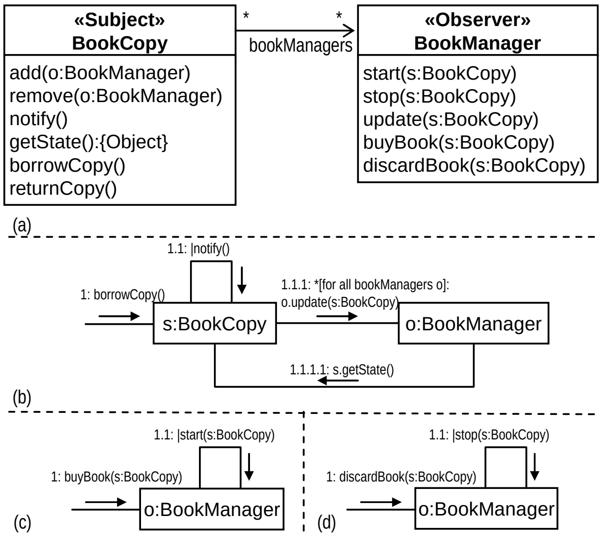


Figure 3.16: The Context-Specific Aspect Model, France et al.

SYMMETRICCONCERNCOMPOSITION

The composition of class diagrams is specified with the composition metamodel defined in [RGR+06]. Composable elements in the composition metamodel are realized with the meta-class Mergeable. Currently, the composition metamodel has been operationalized for class diagrams only. In this respect, mergeable elements are Operation, Association, Classifier, and Model (S.SCE), (S.BCE). Originally, the approach used only name-based matching method in order to identify the corresponding elements in different models. In [RGR$^+$06], this mechanism has been extended. The authors introduce a signature-based method, which means that elements are matched according to their syntactic properties, i.e., an attribute or an association end defined in the element's meta-class. This match method is realized with the getMatchingElements() operation of the composition metamodel (S.MM). The approach basically, supports a merge integration strategy, only (S.M). Support for the bind integration strategy is realized through the instantiation of aspect models to context-specific aspect models. The template parameters of the aspect models denoted using '|' need to be bound to concrete modeling elements of the primary model. For example, the class *|Subject* (cf. Figure 3.15 (a)) is bound to *BookCopy* and the operation *|stateChange()* is bound to *borrow()* and *return()* (cf. Figure 3.15 (b)-(d)). This is done with the textual binding as specified before and the resulting context-specific aspect model is shown in Figure 3.16 . While the class diagram templates model crosscutting structure, the communication diagram templates model crosscutting behavior. The context-specific aspect models then can be composed with the primary model using the original merge integration strategy (S.B). Lately, the possibility of overriding model elements has been introduced with the introduction of composition directives. The "override" element composition directive defines an override relationship between two potentially conflicting model elements [RGR$^+$06] (S.O). Consequently, with respect to symmetric concern composition the approach supports both modeling at a high level of abstraction (e.g., with a high-level model view and model composition directives) as well as at a low level (e.g., with detailed class and communication diagrams as well as element composition directives) (S.A)

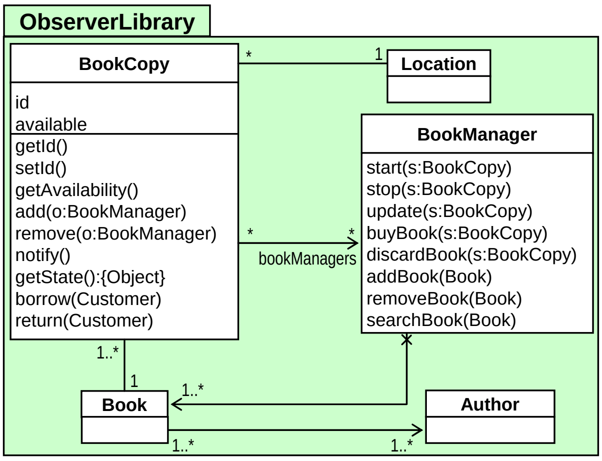


Figure 3.17: The Composed Model, France et al.

AsymmetricConcernComposition

The composition of sequence diagrams realizes the pointcut-advice composition mechanism.

AspectualSubject. The join point model is implicitly defined (AS.JPM) as the set of primary sequence model elements, e.g., lifelines and messages to which the aspect sequence model elements need to be composed, thus supporting solely behavioral-static join points. (AS.SJP), (AS.BJP) [RSFG06]. The locations of where to apply the advice in the primary sequence diagram are

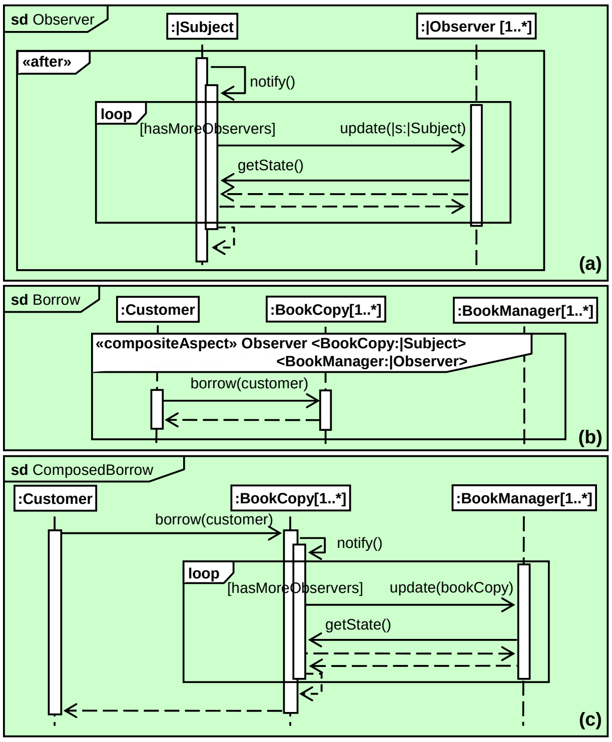


Figure 3.18: Weaving Aspectual Behavior With Sequence Diagrams, France et al.

"tagged" with special stereotypes (AS.P). Thereby, two stereotypes can be distinguished: A ≪simpleAspect≫ is a stereotyped UML message originating from and targeting the same lifeline, which is used when the aspectual behavior just needs to be inserted. A ≪compositeAspect≫ is a stereotype for UML's Combined fragment, that captures a message or a sequence of messages in the primary model as join point (AS.SP), (AS.QM). In the running example, the ≪compositeAspect≫ *Observer* is used to tag the primary sequence diagram *Borrow* (cf Figure 3.18 (b)). The ≪compositeAspect≫ also includes the binding of the *BookCopy* and *BookManager* classes to the corresponding template lifelines of the aspect sequence diagram template. This pointcut mechanism in terms of tagging a model does not allow for composed pointcuts (AS.CP). Concerning the aspectual subjects, thus, models provide information at a detailed level (AS.A). The relative position is modeled within the aspect sequence diagram template using stereotyped combined fragments. Besides the typical before, after, and around relative position kinds, the approach provides two special stereotypes, namely, ≪begin≫ and ≪end≫. The begin/end combined fragment captures the aspectual behavior that should preced/follow the messages encompassed in the ≪compositeAspect≫ of the primary model. In contrast, the ≪after≫ combined fragment shown in Figure 3.18 (a) defines the aspectual behavior that will appear after each message encompassed in the ≪compositeAspect≫ of the primary model.

AspectualKind. On the basis of sequence diagram templates, the approach of France et al. provides for behavioral advice, only (AK.BA), (AK.SA). The approach does not forsee possibilities of combining two or more behavioral advice to form a more complex one (AK.CA). The running example shows that a behavioral advice is modeled at a detailed level within the approach (AK.A).

MATURITY

The approach of France et al. is among the most mature AOM approaches and has been elaborated on in numerous publications (M.I) providing examples such as a simple banking system including the authorization aspect, the replicated repository aspect, the redundant controller aspect, and the transaction aspect for controlling money transfers, as well as the buffer aspect which decouples output producers from the writing device in a system (M.E) Currently, the approach is further developed with respect to its composition mechanism and tool support thereof (M.T). Yet, it has not been applied in a real-world project (M.A).

TOOL SUPPORT

Currently, a prototype implementation of an integrated toolset is under development. This toolset provides modeling support (T.M), i.e., for modeling aspect model diagram templates built on top of the Eclipse Modeling Framework[8] and for instantiating context-specific aspect models from these templates built on top of Rational Rose. Support for composition (T.C), i.e., for composing structural base and context-specific aspect models [RFG05] is built on top of KerMeta. A tool supporting composition of behavioral models, i.e., sequence diagrams is currently under development [RSFG06], while a tool for code generation currently is not planned (T.C).

3.4 Lessons Learned

The results of the evaluation have revealed interesting peculiarities of current AOM approaches. In the following, the findings are summarized and the results are illustrated at a glance with tables according to the six categories of criteria from the criteria catalogue within Sections 3.4.1 to 3.4.6. Finally, Section 3.4.7 presents the general findings and conclusions concerning the AOM research field and specifically points out what needs to be done in terms of further development of AOM approaches.

3.4.1 Language

The summary of the evaluation with respect to the *Language* category can be found in Table 3.1.

Popularity of UML Profiles and UML 2.0. For the design phase, one can observe that UML is the choice for designing an aspect-oriented design language with two exceptions [SR05], [SVWJ05], only. With respect to the UML version used, there is quite a balance between UML 1.x and UML 2.0. Typically, recent approaches already are based on the new UML 2.0 specification. Furthermore, it is advisable that existing UML 1.x approaches are updated to also support UML 2.0. For extending UML with aspect-oriented concepts, UML's inherent extension mechanism, i.e., profiles, is popular. In terms of tool support this is beneficial, since UML profiles are supported by almost any UML tool. Nevertheless, it has to be noted, that the profile-based approaches of Stein et al. and Aldawud et al. do not provide a reference implementation of their profiles within a UML tool. Consequently, designers first are required to manually redefine the profile specification in their UML tool of choice. With respect to those approaches that are based on metamodel extensions, i.e., Clark et al., Jacobson et al., and France et al., almost no modeling support is available. Only France et al. are currently developing an integrated toolset providing modeling support upon the EMF.

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

	Modeling Language (L.L)		Extension Mechanism (L.E)		Platform Influences(L.I)	Diagrams (L.D)		Design Process (L.DP)		Scalability (L.S)		Refinement Mapping (L.R)	Alignment to Phase (L.A)	Traceability (L.T)	
	UML 1.x	UML 2.0	Metamodel	UML Profile		Structural Diagrams (L.D)	Structural Diagrams (L.D)	process description	guidelines	high-level modeling elements	proven with examples			internal	external
Stein	✓	🟦	✓	✓	AspectJ	CD, ColD	SD, UCD	🟦	🟦	🟦	🟦	🟦	I	n/a	D->I
Pawlak	✓	🟦	✓	✓	JAC	CD	🟦	🟦	🟦	🟦	🟦	🟦	I	n/a	D->I
Jacobson	🟦	✓	✓	🟦	AspectJ, Hyper/J	CD, CompD	UCD, SD, ComD	✓	🟦	✓	✓	c,e	R,I	✓	R->A->D->I
Klein	🟦	✓	🟦	🟦	🟦	🟦	SD	🟦	🟦	🟦	🟦	🟦	n/a	n/a	🟦
Cottenier	🟦	✓	🟦	✓	🟦	CD, CSD, DD	SD, SMD	🟦	🟦	✓	✓	c,e	I	✓	🟦
Aldawud	✓	🟦	🟦	✓	🟦	CD	SMD	🟦	✓	🟦	🟦	e	R	n/a	R->D
Clarke	✓	🟦	✓	🟦	SOP, AspectJ, Hyper/J	PD, CD	SD	✓	🟦	✓	✓	c,e	R,I	✓	R->D->I
France	🟦	✓	✓	🟦	🟦	PD, CD	ComD, SD	🟦	✓	✓	🟦	c,e	n/a	✓	🟦

Legend:

✓	supported	CD	Class Diagram	SD	Sequence Diagram	R	Requirements
🟦	not supported	CompD	Component Diagram	UCD	Use Case Diagram	A	Analysis
n/a	not applicable	CSD	Composite Structure Diagram	ComD	Communication Diagram	D	Design
c	creating	PD	Package Diagram	SMD	State Machine Diagram	I	Implementation
e	extending	ColD	Collaboration Diagram (UML 1.x)	DD	Deployment Diagram		

Table 3.1: Language

Behavioral Diagrams are Catching up. Except for the approach of Klein et al. all approaches make extensive use of structural diagrams, i.e., class diagrams and package diagrams as well as component diagrams and composite structure diagrams. The use of behavioral diagrams in order to describe crosscutting behavior is more and more emphasized by also composing behavioral diagrams such as in the approaches of Klein et al., Cottenier et al., and France et al.

Missing Guidance in the Design Process. Since AOM is still a quite young research field, the support of design processes has often been disregarded and thus is rather underdeveloped. Only two surveyed approaches, i.e., Clarke et al. and Jacobson et al., provide a detailed design process description. Additionally, two approaches have some guidance in designing aspect-oriented models in terms of guidelines, namely France et al. and Aldawud et al. It is not surprising, that the two approaches offering a design process have been chosen by the AOSD Europe Network of Excellence to form the basis for defining a generic aspect-oriented design process [CJ06].

Missing Full External Traceability. The majority of approaches does not support external traceability at all or with respect to either prior or later development phases, only. The approaches of Jacobson et al. and Clarke et al. are the only ones that do support external traceability from the requirements engineering phase until implementation. Since traceability is a crucial issue in any kind of software development, AOM approaches need to take care to support for traceability.

Moderate Scalability. Half of the approaches provides for scalability by supporting modeling concepts that allow hiding details from the modeler and thus, modeling at a high level of abstraction. The modeling examples used, however, are seldom of a size that justifies scalability. Only, the approaches of Jacobson et al., Cottenier et al., and Clarke et al. have provided proof that their approaches can cope with the composition of three or more concerns. Due to this limited interest in proving the applicability of the AOM approaches in large-scale applications, it will be required in the future to evaluate how scalable those approaches are in real large-scale applications by means of quantitative studies.

3.4.2 ConcernComposition

For the *ConcernComposition* category, the lessons learned are drawn from Table 3.2.

Popularity of Asymmetric Concern Composition. The majority of AOM approaches follows the pointcut-advice mechanism or a combination of the pointcut-advice and open class mechanisms. The approach of France et al. seems to be the first that combines the compositor mechanism and the pointcut-advice mechanism. It is however interesting to note that up to now little interest has been shown in evaluating when asymmetric and symmetric approaches have prevailing advantages and shall be employed.

Influence of Composition Mechanism on Element, Composition, and Rule Symmetry. Generally, one can observe that asymmetric composition mechanisms usually imply element asymmetry, composition asymmetry as well as rule asymmetry, although this is not an inherent characteristic of asymmetric approaches. On the opposite, a symmetric composition continuously achieves symmetric values. The approach of Klein et al. is one exception to the rule supporting element and composition symmetry, since the modeling concepts used for modeling the base behavior, the crosscutting behavior as well as the pointcuts is done using sequence diagrams, which in different contexts can play different roles. Other examples are the approaches of Jacobson et al. and Aldawud et al. which at a higher level do support element symmetry (≪use case slice≫) and element asymmetry (≪aspect≫), respectively. At a lower level, however, one can observe the reverse: Jacobson et al. model and compose ≪aspect≫ classes with normal classes while Aldawud et al. uses state machine regions to model (non-)crosscutting concerns. The approach of France et al. that combines the compositor mechanism and the pointcut-advice mechanism results in rule symmetry for the compositor mechanism and rule asymmetry for the pointcut-advice mechanism.

Composition often Deferred to Implementation. Composition at modeling level is only supported by half of the surveyed approaches. The composition always yields a composed model conforming to standard UML except for the approach of Clarke et al., where the outcome is represented by a composite ≪theme≫. The composition semantics of Cottenier et al. and France et al. have already been implemented within (prototype) tools, while Klein et al. have defined a weaving algorithm. In contrast to Clarke et al., this operationalization enables dynamic composition at modeling level. Well-defined semantics already at the modeling level is a necessary prerequisite for achieving more than models-as-blue-print. If, as intended in MDE, models shall replace code appropriate semantics along with composed models to assist the designer will be required.

Moderate Support for Modeling Interactions. Modeling of interactions both at the level of modules and the level of rules is still considered rather moderately. Typically the means for specifying interactions at the same time are modeling concepts for resolving conflicts, e.g., an ordering for composing concern modules or concern composition rules. The approach of Cottenier et al. represents an exception by proposing ≪hidden_by≫ and ≪dependent_on≫ dependencies between aspects. Since it is natural to expect that large-scale systems might put forward interaction of modules, for an unambiguous specification of the system it will be necessary to make module interaction explicit.

	ConcernModule (CC.CM)	Composition Mechanism (CC.M)	Element Symmetry (CC.ES)	Composition Symmetry (CC.CS)	Rule Symmetry (CC.RS)	Effect (CC.E)	Composition Semantics (CC.S)		Composition (CC.C)		Composed Module (CC.CP)	Interaction (CC.I)		Conflict Resolution (CC.CR)		
							detection	composition	static	dynamic		Module	Rule	avoid	detect	resolve
Stein	<<aspect>> Class	PA, OC	🟦	🟦	🟦	🟦	✓	✓	✓	🟦	n/a	✓	🟦	🟦	🟦	✓
Pawlak	<<aspect>> Class	PA	🟦	✓	🟦	✓	🟦	🟦	🟦	🟦	n/a	🟦	🟦	🟦	🟦	🟦
Jacobson	<<use case slice>> Package; <<aspect>> Classifier	PA, OC	~	🟦	🟦	🟦	🟦	🟦	🟦	🟦	n/a	✓	🟦	✓	🟦	🟦
Klein	Pair of basic SDs representing Pointcut & Advice	PA	✓	✓	🟦	🟦	✓	✓	✓	~	Stand. UML	🟦	🟦	🟦	🟦	🟦
Cottenier	<<Aspect>> Class	PA	🟦	🟦	🟦	🟦	✓	✓	✓	✓	Stand. UML	✓	✓	🟦	🟦	✓
Aldawud	<<aspect>> Class; State machine regions	CMP	~	🟦	🟦	🟦	✓	✓	🟦	🟦	n/a	✓	🟦	🟦	🟦	✓
Clarke	<<theme>> Package (Template)	CMP	✓	✓	✓	🟦	✓	✓	✓	🟦	Themes	✓	✓	🟦	🟦	✓
France	Package (Template)	CMP, PA	✓	✓	~	🟦	✓	✓	✓	~	Stand. UML	✓	✓	🟦	✓	✓

Legend:

✓	supported
🟦	not supported
~	partly supported
n/a	not applicable

PA	Pointcut-Advice
OC	Open Class
CMP	Compositor

Table 3.2: ConcernComposition

Conflict Resolution Based on an Ordering for Composition, Only. A conflict resolution is provided by half of the approaches focusing on resolving conflicts. The conflict resolution mechanisms in most cases comprise means for specifying an ordering of how concern modules are composed, some provide further means, e.g., to resolve naming conflicts. Only one approach's composition semantics (France et al.) allows to detect (syntactical) conflicts. The approach of Jacobson et al. is the only one that explicitly avoids conflicts by continuously refactoring models. Consequently, the provision of more sophisticated conflict resolution mechanisms including the detection of conflicts should be focussed in future.

Effect Not Considered. Modeling the effect of composition is not considered at all. Only the JAC design notation of Pawlak et al. provides a stereotype ≪replace≫ for advice which indicates an effect of either a replacement or a deletion. The possibility of modeling the effect, however, would enhance the explicitness of models and thus allow for providing better conflict identification.

3.4.3 AsymmetricConcernComposition

This part of the lessons learned specifically summarizes the results for the approaches adhering to the pointcut-advice and/or open class composition mechanism, i.e., Stein et al., Pawlak et al., Jacobson et al., Klein et al., Cottenier et al., and France et al.

3.4.3.1 AspectualSubject

The results of evaluating the approaches according to the criteria encompassed by the *Aspectual-Subject* sub-category are shown in Table 3.3.

Missing Formal Definition of Join Point Models. Half of the surveyed approaches made the Join Point Model not explicit but defined it ”implicitly” via their pointcut mechanism. The remaining did provide a Join Point Model but mostly in terms of a natural language description, only. Consequently, formal definitions of join point models are missing and shall be considered in future development of AOM approaches.

	Structural Join Point (AS.SJP)		Behavioral Join Point (AS.BJP)		Explicit JoinPointModel (AS.JPM)	Standardized Pointcut (AS.P)	SimplePointcut (AS.SP)		CompositePointcut (AS.CP)		Quantification Method (AS.QM)			Relative Position (AS.RP)			Abstraction (AS.A)	
	static	dynamic	static	dynamic			graphical	textual	graphical	textual	declarative	imperative	enumeration	before	around	after	high	low
Stein	✓	🟦	✓	✓	✓	✓	✓	✓	🟦	✓	✓	🟦	✓	✓	✓	✓	✓	✓
Pawlak	🟦	✓	✓	🟦	✓	🟦	🟦	✓	🟦	✓	✓	🟦	✓	✓	✓	✓	🟦	✓
Jacobson	✓	🟦	✓	✓	🟦	✓	🟦	✓	🟦	✓	✓	🟦	✓	✓	✓	✓	✓	✓
Klein	🟦	🟦	✓	🟦	✓	✓	✓	🟦	~	🟦	✓	🟦	✓	🟦	✓	🟦	🟦	✓
Cottenier	🟦	✓	✓	🟦	✓	✓	✓	🟦	🟦	✓	✓	🟦	✓	🟦	✓	🟦	✓	✓
France	🟦	🟦	✓	🟦	🟦	✓	✓	🟦	🟦	🟦	✓	🟦	✓	✓	✓	✓	🟦	✓

Legend:

✓	supported
🟦	not supported
~	partly supported

Table 3.3: AspectualSubject

Limited Support for Structural Join Points. Supporting the full spectrum of join points in AOM approaches would be beneficial with respect to possible enhancement, replacements, and deletions made in the form of advice. The approaches realizing the open class composition mechanism support structural-static join points. The approaches of Pawlak et al. and Cottenier et al. allow for structural advice which are introduced at run-time, however, thus requiring structural-dynamic join points (e.g., object instances). The approaches of Klein et al. and France et al. do not provide structural join points at all. Nevertheless, it has to be noted, that the approach of France et al. compensates the lack of at least structural-static join points due to also supporting the compositor composition mechanism. All approaches consider either behavioral-static or behavioral-dynamic join points or both.

Standards-Based Pointcut Mechanism Preferred. All but one of the approaches provide a standards-based pointcut mechanism. Pawlak et al. proposes the only proprietary pointcut language based on regular expressions and/or keywords as well as on UML associations. The approaches of Jacobson et al. and Stein et al. reuse AspectJ’s pointcut language. The rest relies on UML behavioral diagrams to specify pointcuts, e.g., sequence diagrams (Klein et al.) and combined fragments in sequence diagrams (France et al.), as well as state machines (Cottenier et al.)

Good Support of Composite Pointcuts. The reuse of simple pointcuts is fostered by the use of composite pointcuts, for which good support is provided in almost all the approaches. The textual pointcut mechanism of Stein et al., Pawlak et al., and Jacobson et al. provide textual mechanisms that allow for composing simple pointcuts using logical operators. Cottenier et al. allow composing pointcuts with special logical composition operators. While France et al. provides no support for modeling composite pointcuts, the approach of Klein et al. in principle could support composition of sequence diagrams on the basis of the sequential composition operator.

No Imperative Pointcuts. The approaches exclusively allow to select join points declaratively and/or by enumeration but not in the form of imperative pointcuts, which could serve as a more verbose pointcut definition.

Good Support of Relative Position Kinds. Except for the approaches of Klein et al. and Cottenier et al., which only cater for the *around* relative position kind and therefore subsume the *before* and *after* kinds, the other approaches provide full support of relative position kinds. Interestingly, France et al. support two uncommon positions, namely "begin" and "end". Furthermore, it might be interesting to discuss how the relative positions shall be interpreted in the light of model elements, since finally the composition in any case will do concrete insertions and deletions of metamodel instances.

Modeling Aspectual Subjects at Different Levels of Abstraction. All approaches allow to apply advices to the base at a low level of abstraction, but half of the approaches also allows modeling the subjects of adaptation at a higher level of abstraction. For the applicability of AOM, a high level of abstraction is beneficial, whereas for code generation purposes as well as an automated execution of the model a detailed specification at a low level of abstraction is necessary.

3.4.3.2 AspectualKind

In Table 3.4, the results for the *AspectualKind* sub-category are provided.

Composite Advice Not Considered. While most approaches provide modeling means for both behavioral and structural advice, composing advice to form more complex ones and to foster reuse is not considered by any of the approaches. Nevertheless, the approach of Klein et al. in principle could support composition of sequence diagrams, i.e., behavioral advice, on the basis of the sequential composition operator.

	Behavioral Advice (AK.BA)	Structural Advice (AK.SB)	Composite Advice (AK.CA)	Abstraction (AK.A)	
				high	low
Stein	✓	✓	🟦	🟦	✓
Pawlak	✓	✓	🟦	🟦	✓
Jacobson	✓	✓	🟦	🟦	✓
Klein	✓	🟦	~	🟦	✓
Cottenier	✓	✓	🟦	✓	✓
France	✓	🟦	🟦	🟦	✓

Legend:

✓	supported
🟦	not supported
~	partly supported

Table 3.4: AspectualKind

Modeling Aspectual Kinds at a Low Level of Abstraction. Again, as for abstraction with respect to the aspectual subjects all approaches allow modeling advice at a low level of abstraction. The approach of Cottenier et al. is the only one supporting modeling also at a high level of abstraction. It would be beneficial, however, if approaches would provide for high as well as low level of abstraction.

3.4.4 SymmetricConcernComposition

This part of the lessons learned specifically summarizes the results for those approaches supporting the compositor composition mechanism, i.e., Aldawud et al., Theme et al., and France et al (cf. Table 3.5).

Equal Support of Structural and Behavioral Composable Elements. While the approach of Aldawud et al. is based on state machines, the approach of France et al. allows composing class diagrams. Thus, the supported composable elements are for the first case behavioral and in the second case structural. Ideally, composition is possible for both kinds, such as in the composition metamodel of Clarke et al. Nevertheless, for the approach of France et al., it has to be noted that the lacking support for behavioral composable elements can be seen as compensated due to supporting the pointcut-advice composition mechanism for composing sequence diagrams.

	Comp. Elements		Match Method (S.MM)			Integration Strategy			Abstraction (S.A)	
	Structural Composeable Elements (S.SCE)	Behavioral Composeable Elements (S.BCE)	match-by-name	match-by-signature	no-match	Merge (S.M)	Override (S.O)	Bind (S.B)	high	low
Aldawud	🟦	✓	✓	🟦	🟦	✓	🟦	🟦	🟦	✓
Clarke	✓	✓	✓	🟦	✓	✓	✓	✓	✓	✓
France	✓	🟦	✓	✓	🟦	✓	✓	✓	✓	✓

Legend:

✓	supported
🟦	not supported

Table 3.5: SymmetricConcernComposition

Predominance of Matching with Names. Finding corresponding elements on the basis of a name-based match method represents an easy-to-implement method and in many cases quite an effective way, which is consequently supported by all three approaches. Clarke et al. additionally allow to explicitly model which elements shall not match by supporting a *no-match* method. Recently, France et al. have proposed a more expressive match method based on the elements signatures. The advantage of such a matching method lies in the possibility of finding corresponding elements with more fine-grained matching criteria other than the element name, e.g., the values of meta-class properties such as the "isAbstract" meta-attribute of classes, as well as in the possibility of detecting and resolving conflicts.

Merge Integration as a Default Strategy. With respect to supporting different integration strategies, merge is supported by all surveyed approaches. The override and bind strategies are also supported by all approaches but Aldawud et al. In terms of expressivity, ideally all integration strategies are supported by an approach.

Modeling at Different Abstraction Levels. The approaches generally provide good support for modeling at a high and a low level of abstraction. In particular, the approaches of Clarke et al. and France et al. offer high level views on the concern modules to be composed in terms of UML packages, while Aldawud et al. model at a detailed level by means of state machines. A high level of abstraction is beneficial in terms of an approach's scalability, whereas for code generation purposes as well as an automated execution of the model a low level view such as the one of

Aldawud et al. is required. In this respect, state machines probably represent the most elaborate mechanism for describing an objects life cycle, and are supported by code generation tools such as Raphsody and StateMate[9].

3.4.5 Maturity

In Table 3.6, the measures for the *Maturity* category are depicted for all surveyed approaches.

	Modeling Examples (M.E)		Applications (M.A)	Publications (M.I)	Topicality (M.T)
	Concerns	Examples			
Stein	2	2	🟦	4	`02
Pawlak	1	1	✓	3	`05
Jacobson	>3	1	🟦	3	`05
Klein	1	1	🟦	9	`06
Cottenier	>4	2	✓	9	`06
Aldawud	>2	1	🟦	5	`05
Clarke	>15	>5	🟦	>15	`06
France	1	3	🟦	>10	`06

Legend:

✓	supported
🟦	not supported
n	number of concerns, examples, publications
`YY	year of most recent publication

Table 3.6: Maturity

Missing Complex Examples. The majority of the approaches, have demonstrated their approaches on the basis of rather trivial examples in which not more than two concerns are composed. In this respect, Jacobson et al., Cottenier et al., and Clarke et al. set a good example by demonstrating their approaches with non-trivial modeling problems. It would therefore be beneficial if all AOM approaches would document their capabilities on bases of more complex examples.

Lack of Application in Real-World Projects. The applicability of AOM languages has rarely been tested in real-world projects. An exception is the approach of Pawlak et al., which has already been applied to real industrial projects like an online courses intranet site, an incident reporting web site, and a business management intranet tool. Another exception is the approach of Cottenier et al., since their Motorola Weavr tool is already being deployed in production at Motorola.

3.4.6 Tool Support

Finally, the results concerning the approaches' *Tool Support* are summarized in Table 3.7.

Missing Tool Support for Composition and Code Generation. While modeling support in many approaches is implicitly available due to the use of UML's profile mechanism, support for code generation and composition is rare. The approach of Cottenier et al. is the only one that allows for modeling, composition, and code generation. On the one hand, for those approaches that defer composition to the implementation phase, code generation facilities that produce code for the target AOP platform would be beneficial. For example, the approach of Pawlak et al. allows

[9] http://www.ilogix.com/

for code generation for the JAC Framework. On the other hand, composition tool support is essential for those approaches, that have specified the composition semantics for model composition. The acceptance of an AOM approach will be minimal, if it requires the designers to first model each concern separately and then to manually compose them. In this respect, the approaches of France et al. and Cottenier et al. provide appropriate tool support, while the "weaving algorithm" of Klein et al. is currently being implemented.

	Modeling Support (T.M)	Composition Support (T.C)	Code Generation (T.G)
Stein	✓	🟦	🟦
Pawlak	✓	🟦	✓
Jacobson	🟦	🟦	🟦
Klein	✓	~	🟦
Cottenier	✓	✓	✓
Aldawud	✓	🟦	🟦
Clarke	🟦	🟦	🟦
France	✓	✓	🟦

Legend:

✓	supported
🟦	not supported
~	partly supported

Table 3.7: Tool Support

3.4.7 General Findings

This section sums up the most important conclusions that are valid for the academic community as well as for practitioners.

No Explicit Winner. From the results obtained in the evaluation, it is not possible to nominate a winner. The selection of an AOM approach thus has to be made in the context of a specific project. For instance, if the requirement is enabling documentation and communication between partners of an AspectJ-based project, a design notation for AspectJ programs is needed and the approach of Stein et al. would be a good solution. On the other hand, one might wish for separating concerns in different class diagrams at design time and then before implementation compose the different views. In this respect, both approaches of Clark et al. as well as France et al. would be possible options. Depending on when the current prototype implementation of their integrated toolset is made available, the approach of France et al. might even be preferred for its tool support. Staying with tool support, the approaches of Pawlak et al. and Cottenier et al. might be of interest. In contrast to the approach of Stein et al., the *JAC Design Notation* of Pawlak et al. has been specifically designed for the JAC Framework but comes with modeling support as well as a code generation facility. What might argue in favor of the *AODM* of Stein et al. is AspectJ's maturity. The unique selling point of the Motorola Weavr of Cottenier et al. is its comprehensive tool support and in particular its composition mechanism for state machines, for which an academic license can be obtained. Nevertheless, the approach does not allow composing structural diagrams.

Full Spectrum of UML not Exploited. Interestingly enough, apart of the approaches of Palawak et al. and Klein et al. the surveyed approaches support structural as well as behavioral diagrams. Thus, in principle, the approaches allow the modeler to consider both structure and behavior through their approach. Nevertheless, currently no approach addresses the full spectrum of UML in terms of UML's structural and behavioral diagrams as well as their composition. It is comforting that the presumably most often employed UML diagrams have been addressed by AOM approaches. As can be seen in Table 3.1, the most important structural diagram, i.e., class diagram, is supported by all approaches addressing structural modeling some of them also allowing their composition. Likewise for modeling behavior the sequence diagram is covered by all approaches addressing behavior, some of them also supporting their composition. It would therefore be interesting to investigate how to compose diagram types for which composition has not yet been specified, e.g., composite structure diagrams. As a consequence, it would also be interesting how the approaches can be combined in order to gain from best practises in AOM. In this respect, a first promising attempt has recently been conducted by Clarke et al. and Klein et al. by proposing *KerTheme* [JKBC06], a combination of their approaches. Furthermore, the *Theme/UML* approach of Clarke et al. has also been extended with join point designation [JC06] diagrams of Stein et al.

Missing Guidance on When to use Asymmetric vs. Symmetric Approaches. It might be a natural pre-assumption that approaches either follow the asymmetric school of thought or the symmetric school of thought. France et al. is interesting in this respect since it provides for both. The recent extension made to *Theme/UML* in the *KerTheme* proposal also follow this direction of combining different composition mechanisms. In terms of expressivity, the advantages of using different composition mechanisms are obvious. Nevertheless, the question when to best apply an asymmetric or a symmetric approach has not yet been answered sufficiently.

Missing Tool Support. Certainly, one of the most vital factors for the adoption of AOM in practise but also in academia is the provision of appropriate tools. Basic modeling support is provided for some approaches, i.e., for those approaches which rely on UML profiles and consequently can rely on existing UML modeling tools. AOM, however, is also about the composition of various concerns that have been carefully separated beforehand. This is a complex task to be understood by the modeler hence, support for model composition is vital. Still, this is not commonly provided by the AOM approaches and is also hampered by the fact that not all AOM approaches provide for a well-defined composition semantics. Finally, code generation, which is an important requirement for MDE, is least supported by tools. Only the approaches of Pawlak et al. and Cottenier et al. provide such facilities.

Adoption of Approaches Requires Scalability. For the adoption of AOM, it would be beneficial if its applicability would be better evaluated with respect to large scale applications and real-world scenarios. This is currently only sufficiently addressed by very few approaches, namely, Clarke et al., Jacobson et al., and Cottenier et al. Nevertheless, scalability is a feature important to practitioners and has a great impact on the chances of AOM approaches to be adopted.

3.5 Related Surveys

In an effort to shed light on different approaches to aspect-orientation, some surveys comparing aspect-oriented approaches at different levels in the software development life cycle have already been presented. In the following, such related surveys can be distinguished into ”closely related” surveys particularly emphasizing on AOM (cf. Section 3.5.1) and more ”widely related” ones focusing on AOP (cf. Section 3.5.2), which are nevertheless of interest in the context of this survey. Furthermore, there exists some work aiming at ”unifying” the currently prevailing diversity of concepts in the aspect-orientation paradigm. The influence of those on this survey in terms of the CRM is discussed in detail in the Section 3.1.

3.5.1 Aspect-Oriented Modeling Surveys

With respect to closely-related surveys on AOM approaches, the most extensive work is provided by Chitchyan et al. [CRS^{+}05]. This survey's goal is to "elicit initial insights into the roadmap for developing integrated aspect-oriented requirements engineering, architecture, and design approaches". Therefore, for each phase of the software development process a review of prominent contemporary AOSD as well as non-AOSD approaches is provided. For the design phase, the survey presents the evaluation results of 22 aspect-oriented design approaches along with UML as the only non-AOSD approach on the basis of a set of 6 evaluation criteria.

Similar, but less extensive AOM surveys with respect to both the set of criteria and the amount of surveyed approaches have been provided by Reina et al. [RTT04], Blair et al. [BBR^{+}05], and Op de beeck et al. [dbTB^{+}06]. Reina et al. have evaluated 13 approaches with respect to a set of 4 criteria, only. More specifically, the goal of Reina et al. has been to investigate each approach with respect to its dependency on particular platforms as well as its dependency on specific concerns, i.e., if the approach is general-purpose or not.

The major goal in Op de beeck et al. is to investigate 13 existing AOM approaches within the realm of product-line engineering of large-scale systems and to position them within the full life cycle of a software development process. In this respect, the authors have evaluated a subset of approaches already presented by Chitchyan et al., as well as refined a set of six criteria, which partly have been presented in Chitchyan et al. In addition, the authors provide a discussion of the criteria's impact on certain software quality factors.

Blair et al. provide separate sets of criteria for the phases of aspect-oriented requirements engineering, specification, and design. Concerning the design phase, the authors evaluate 5 approaches according to a set of 8 criteria.

With respect to these existing surveys the present one differs in several ways:

Evaluation Granularity. One major difference between this survey and others concerns the breadth and depth of the evaluation. In particular, the survey investigating most approaches, i.e., the survey of Chitchyan et al., aims at providing a broad overview by including all existing aspect-oriented design approaches as well as not so well-elaborated proposals for such design approaches. In contrast, this survey tries to provide an in-depth investigation of selected approaches that have already gained a certain level of maturity in terms of publications at acknowledged conferences and/or journals as well as are based on the Unified Modeling Language (UML) [OMG05d] as the prevailing standard in object-oriented modeling. For an in-depth evaluation a catalogue of criteria is provided which encompasses more than 40 criteria. In contrast, the other sets of criteria [BBR^{+}05], [CRS^{+}05], [dbTB^{+}06], and [RTT04], do not include more than 8 criteria. In literature, 14 mature, UML-based AOM approaches have been identified, including the approaches already investigated in related surveys. In this survey, a representative set of 8 UML-based aspect-oriented design approaches is evaluated, which has been carefully selected from the above mentioned 14 approaches with respect to maintaining the ratio between the extension mechanisms used (metamodel vs. profile) as well as the ratio between symmetric and asymmetric approaches (cf. Section 3.3).

Methodology. Another important difference of this survey to the aforementioned ones lies in the applied methodology, which bases on a carefully established catalogue of criteria. Great emphasis has been put on the selection of criteria and their definition in a top-down as well as a bottom-up manner. A so-called *Conceptual Reference Model* for AOM (cf. Section 3.1) has been proposed, which identifies the basic AOM concepts as well as their interrelationships and thus, forms the basis for deriving the set of criteria in a top-down manner. Furthermore, for all criteria used, a clear definition along with the specification of the measurement scale is given. At the same time, this survey aims at complementing the set of criteria in a bottom-up manner by those criteria used in related AOM surveys [BBR$^+$05], [CRS$^+$05], [dbTB$^+$06], [RTT04]. More specifically, criteria found in other surveys have been adopted where appropriate or they have been refined where necessary, e.g., with respect to their definition or in terms of a decomposition into sub-criteria. In the catalogue of criteria (cf. Section 3.2), it is indicated which criteria have been adopted and which have been refined. Nevertheless, 6 criteria proposed in related surveys have been explicitly excluded from the evaluation framework due to methodological issues. Specifically, these criteria encompass reusability, comprehensibility, flexibility, ease of learning/use, parallel development, as well as change propagation [BBR$^+$05], which corresponds to the evolvability criterion [CRS$^+$05] and cannot reasonably be measured without empirical studies, e.g., user studies and extensive case studies. Thus, the catalogue of criteria subsumes the criteria derived from the CRM and the criteria provided by other surveys.

Inclusion of Recent Approaches. Furthermore, this survey also considers recently published approaches, namely [CvdBE07], [KHJ06], not included in the other surveys. In this way, this survey is complementary to the aforementioned surveys by considering also very recent developments.

Running Example. Finally, in contrast to all other surveys, the evaluation is supported by a running example that is realized with each of the surveyed approaches. This further supports the evaluation in that it first, illustrates each approach and second, allows to better compare the modeling means of the approaches and understand their strengths and shortcomings. If at all, other surveys rely on diverse examples sometimes taken directly from the respective approach's publications (cf. [dbTB$^+$06]).

3.5.2 Aspect-Oriented Programming Surveys

Less closely related, since focusing on AOP, is the survey of Hanenberg [Han05] which presents a set of criteria used to evaluate four AOP languages. Kersten [Ker05] also provides a comparison of four leading AOP languages having only AspectJ in common with Hanenberg. In addition, Kersten also investigates the development environments of these AOP languages.

Although focused on AOP, the evaluation criteria defined in those surveys are also of interest, since some of the surveyed AOM approaches are aligned to a certain aspect-oriented programming language. Nevertheless, some of them are not applicable in the context of this survey, since they are specifically related programming level issues, only. For example, Hanenberg distinguishes between ”code instrumentation” and ”interpretation weaving techniques” in the context of AOP weavers. In this survey, some of their criteria have been adopted and refined such that they can be applied at the modeling level, too. For example, the idea of evaluating tool support for AOM approaches has been inspired by Kersten's criteria on IDE support (e.g., editor, debugger).

3.6 Summary

This chapter presents an in-depth evaluation of eight aspect-oriented modeling approaches. Since the research field of aspect-oriented modeling is quite young and a common understanding of concepts has not yet been established, prior to evaluating the set of aspect-oriented modeling approaches, the important concepts of aspect-oriented modeling have been identified the form of a *Conceptual Reference Model* for aspect-oriented modeling, i.e., a UML class diagram representing aspect-oriented concepts and their interrelationships.

In a domain, where a generally acknowledged terminology is missing, this Conceptual Reference Model is an intermediate but essential step towards defining an evaluation framework, i.e., it forms the basis for inferring a set of concrete criteria. In the context of this thesis, the Conceptual Reference Model furthermore represents an important step towards extending an existing web modeling language with aspect-oriented modeling concepts (cf. Chapter 5).

The actual evaluation according to the presented catalogue of criteria is supported by a running example, which has proven to be helpful, on the one hand, to explore the applicability of each individual approach and, on the other hand, to allow for a direct comparison of the approaches. The evaluation results reveal that currently, there is no decidedly superior aspect-oriented modeling approach but that each individual approach has it specific strengths and shortcomings. For applying aspect-oriented modeling in a project, this means selecting an aspect-oriented modeling approach by matching the projects requirements with the approach's features. In this respect, the evaluation's results and lessons learned represent the basis for a well-founded decision.

4 Bridging WebML to Model-driven Engineering

Contents

This chapter presents the fundamental prerequisites for extending the WebML language with concepts from the aspect-orientation paradigm (cf. Chapter 5). The WebML language has been partly specified in terms of XML Document Type Definitions (DTD) and partly hard-coded within the tool accompanying the language. Consequently, in order to support model-driven development of web applications in the sense of Model-Driven Engineering (MDE), the WebML language needs to be specified in a MDE-suitable way, e.g., in terms of a metamodel. Considering the language's size, however, we refrain from manually re-modeling WebML from scratch, since this would be a cumbersome and error-prone process. Instead, the existing DTD-based language specification shall be reused in a semi-automatic process for metamodel generation from DTDs. In this respect, constraints hard-coded within the language's modeling tool WebRatio shall be extracted as well. After an introduction to this chapter's particular motivation in Section 4.1, Section 4.2 is dedicated to the explanation of the concepts of DTDs and metamodels as well as certain deficiencies of DTDs when used as a mechanism for defining modeling languages. Section 4.3 then describes the transformation process, including a set of transformation rules, and a set of heuristics giving indication for a manual refactoring, as well as a presentation of the implementation of the semi-automatic transformation approach in the form of the so-called *MetaModelGenerator* (MMG). In Section 4.4, the transformation framework is applied to the WebML DTD and the resulting WebML metamodel is presented. A discussion of the metamodel's completeness with respect to the WebML DTD and the metamodel's quality is given in Section 4.5[1]. Besides this, over the years, WebML has been subject to several extensions which have not been captured in the aforementioned DTD but are discussed in literature, only. In the context of this thesis, the recent extension of WebML with concepts allowing to model customization functionality of UWAs is particularly vital. Hence, Section 4.6, is dedicated to the introduction of WebML's new concepts

[1] The results of semi-automatically generating a MOF-based metamodel for the core WebML language have already been published in [SWK^{+}07]

for addressing customization as well as a discussion of their incorporation into the metamodel. The final WebML metamodel including concepts for customization modeling is available online[2]. While in Section 4.7 an overview of related work is given, a summary of this chapter is provided in Section 4.8.

4.1 Motivation

Metamodels are a prerequisite for MDE in general and consequently for Model-Driven Web Engineering (MDWE) in particular. As already stated in previous chapters, various modeling languages, just as in the web engineering field, however, are not based on metamodels and standards, like OMG's prominent Meta Object Facility (MOF) [OMG04]. While web modeling approaches originally were based on proprietary languages and rather focused on notational aspects, today more and more approaches do provide language specifications based on standards though not always from the model technical space (cf. Chapter 2). Consequently, MDE techniques and tools cannot be deployed for such languages, which prevents exploiting the full potential of MDE in terms of standardized storage, exchange, and transformation of models.

Amongst the approaches not yet in line with MDE, WebML [CFB$^+$03] is one of the most elaborated web modeling languages stemming from academia and is supported already over several years by the commercial tool WebRatio as already explained in Chapter 2. WebML's language concepts are partly defined in terms of XML document type definitions (DTDs) [W3C06], i.e., a grammar-like textual definition, specifying an XML document's structure, and partly hard-coded within the corresponding modeling tool. In contrast to MOF, DTDs represent a rather restricted mechanism for describing modeling languages, e.g., with respect to expressiveness, extensibility as well as readability and understandability for humans. Furthermore, since WebRatio internally represents models in XML [W3C06], it uses XSLT for generating code directly from WebML models. In contrast to dedicated MDE code generation technologies, e.g., MOFScript[3], writing XSLT programs for code generation, however, is difficult and error-prone. Concerning these problems, a metamodel-based approach allows expressing transformation rules in a more compact and readable way by using existing MDE-conform code generation techniques or model transformation languages such as QVT [OMG05a] and ATL [JK06] in order to produce platform-specific models in an additional step before generating code.

Furthermore, over the years WebML has been subject to several extensions which have not been captured in the WebML DTD, e.g., additional concepts addressing context-awareness [CDMF07], service-enabled [MBC$^+$05], and workflow-based [BCFM06] web applications. These extensions have partly been implemented in WebRatio via the tool's plugin mechanism. With respect to the extensions for modeling UWA's, a prototype extension to the WebRatio tool has been described in literature [CDMF07].

In order to define WebML's language concepts in an MDE-suitable way and thus to bridge WebML to MDE, a MOF-based metamodel for WebML is a fundamental prerequisite. Considering the language's size, however, we refrain from re-modeling WebML from scratch, since this would be a cumbersome and error-prone process. Instead, the existing DTD-based language specification as well as constraints hard-coded within WebRatio shall be reused in a semi-automatic process for MOF-based metamodel generation from DTDs. This process is illustrated in Figure

[2] *www.wit.at/people/schauerhuber/aspectUWA*

[3] www.eclipse.org/gmt/mofscript/

4.1 and encompasses three phases, whereby the first two phases concern the semi-automatic generation of the basic WebML language concepts defined within the WebML DTD. During the first phase a preliminary version of the metamodel is automatically generated from the available DTD, while in the second phase this preliminary version is manually validated and refactored according to constraints captured within the WebRatio tool support. Moreover, in the context of this thesis, the recently introduced concepts for supporting customization need to be considered besides WebML's original modeling concepts. Hence, in a third phase, they need to be included into the MOF-based metamodel as well. In this respect, the literature concerning WebML's concepts for context-awareness [CDMF07] serves as the only input for a manual extension of WebML's metamodel towards customization.

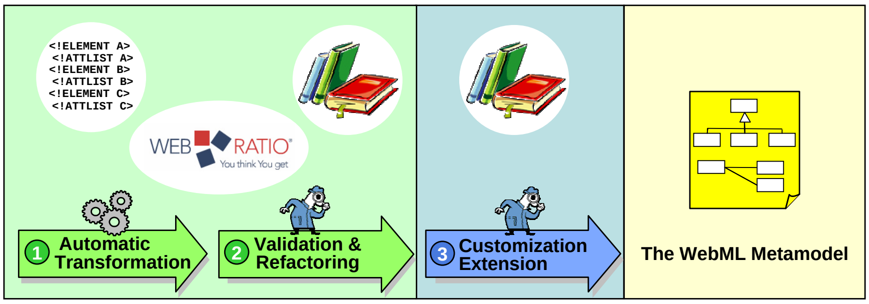


Figure 4.1: Process of Designing the WebML Metamodel

The following section explains the concepts of DTDs and MOF and points out the aforementioned DTD deficiencies.

4.2 DTDs and Ecore at a Glance

As a first step towards bridging WebML to MDE, this section elaborates on the expressiveness of DTDs, i.e., the concepts used to describe the WebML language, with respect to MOF. In the context of OMG's meta-level architecture [OMG05d], this means that a WebML model, which is represented by an XML document, relates to the instance level (M1) (cf. Figure 4.2).

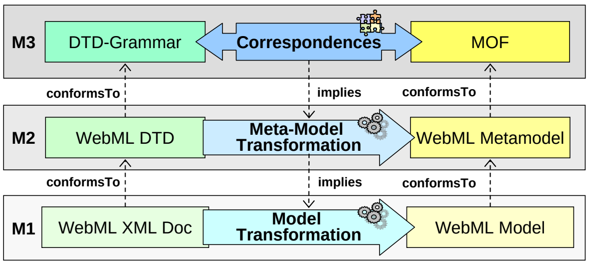


Figure 4.2: Interrelationships Between the Language Layers of DTD and MOF

Such a model has to conform to the WebML DTD describing the WebML language concepts at the meta-level (M2). The WebML DTD in turn is based on the DTD-grammar [W3C06] defined at the meta-meta-level (M3). Note that, while in case of WebML, a DTD is used to define a modeling language and therefore can be assigned to the M2 level, DTDs typically are used at M1 in order to describe the structure of data stored in XML documents. Analogously, MOF concepts defined at M3 are used to describe metamodels in the sense of MDE at M2. In the present case, this is the targeted WebML metamodel of which instances in terms of WebML models can be formulated at M1. This discussion shows that the two M3 level formalisms, in terms of the DTD-grammar and MOF respectively, represent the concepts on which to identify correspondences. In turn, these correspondences serve as a basis for M2-level done by the framework presented in Section 4.3.

In the following, UML class diagrams [OMG05d] are used as a common formalism to explain and to illustrate the major concepts of the DTD-grammar (cf. Section 4.2.1) and MOF (cf. Section 4.2.2). This explanation serves as the basis for identifying differences in the expressiveness of the two meta-meta-languages (cf. Section 4.2.3).

4.2.1 Document Type Definition (DTD) Concepts

The UML class diagram given in Figure 4.3 is based on previous work [KKR04] and depicts those DTD concepts present in the WebML DTD. These concepts need to be considered for finding correspondences to MOF concepts and consequently are reviewed briefly in the following.

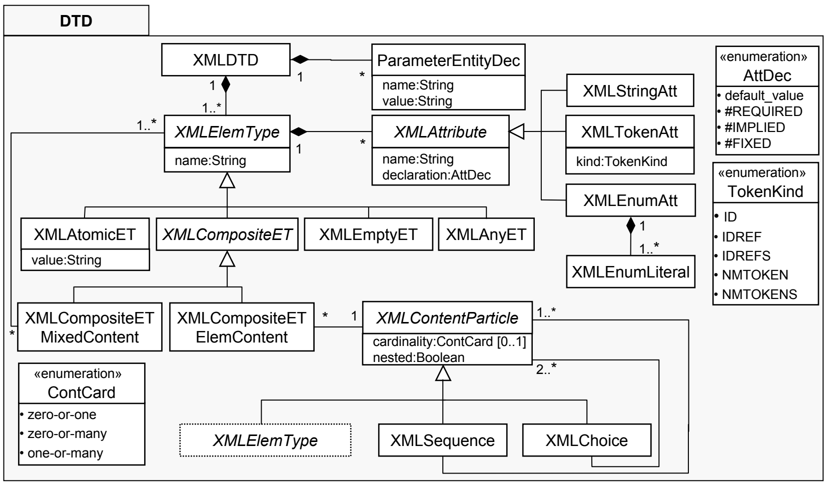


Figure 4.3: Overview of Relevant DTD Language Concepts

In general, DTD's contain markup declarations comprising element types (`XMLElemType`) and attributes (`XMLAttributes`) for defining the logical structure as well as primarily entity declarations (e.g., `ParameterEntityDec`), as a reuse mechanism for certain re-occurring markup declarations, for defining the physical structure.

Element types, being first-class citizens in DTDs, have a name and are specialized into `XMLAtomicET` (contains no other element types but character data), `XMLEmptyET` (no content is allowed), `XMLAnyET` (the content is not constrained), `XMLCompositeETMixedContent` (a mix of character data and child element types), and `XMLCompositeETElemContent` (consists of an `XMLContentParticle`). An `XMLContentParticle` either is an `XMLSequence`, an `XMLChoice`, or an `XMLElemType`. An `XMLChoice` or an `XMLSequence` can be enclosed in parentheses for grouping purposes and suffixed with a '?' (zero or one occurrences), '*' (zero or more occurrences), or '+' (one or more occurrences), whereas the absence of a particular symbol denotes a cardinality of exactly one.

Attribute declarations define one or multiple `XMLAttributes` (i.e., name-value pairs) for a single element type. Each `XMLAttribute` has a name, a data type, and a default declaration. The most commonly used data types for attributes are: `CDATA` (`XMLStringAtt`), `ID`, `IDREF` (refers to one `ID`-typed element), `IDREFS` (refers to multiple `ID`-typed elements), and Enumeration (`XMLEnumAtt`). For default declarations there are four possibilities: `#IMPLIED` (zero or one), `#REQUIRED` (exactly one), `#FIXED` (the attribute value is constant and immutable), and `Literal` (the default value is a quoted string).

Please note that, the order constraints imposed by DTDs and the majority of physical structures of the DTD-grammar (i.e., general entity declarations, notation declarations as well as `XMLAttributes` of type `ENTITY`, `ENTITIES`, and `NOTATION`) are ignored, since they are actually relevant to XML documents than to DTDs and the purpose of finding correspondences to MOF concepts is rather questionable [LM05].

4.2.2 MOF Concepts in Terms of Ecore

In the following, a brief overview of the most important concepts of MOF with respect to finding correspondences to DTD concepts is given. Note that, by the time of writing there is no standardized implementation of MOF 2.0 available. Therefore, in this thesis, Ecore - a slightly modified Essential MOF (EMOF) implementation in Java - which is provided by the widespread Eclipse Modeling Framework (EMF) [BSM+04], is used. In Figure 4.4, the most important concepts of Ecore with respect to finding correspondences to DTD concepts are summarized.

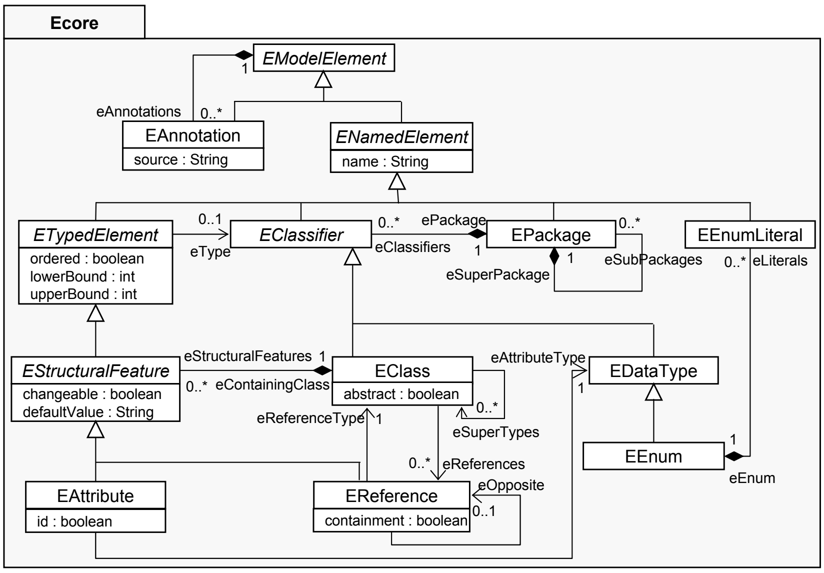


Figure 4.4: Overview of Relevant Ecore Language Concepts

In Ecore there is a single root concept (`EModelElement`) being the base class for all modeling elements. Its sub-class `EAnnotation` is used for describing additional information which cannot be presented directly in Ecore-based metamodels. ENamedElement is the base for the remaining Ecore modeling elements, because it provides for a 'name' meta-attribute. An EClassifier represents a type in a model and as such, is the base class for EClass and EDataType, whereas an ETypedElement serves as the base for other modeling concepts having a type such as EStructuralFeature, which in turn represents a structural feature of an EClass. EClasses are the first-class citizens in Ecore-based metamodels. An EClass may have multiple EReferences and EAttributes for defining its structural features as well as multiple super-EClasses. An EAttribute is part of a specific EClass and can have, as any ETypedElement, a lower and an upper bound multiplicity. Additionally, it can be specified as being able to uniquely identify the instance of its containing EClass (cf. 'id' meta-attribute) and as being ordered. The type of an EAttribute is either a simple EDataType or an enumeration. EString, EBoolean, EInt, and EFloat are part of Ecore's default data type set. EEnum allows to model enumerations defined by an explicit list of possible values, i.e., its literals (cf. EEnumLiterals). Analogous to EAttribute, an EReference is part of a specific EClass and can have a lower and an upper bound multiplicity. An EReference refers to an EClass and optionally to an opposite EReference for expressing bi-directional associations. Besides, an EReference can be declared as a being ordered and as a containment reference. EPackages group related EClasses, EEnums, as well as related EPackages. Each element is directly owned by an EPackage and each EPackage can contain multiple model elements.

4.2.3 DTD Deficiencies

When comparing a language specified in Ecore to one specified on the basis of DTDs, it is obvious that DTDs considerably lack extensibility, readability, and understandability for humans, and above all expressiveness [LM05]. In the following, the major deficiencies of DTDs when used as a mechanism for defining modeling languages are described. Note that some of these deficiencies have been resolved with the introduction of XMLSchema [W3C04d], such as limited set of data types (cf. Section 4.2.3.1), awkward cardinalities (cf. Section 4.2.3.4), missing inheritance concept (cf. Section 4.2.3.6), and lack of an explicit grouping mechanism (cf. Section 4.2.3.7). A profound comparison between DTD and XMLSchema can be found in Lee et al. [LC00]. Nevertheless, in the context of WebML, which is based on DTDs, the following shortcomings need to be addressed:

4.2.3.1 Limited Set of Data Types

In contrast to Ecore, DTDs have a limited set of data types that cannot be extended to support, e.g., Integer or Float data types. While the provided data types generally are based on Strings, some other data type may be simulated by defining an enumeration with specific literals. In this way, a Boolean attribute can be simulated by an attribute of type Enumeration having two literals, e.g., 'true' and 'false'. Enumerations, however, cannot capture numeric data types such as Integer or Float, which are naturally infinite.

4.2.3.2 Unknown Referenced Element Type(s)

DTDs referencing mechanism is based on IDREF(S)-typed attributes, which are able to reference any element type having an ID-typed attribute. Unlike Ecore, which provides typed references, it is not possible to identify the element type that may be referenced from an IDREF(S)-typed attribute based on the information given in DTDs. DTDs even allow to reference different element types. These referenced element types potentially have a common super-type, which, however, cannot be specified in the DTD. Due to this peculiarity of DTDs, it is neither possible to determine which element type(s) may be referenced based on the information given in the DTD nor if a certain set of element types may be referenced, only.

4.2.3.3 No Bi-directional Associations

While Ecore offers bi-directional associations, in DTDs only uni-directional references can be specified. There is no way to specify that two uni-directional references in combination form a bi-directional association either.

4.2.3.4 Awkward Cardinalities

DTDs offer a restricted mechanism to specify cardinalities. More specifically, in contrast to Ecore there are no explicit concepts for defining cardinalities having a lower bound greater than '1' and for defining cardinalities having an upper bound other than '1' or '*'. This can only be simulated in an awkward way by redundantly specifying a certain element type within the content specification of its related (parent) element type according to the required cardinality.

4.2.3.5 Missing Role Concept

In DTDs, there is no explicit concept to express that an element type can be deployed in different contexts, i.e., a role concept such as in Ecore is missing. Thus, in DTDs this is sometimes bypassed by defining each role as a separate element type each named after the specific role they represent, and redundantly defining the same content and attribute specifications.

4.2.3.6 Missing Inheritance Concept

DTDs are not able to express inheritance relationships between element types as naturally provided for Ecore. Hence, DTDs cannot profit from the typical benefits of inheritance such as reuse.

4.2.3.7 No Explicit Grouping Mechanism

There is no explicit mechanism to group parts of a DTD that semantically belong together as it is supported in Ecore. Nevertheless, this deficiency can be bypassed by encapsulating parts of a DTD in separate files and employing parameter entities to import these separated definitions where appropriate.

4.2.3.8 Missing Constraint Mechanism

A mechanism for defining complex constraints, as it is supported in Ecore by using the OCL standard [OMG05c], is not provided for DTDs. Thus, even simple XOR constraints, which are often required in metamodels, cannot be specified. This deficiency is specifically problematic, since possible ambiguities in DTDs cannot be resolved and XML documents, while valid according to their DTD, might still not represent the domain data correctly.

4.3 A DTD to Ecore Transformation Framework

On the basis of the discussion of DTD and Ecore concepts as done in the previous section, it is now possible to give more insight into the semi-automatic transformation approach, which is based on previous work [SWK06]. Generally, the transformation approach consists of an automatic phase and a manual phase (cf. Figure 4.5). The first phase is responsible for automatically generating a first version of the WebML metamodel and is performed by a component called *MetaModel-Generator* (MMG). The metamodel generator employs, in a first step, a set of transformation rules expressing all identified non-ambiguous correspondences between DTD concepts and Ecore concepts (cf. Section 4.3.1). In a second step of that phase, a set of heuristics is applied, dealing mainly with the aforementioned deficiencies by proposing possible correspondences (cf. Section 4.3.2). On the basis of these suggestions, in the second phase, the user needs to manually validate the generated metamodel and refactor it accordingly (cf. Section 4.3.3). The implementation architecture of the transformation framework is presented in Section 4.3.4.

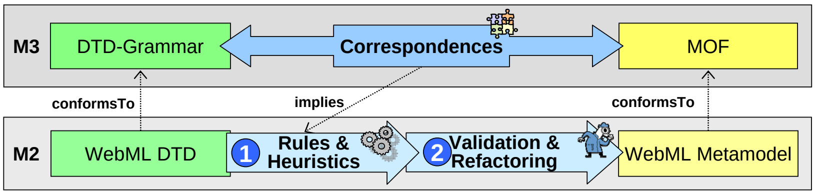


Figure 4.5: Two Phase Semi-Automatic Transformation Approach

To illustrate the transformation approach, a small sub-set of the WebML DTD is used to show the effects of applying transformation rules, heuristics and refactoring steps in terms of the resulting WebML metamodel (M2). In particular, this small sub-set consists of part of the concepts provided by WebML to represent a web application's content layer which in fact resembles the well-known ER-model [Che76]. It has to be emphasized that the focus in this section is on the illustration of the transformation approach, i.e., the consecutive application of (some) transformation rules, heuristics, and refactoring steps. As a consequence, using an example requiring concepts for modeling a web application's hypertext has been avoided, since the concepts are too numerous and often relate to concepts defined for the content layer. In this respect, the WebML content layer serves as a self-contained and small example. For those rules and heuristics that cannot be illustrated in the context of WebML's content model, an abstract example is provided in this section as well as a reference to an illustration of their concrete application in the context of WebML's hypertext model in Section 4.4.

4.3.1 Transformation Rules

A couple of rules for transforming concepts of DTDs into Ecore concepts have been designed. Table 4.1 summarizes these rules by differentiating between rules for `XMLElementTypes`, `XML-Attributes` and XOR-Constraints, denoting DTD concepts on the left-hand side and their Ecore counterparts on the right-hand side. Rules are marked using a decimal numbering schema and may contain sub-rules, further specializing the correspondences between DTD concepts and Ecore concepts. Finally, alternative correspondences depending on the concrete DTD concept are depicted by a distinction of cases. Following, the application of some of these rules are illustrated using the running example introduced above.

	Rule	DTD Concept		Ecore Concept
XML Element Type	**R 1**	*XMLElemType (ET)*		*EClass*
		XMLElemType. Name		*EClass.name*
	(1)	XMLEmptyET		no additional elements required
	(2)	XMLAnyET		no additional elements required
	(3)	XMLAtomicET		add EAttribute EAttribute.name="PCDATA", EAttribute.eAttributeType=EString, EAttribute.defaultValue=XMLAtomicET.value
	(4)	XMLCompositeET ElemContent	If XMLSequence with cardinality=1 and nested=false	add EReference for each XMLElementType in XMLSequence EReference.name=XMLElementType.name, EReference.containment=true
			If XMLChoice with cardinality=1 and nested=false	add EReference for each XMLElementType in XMLChoice EReference.name=XMLElementType.name, EReference.containment=true add OCL constraints restricting the alternative EReferences
			If XMLContentParticle with cardinality>1 or nested=true	add helper EClasses for each XMLSequence or XMLChoice serving as containers for nested XMLContentParticles

	(5)	XMLCompositeETMixedContent		add EReference for each XMLElementType
				EReference.name=XMLElementType.name, EReference.containment=true
				add EAttribute
				EAttribute.name="PCDATA", EAttribute.eAttributeType=EString, EAttribute.defaultValue= XMLCompositeETMixedContent.value

	R1.1	*XMLContentParticle.cardinality*		*EReference.multiplicity*
	(1)	? (Zero-or-one)		EReference.lowerBound=0, EReference.upperBound=1
	(2)	* (Zero-or-more)		EReference.lowerBound=0, EReference.upperBound=-1
	(3)	+ (One-or-more)		EReference.lowerBound=1, EReference.upperBound=-1
	(4)	no symbol		EReference.lowerBound=1, EReference.upperBound=1
XML Attribute	**R2**	*XMLAttribute*		*EAttribute*
		XMLAttribute.name		*EAttribute.name*
	(1)	XMLStringAtt, NMTOKEN(S), IDREF(S)		EAttribute.eAttributeType=EString
	(2)	ID		EAttribute.eAttributeType=EString, EAttribute.id=true

	(3)	XMLEnumAtt		add EEnum
				EEnum.name= XMLEnumAtt.name+"_ENUM"
				for each XMLEnumLiteral add EEnumLiteral
				EAttribute.eAttributeType=EEnum

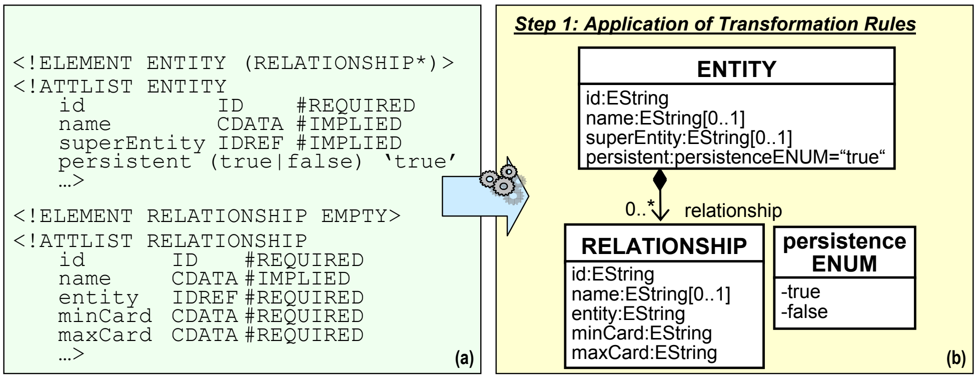
	R2.1	*XMLAttribute.cardinality*		*EAttribute.multiplicity*
	(1)	default_value	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1, EAttribute.defaultValue=XMLAttribute.default_value
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1, EAttribute.defaultValue=XMLAttribute.default_value
	(2)	#FIXED	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1, EAttribute.defaultValue=default_value, EAttribute.unchangeable=true
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1, EAttribute.defaultValue=default_value, EAttribute.unchangeable=true
	(3)	#REQUIRED	Single-valued	EAttribute.lowerBound=1, EAttribute.upperBound=1
			Multi-valued	EAttribute.lowerBound=1, EAttribute.upperBound=-1
	(4)	#IMPLIED	Single-valued	EAttribute.lowerBound=0, EAttribute.upperBound=1
			Multi-valued	EAttribute.lowerBound=0, EAttribute.upperBound=1
XOR	R3	If XMLElemType is part of several XMLCompositeETElemContent		then add OCL constraint to contained EClass specifying that the produced EReferences are exclusive

Table 4.1: Transformation Rules from DTD to Ecore

4.3.1.1 Rule 1 - Element Type

For each `XMLElemType` an `EClass` is created and the name of the `EClass` is set to the element type's name. Depending on the particular sub-class of `XMLElemType`, additional metamodel elements have to be created in the transformation process, which is outlined in Table 4.1.

Example. In Figure 4.6*(a)*, the WebML DTD specifies amongst others element types for *ENTITY* and *RELATIONSHIP*, since WebML's content model is based on the ER-model. According to Rule 1, two `EClasses` are generated and named after the element types (cf. Figure 4.6*(b)*). In addition, the ENTITY `XMLCompositeETElemContent` contains the *RELATIONSHIP* `XMLEmptyET`, and with respect to case (4) in Table 4.1 an `EReference` is produced, specifying *RELATIONSHIP* as the contained `EClass`.



```
<!ELEMENT ENTITY (RELATIONSHIP*)>
<!ATTLIST ENTITY
    id          ID    #REQUIRED
    name        CDATA #IMPLIED
    superEntity IDREF #IMPLIED
    persistent  (true|false) 'true'
    ...>

<!ELEMENT RELATIONSHIP EMPTY>
<!ATTLIST RELATIONSHIP
    id       ID    #REQUIRED
    name     CDATA #IMPLIED
    entity   IDREF #REQUIRED
    minCard  CDATA #REQUIRED
    maxCard  CDATA #REQUIRED
    ...>
```

Figure 4.6: Example of Applying the Transformation Rules (Step 1)

4.3.1.2 Rule 1.1 - Content Particle Cardinality

Each `XMLContentParticle` may have a certain cardinality, which is represented in metamodels through the `EReference`'s multiplicity in terms of lower and upper bound.

Example. Considering the running example in Figure 4.6*(b)*, according to this rule the cardinality of the relationship from *ENTITY* to *RELATIONSHIP* is set to '0..*'.

4.3.1.3 Rule 2 - Attribute

For each `XMLAttribute` an `EAttribute` is created and attached to the `EClass` representing the `XMLElemType`, which in turn owns the `XMLAttribute`. The name of the `EAttribute` is set to the name of the `XMLAttribute`. The data type of `XMLAttribute` is one of the following: `CDATA`, `NMTOKEN`, `NMTOKENS`, `ID`, `IDREF`, `IDREFS`, and `Enumeration`. Each of these possibilities requires an appropriate transformation as is outlined in Table 4.1.

Example. The example in Figure 4.6*(b)*, shows that all `XMLAttributes` of type `ID`, `CDATA`, and `IDREF` have been transformed into `EAttributes` of type `EString`, while the `XMLEnumAtt` *persistent* has been transformed to an `EEnum` having two `EEnumLiterals`.

4.3.1.4 Rule 2.1 - Attribute Cardinality

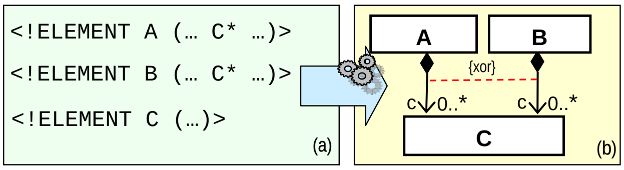
Attributes in both, DTDs and Ecore have a certain kind of cardinality. In DTDs, the cardinality of an `XMLAttribute` is determined on the one hand by the differentiation between single-valued (e.g., `ID`, `CDATA`, `IDREF`, `NMTOKEN`, and `XMLEnumAtt`) and multi-valued (e.g., `IDREFS`, `NMTOKENS`) attributes and on the other hand by the `XMLAttribute` declaration (`#REQUIRED`, `#IMPLIED`, `#FIXED`, and `Literal`). Table 4.1 illustrates how `XMLAttribute` cardinalities are transformed into `EAttribute` multiplicities.

Example. In Figure 4.6*(b)*, all `XMLAttributes` are single-valued meaning that the upper bound is set to one. Only, the `EAttributes` *name* and *superEntity* of `EClass` *ENTITY* as well as *name* of `EClass` *RELATIONSHIP* have a multiplicity of '0..1' since their corresponding `XMLAttributes` have been defined `#IMPLIED`. The default value of the `EAttribute` *persistent* is set to one of the `EEnumLiterals`, i.e., 'true'.

4.3.1.5 Rule 3 - XOR Containment References

An `XMLElemType` can be part of an `XMLContentParticle` of different instances of `XMLCompositeETElemContent`. In the corresponding Ecore-based metamodel an `EClass` can participate as the contained element in an arbitrary number of containment references. At instance level, the contained object, however, can be contained by an instance of only one of the container `EClasses` at the same time. Hence, this rule adds an OCL constraint to the contained `EClass` specifying this restriction.

Example. In the abstract example in Figure 4.7*(a)*, the `XMLElemType` *C* is an `XMLContentParticle` of `XMLElemType` *A* and `XMLElemType` *B*. The corresponding metamodel in Figure 4.7*(b)* must ensure that an instance of `EClass` *C* is contained either by an instance of `EClass` *A* or by an instance of `EClass` *B*. Therefore, an XOR constraint is introduced between the relationship *c* from *A* to *C* and the relationship *c* from *B* to *C*. For an example application of Rule 3 in the context of WebML the reader is referred to Listing 4.3 in Section 4.4.3.



```
<!ELEMENT A (… C* …)>
<!ELEMENT B (… C* …)>
<!ELEMENT C (…)>
```

Figure 4.7: Rule 3 - XOR Containment References

4.3.2 Heuristics

As mentioned before, transformation rules are not enough to obtain an Ecore-based metamodel from a specific DTD, due to the deficiencies of DTDs described in Section 4.2.3. Thus, for resolving most of these deficiencies, a set of six heuristics (cf. Table 4.2) is proposed, exploiting the assumption that design patterns and naming conventions have been used by DTD designers that have also been found when analyzing the WebML DTDs. This means that the effectiveness of the heuristics, however, is strongly correlated with the quality of the design of the DTDs. For example, the heuristics operate more effectively if naming conventions are used, e.g., for `IDREF(S)`-

typed XMLAttributes, (cf. Heuristic 1 - IDREF(S) Resolution) or a common DTD design pattern [VIG05] for grouping related element types by splitting up a DTD into several external DTDs (cf. Heuristic 3 - Grouping Mechanism).

In any case, these heuristics propose possible correspondences along with annotations guiding the validation and refactoring step in phase 2. In this way, semantically rich language concepts of Ecore such as typed references, data types, and packages can be used to equalize the DTD deficiencies. Following, the application of some of the heuristics is shown in using the running example in Figure 4.8.

Heuristic	DTD Concept	Ecore Concept	DTD Deficiency Resolved
H1	If (XMLTokenAtt.kind=IDREF) AND (XMLElemType.name=XMLAttribute.name)	then add EReference to EClass with name= XMLElemType.name, EReference.name=XMLAttribute.name annotate with «Validate IDREF(S)»	Unknown Referenced Element Type(s)
	else	then annotate EAttribute with «Resolve IDREF(S) manually»	
H2	If XMLEnumAtt has two XMLEnumLiterals and XMLEnumAtt is one of {true, false}, {1, 0}, {on, off}, {yes, no}	then EAttribute.eAttributeType=EBoolean annotate with «Validate Boolean»	Limited Set Of Data Types
	else if XMLEnumAtt has two XMLEnumLiterals	then annotate EEnum with two EEnumLiterals with «Resolve possible Boolean type manually»	
H3	If DTD imports external DTDs	then add EPackages for each external DTDs to the root EPackage	No Explicit Grouping Mechanism)
H4	If the name of two or more XMLElemTypes in a XMLSequence are equal	then annotate container EClass with «Resolve multiplicity manually»	Awkward Cardinalities
H5	If XMLElemType has two or more XMLTokenAtts with declaration=#IMPLIED and (kind=IDREF or kind=IDREFS)	then annotate each EAttribute or EReference (cf. Heuristic 1) with «Resolve XOR constraint manually»	Missing Constraint Mechanism
H6	If XMLElemType is of type XMLAnyET	then annotate EClass with «Resolve XMLAnyET manually»	Missing Inheritance Concept

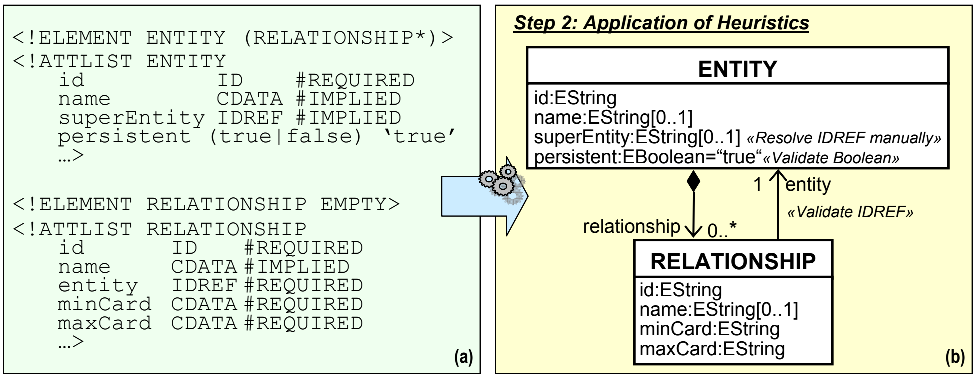
Table 4.2: Heuristics from DTD to Ecore

4.3.2.1 Heuristic 1 - IDREF(S) Resolution.

The first heuristic is based on the assumption that an IDREF(S)-typed XMLAttribute might be named after the XMLElemType it is intended to reference. Thus, although DTDs lack the possibility to explicitly specify the referenced element types (cf. Section 4.2.3.2 Unknown referenced element type(s)), it is possible to find them relying on naming conventions of element types and attributes. Heuristic 1 is intended to find such name matches in DTDs. If a match is found, an EReference is generated pointing to the identified EClass. In addition, the EReference is annotated with ≪*Validate IDREF(S)*≫. Furthermore, the multiplicity of the EReference is set to the multiplicity of the XMLAttribute.

It has to be emphasized that, since this heuristic is based on name matches, two problematic cases can occur. On the one hand, it may falsely resolve references in case IDREF(S) attributes are incidentally named like XMLElemTypes but in fact do not reference them. On the other hand, it may not be possible to resolve a reference in case IDREF(S) attributes are not named according to the `XMLElemType` they shall refer to. Consequently, the user has to validate if the resolution of the `IDREF(S)` is correct or if another `EClass` should be referenced.

Example. In the running example the `XMLAttribute` *entity* in Figure 4.8*(a)* is resolved to an `EReference` in Figure 4.8*(b)*. In case no name match is found, the `IDREF(S)`-typed `XMLAttribute` is transformed into an `EAttribute` of type `EString` annotated with ≪*Resolve IDREF(S) manually*≫, such as the *superEntity* `EAttribute` in Figure 4.8*(b)*.



```
<!ELEMENT ENTITY (RELATIONSHIP*)>
<!ATTLIST ENTITY
   id          ID    #REQUIRED
   name        CDATA #IMPLIED
   superEntity IDREF #IMPLIED
   persistent  (true|false) 'true'
   ...>

<!ELEMENT RELATIONSHIP EMPTY>
<!ATTLIST RELATIONSHIP
   id       ID    #REQUIRED
   name     CDATA #IMPLIED
   entity   IDREF #REQUIRED
   minCard  CDATA #REQUIRED
   maxCard  CDATA #REQUIRED
   ...>
```

Figure 4.8: Example of Applying the Heuristics (Step 2)

4.3.2.2 Heuristic 2 - Boolean Identification

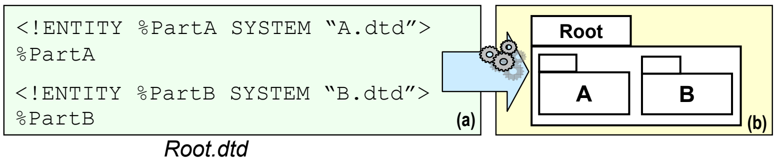
Heuristic 2 is based on the assumption that an `XMLEnumAtt` consisting of two `XMLEnumLiterals` might represent an attribute of type `Boolean` (cf. 4.2.3.1 Limited set of data types). It recognizes such optimization possibilities and, instead of generating an `EEnum`, produces an `EAttribute` of type `EBoolean` for the following sets of enumeration literals: {true, false}, {1, 0}, {on, off}, and {yes, no}. Furthermore, an annotation ≪*Validate EBoolean*≫ is added to the attribute. In case the two `XMLEnumLiterals` are not one of the aforementioned sets, the produced `EEnum` is annotated with ≪*Resolve possible EBoolean type manually*≫, indicating the possibility of replacing the `EEnum` by the `EBoolean` data type.

Example. In the running example the `XMLEnumAtt` *persistent* is identified to be of type `Boolean` (cf. Figure 4.8*(a)*). Thus, in the metamodel, the `EAttribute` *persistent* is of type `EBoolean` and no `EEnum` is generated (cf. Figure 4.8*(b)*).

4.3.2.3 Heuristic 3 - Grouping Mechanism

In Heuristic 3, a parameter entity declaration that points to a further DTD file is interpreted as representing a group of related markup declarations (cf. 4.2.3.7 No explicit grouping mechanism), that can be referenced from within a so called root DTD. A root DTD is equivalent to a root package in a metamodel and external DTDs are equivalent to sub-packages of the root package. Thus, a package for each external DTD and one root package for the root DTD needs to be generated.

Example. In the abstract example of Figure 4.9*(a)* a DTD named *Root* is shown which defines two `ParameterEntityDec` *PartA* and *PartB*, both referencing an external DTD, *A.dtd* and *B.dtd*. The DTD *Root* is transformed into the `EPackage` *Root* which contains an `EPackage` *A* and *B* for the external DTDs (cf. Figure 4.9 (b)). An example application of Heuristic 3 in the context of WebML may be found in Listing 4.1 in Section 4.4.1.

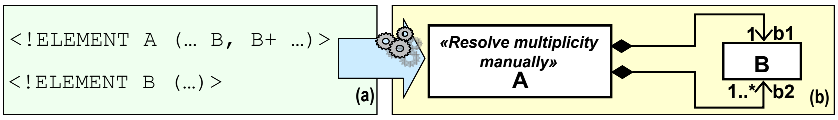


```
<!ENTITY %PartA SYSTEM “A.dtd”>
%PartA

<!ENTITY %PartB SYSTEM “B.dtd”>
%PartB
```

Figure 4.9: Heuristic 3 - Grouping Mechanism

4.3.2.4 Heuristic 4 - Cardinalities Identification

This heuristic is based on the assumption that an `XMLSequence` containing element types of the same name has to be interpreted as ”one piece of information” (cf. 4.2.3.4 Awkward cardinalities). This means that instead of producing an `EReference` for each single element type, only one `EReference` should be generated and the cardinality has to be inferred from all element type cardinalities within the sequence. Heuristic 4 adds an annotation ≪*Resolve cardinality manually*≫ to the `EClass` containing the `EReferences` to indicate that a specific sequence of element types transformed into a set of `EReferences` possibly has to be remodeled into one `EReference` and the appropriate multiplicity has to be assigned.



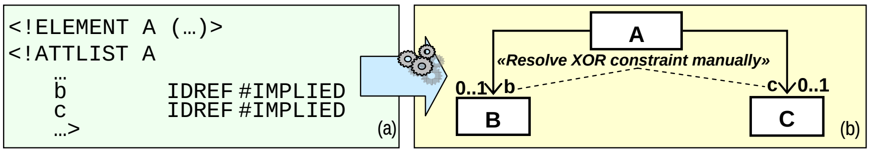
```
<!ELEMENT A (… B, B+ …)>

<!ELEMENT B (…)>
```

Figure 4.10: Heuristic 4 - Cardinalities Identification

Example. In Figure 4.10*(a)*, an example for the awkward cardinality deficiency of DTDs can be found. The problem is that the first and the second `XMLContentParticle` *B* of `XMLElemType` *A* together maybe represent one set of elements with a cardinality restriction of ’2..*’ instead of representing two separate sets of elements. Consequently, in the corresponding metamodel (cf. Figure 4.10*(b)*), this ambiguity is marked by the annotation ≪*Resolve cardinality manually*≫. This annotation indicates that in the validation and refactoring step the user has to decide, if *b1* and *b2* are two separate sets or if *b1* and *b2* should be merged into one set with a cardinality of ’2..*’. This heuristic has been applied for example in the context of WebML’s hypertext model (cf. Listing 4.2 in Section 4.4.3).

4.3.2.5 Heuristic 5 - XOR Constraints Identification

If two `XMLAttributes` within an attribute list declaration of an element type are both of type `IDREF(S)` and have been declared as `#IMPLIED` they might represent two excluding `EReferences` in the metamodel and hence require an XOR constraint. Heuristic 5 annotates these `EReferences` (or `EAttributes` if the reference could not be resolved automatically by Heuristic 1) with ≪*Resolve XOR constraint manually*≫ to indicate the possible need for an XOR constraint.



```
<!ELEMENT A (…)>
<!ATTLIST A
    …
    b        IDREF #IMPLIED
    c        IDREF #IMPLIED
    …>
```

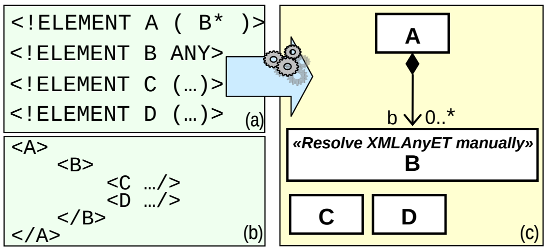
Figure 4.11: Heuristic 5 - XOR Constraints Identification

Example. In Figure 4.11*(a)* an excerpt of a DTD is shown which defines an `XMLElemType` *A* containing two attributes, namely *b* and *c*. Both attributes are `IDREF`-typed and defined as `#IMPLIED` which means both attributes are optional. In this example, Heuristic 1 is used to resolve the `IDREF` attributes as `EReferences`. In some cases, however, two optional `IDREF`-typed attributes are mutually exclusive. Therefore, `EReferences` *b* and *c* are annotated with the annotation ≪*Resolve XOR constraint manually*≫ indicating that the user must decide if actually an XOR constraint exists between these two relationships or not. For an example application of Heuristic 5 in the context of WebML see Listing 4.4 in Section 4.4.3.

4.3.2.6 Heuristic 6 - Inheritance Identification

This heuristic is based on the assumption that the element type `XMLAnyET` sometimes is used as a container element for other concepts in the described language, i.e., concepts that have similar properties and in this way may represent sub-types of the `XMLAnyET` element. Hence, Heuristic 6 annotates `EClasses` resulting from an `XMLAnyET` with ≪*Resolve XMLAnyET manually*≫ in order to propose a possible candidate for inheritance.

Example. In Figure 4.12*(a)*, the abstract example shows a DTD, in which `XMLElemType` *A* contains an `XMLElemParticle` *B*. `XMLElemType` *B*, in turn, is defined as `XMLAnyET` stating that element *B* can be any `XMLElemType` or any text. In Figure 4.12*(b)*, an example XML document is shown, where element *A* contains element *B* which in turn contains elements of type *C* and *D*. In Figure 4.12*(c)* the corresponding metamodel from the DTD definition contains amongst others an `EClass` *B* which is annotated with ≪*Resolve XMLAnyET manually*≫. This annotation indicates that the user has to decide, how to express the possibility that an instance of `EClass` *B* can contain instances of `EClass` *C* and *D*. In general, this possibility can be expressed as an inheritance relationship, defining `EClass` *B* as the super-class of `EClasses` *C* and *D*. In the context of WebML's hypertext model, an example application of Heuristic 6 is provided in Listing 4.5 in Section 4.4.4.



```
<!ELEMENT A ( B* )>
<!ELEMENT B ANY>
<!ELEMENT C (…)>
<!ELEMENT D (…)>
```

```
<A>
    <B>
        <C …/>
        <D …/>
    </B>
</A>
```

Figure 4.12: Heuristic 6 - Inheritance Identification

4.3.3 Manual Validation and Refactoring of the Generated Metamodel

The second, manual phase of the transformation framework requires user interaction for validating and refactoring the automatically produced metamodel on the basis of domain-knowledge and specifically on the basis of the suggestions annotated by the applied heuristics. In the example shown in Figure 4.13*(a)*, two annotations with respect to Heuristic 1 were introduced indicating that the user should validate, on the one hand, the directed reference introduced between *ENTITY* and *RELATIONSHIP* (≪*Validate IDREF*≫) and, on the other hand, the introduced attribute *superEntity* which was marked with ≪*Resolve IDREF(S) manually*≫. While the directed reference appears to be a correct transformation, the introduced attribute *superEntity* is, in fact, a reference to the *ENTITY* in its role as super-entity used to model inheritance in WebML and therefore shall be manually refactored accordingly by replacing the `EAttribute` with an `EReference` from *ENTITY* to itself with the role name *superEntity* (cf. Figure 4.13*(b)*). Furthermore, according to Heuristic 2, in the given example the `EAttribute` *persistent* has been annotated with ≪*Resolve possible EBoolean type manually*≫ to indicate the need of manual validation. In this case no manual refactoring is necessary and the annotation can be deleted.

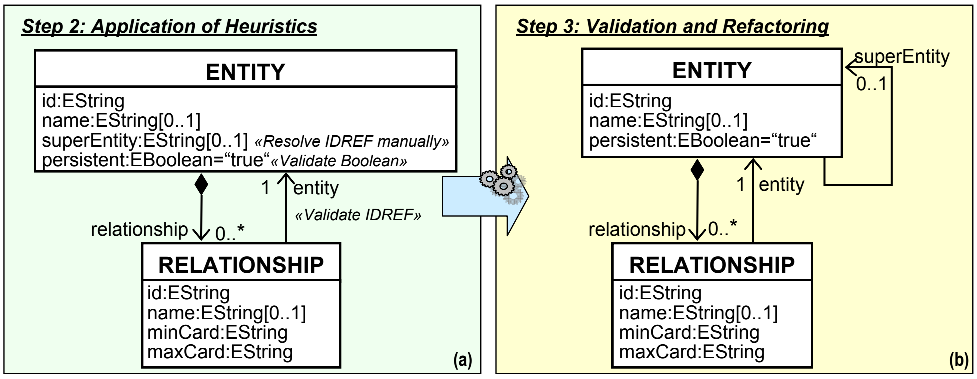


Figure 4.13: Example of Applying Manual Refactoring (Step 3)

4.3.4 Implementation Architecture of the MetaModelGenerator

As already mentioned, the core component of the transformation framework is represented by the MetaModelGenerator. Figure 4.14 shows the details of its implementation architecture. The MMG is based on the EMF and on an open source DTD parser[4]. In a first step a specific DTD, in this case, the WebML DTD, serves as input to the DTD parser, which builds a Java object graph of DTD markup declarations in memory. Then each element type in the object graph is visited and transformed according to the transformation rules and heuristics described in Section 4.3.1 and Section 4.3.2 respectively. Each transformation rule is implemented as a separate Java method which takes DTD element type objects as input and generates the objects for the corresponding Ecore elements. If a transformation rule uses a heuristic, then the corresponding method calls a helper method which implements that heuristic. As soon as the complete element object graph of the Ecore-based metamodel has been generated, the default XMI serializer of EMF is activated

[4] www.wutka.com/dtdparser.html

in order to serialize the metamodel as an XMI file. This XMI file can be loaded into OMONDO[5], a graphical editor for Ecore-based metamodels available as an Eclipse plug-in. In a last step, the annotations created to indicate that a heuristic has been applied, should be validated by the user and the metamodel should be refactored accordingly.

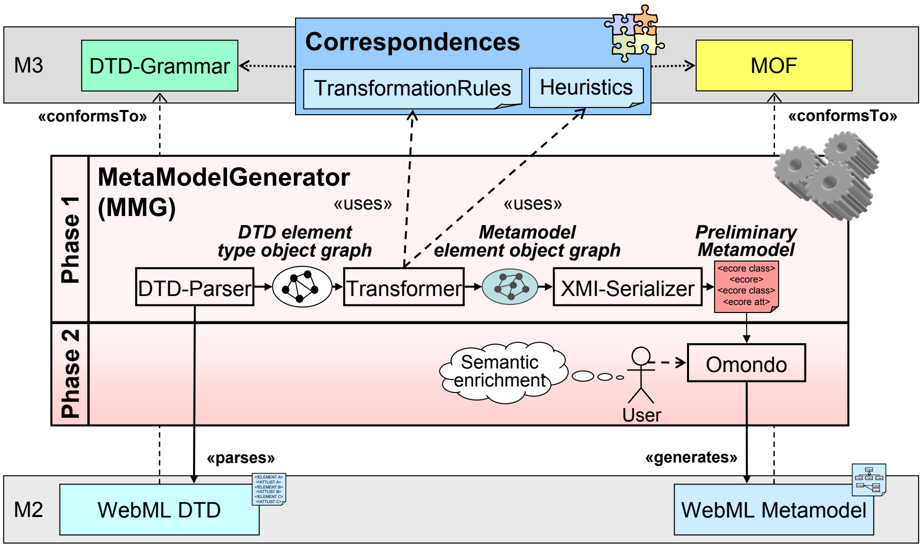


Figure 4.14: Architecture and Mode of Operation of the MMG

4.4 The Resulting WebML Metamodel

The Ecore-based metamodel for WebML resulting from the application of the DTD2MOF transformation framework to the WebML DTD is subject of this section. In particular, the rationale behind some of the manual refactoring decisions shall be explained. Additionally, the WebML metamodel shall be illustrated by relating it to a concrete modeling example, i.e., a demo WebML model that is shipped with WebRatio. This way, the intention is to briefly explain the language and notation to those unfamiliar with WebML and at the same time indicate the relationship between the model and the metamodel specification as well as informally show the equivalence of the metamodel with the original WebML DTD. A more profound evaluation of the WebML metamodel is provided in Section 4.5.

Please note that the following figures depicting some of the metamodel's packages have been simplified for readability purposes. For the same reason, XOR constraints are illustrated in UML syntax. For an in-depth description of each modeling concept the reader is referred to [CFB^{+}03]. Before describing some of the packages in more detail and explaining some of the refactoring actions (cf. Section 4.4.2 - 4.4.7), an overview on the overall package structure is given.

[5] www.omondo.com

4.4.1 Overall Package Structure

The WebML designers have used parameter entities as a mechanism to structure the WebML's language specification. Thus, the WebML language definition consists of several DTDs with *WebML.dtd* being the root DTD that imports the others, which is expressed in Listing 4.1.

Listing 4.1: WebML's Concepts Grouped With External DTDs

```
<!-- WebML.dtd -->
<!ENTITY % StructureDTD SYSTEM "Structure.dtd">
%StructureDTD;
<!ENTITY % NavigationDTD SYSTEM "Navigation.dtd">
%NavigationDTD;
<!ENTITY % PresentationDTD SYSTEM "Presentation.dtd">
%PresentationDTD;
...
```

While *Structure.dtd* and *Navigation.dtd* define the main language concepts that have been introduced in [CFB+03], other rather tool-related DTDs have been introduced in the WebRatio tool. In contrast to previous work [SWK06], where the main focus has been WebML's main language concepts, in this thesis all of WebML's DTDs are considered. This allows for migrating existing WebML models that have been generated using WebRatio into models conforming to the metamodel specification without loosing any information (cf. Section 4.5.1) and thus profiting from further MDE techniques such as model transformation.

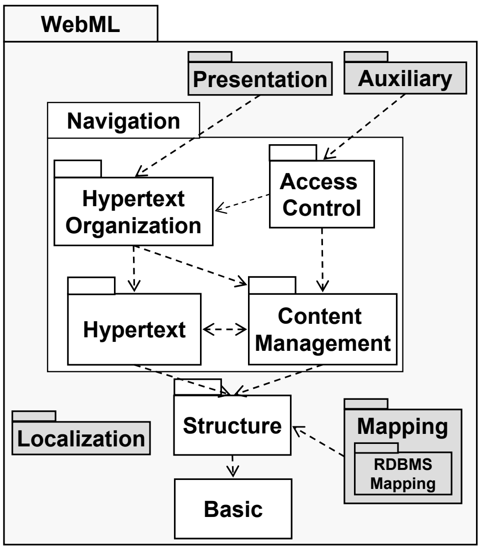


Figure 4.15: WebML Packages View

Figure 4.15 presents a bird's eye view of the resulting WebML metamodel, i.e., its packages and their interrelationships. This structural organization of the WebML concepts has been automatically generated on the basis of Heuristic 3. While *Structure.dtd* corresponds to the *Structure* package in Figure 4.15 and contains concepts for modeling the content level of a web application, the *Navigation* package contains modeling concepts for the hypertext level and has been automatically generated from the *Navigation.dtd*. The rather large *Navigation* package has been manually reorganized into four sub-packages, namely *HypertextOrganization*, *Hypertext*, *ContentManagement*, and

AccessControl. In addition, the package *Basic* has been introduced, which includes typical abstract concepts, e.g., ModelElement and NamedElement, from which all other WebML concepts are derived. The additional gray-shaded packages have been generated from the tool-related DTDs: First, the *Mapping* package imports the *RDBMSMapping* package which provides concepts for specifying the mapping of WebML's content model to a relational database, second, the *Localization* package offers modeling concepts for multilingual web applications, third the *Presentation* package defines concepts for modeling the *Look & Feel* of web applications, and fourth, the graphical illustration and positioning of WebML's notational elements within the WebRatio modeling editor is determined by concepts defined in the *Auxiliary* package.

In the following, a detailed description of the tool-related packages is omitted in favor of presenting the actual WebML language, i.e., the Structure, Navigation, and Basic packages, as well as some of the applied refactoring actions. In order to better illustrate the semantics of the metamodel, the corresponding part of a concrete WebML modeling example will be provided for each package. For this reason, the *ACME* (A Company Manufacturing Everything) example model, which is a demo WebML model shipped with WebRatio, shall be used. It represents a company's website where users can browse and search products as well as special combinations of products. These products and combinations can be edited, extended, and deleted by administrators of the web application.

4.4.2 Structure Package

The Structure package (cf. Figure 4.16*(a)*) contains modeling concepts that allow modeling the content layer of a web application, which regards the specification of the data used by the application. Since, as already mentioned, WebML's content model is based on the ER-model, it basically supports ER modeling concepts: An Entity represents a description of common features, i.e., Attributes, of a set of objects. Note, that unlike UML class diagrams, ER diagrams model structural features, only.

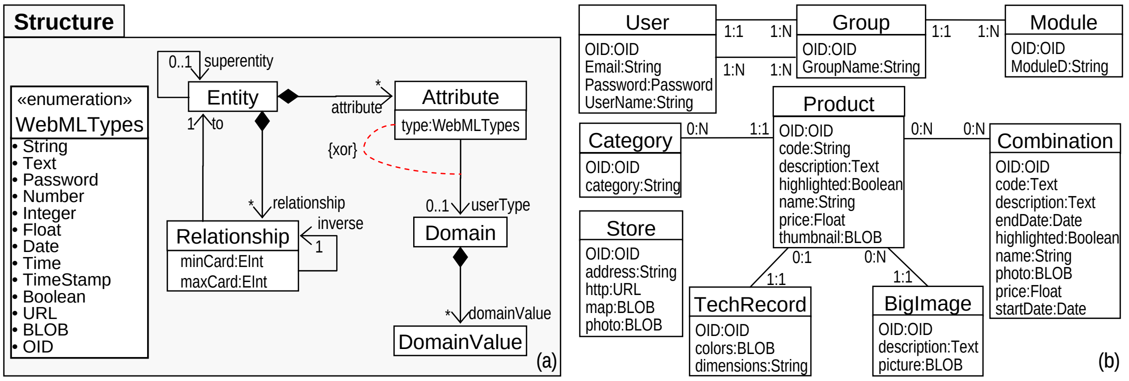


Figure 4.16: Structure Package

With respect to manual refactoring actions, an XOR constraint has been added to the metamodel in order to specify that Attributes can have either a predefined *type*, e.g., String, Integer, Float, Date, Time, and Boolean, or a *userType*, i.e., an enumeration type represented by Domain and DomainValue, respectively. Entities that are associated with each other are connected by Relationships whereby the type of the meta-attributes *minCard* and *maxCard* of Relationship have been changed from `EString` to `EInt`.

In Figure 4.16*(b)*[6], the WebML content model of the ACME web application is depicted. *Products* belong to one *Category* and can be described by a *TechnicalRecord* and several *BigImages*. Furthermore, *Products* can be offered within several *Combinations* with other *Products*. In addition, the web application provides information of available *Stores*. The *User*, *Group*, and *Module* entities are used for user management (e.g., normal users and administrators) and access control purposes.

4.4.3 HypertextOrganization Package

The HypertextOrganization package includes concepts for structuring the hypertext, i.e., it offers concepts for organizing modeling concept from the Hypertext package (cf. Section 4.4.4). More specifically, the Page concept is used to organize and structure information from the content level, e.g., ContentUnits from the Hypertext package. Siteviews and Areas in turn group Pages as well as operations on data from the content level, e.g., OperationUnits from the ContentManagement package (cf. Section 4.4.5). More specifically, Siteviews represent groups of areas and/or pages devoted to fulfilling the requirements of one or more user groups, while Areas are containers of Pages or nested sub-areas related to a homogeneous subject and are used to hierarchically organize the web application. These concepts are encapsulated within the HypertextOrganization package (cf. Figure 4.17*(a)*).

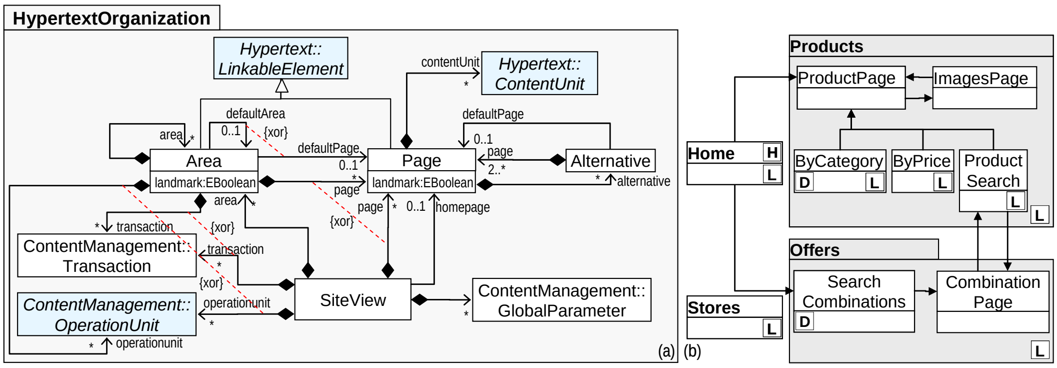


Figure 4.17: HypertextOrganization Package

With respect to refactoring actions, it was possible to identify an example of the awkward cardinalities problem (cf. Section 4.2.3.4) based on `EAnnotations` created by Heuristic 4. The definition of the Alternative concept requires the Alternative to have at least two sub-pages, which is expressed in the WebML DTD as is depicted in Listing 4.2.

Listing 4.2: Alternative has two or More Sub-Pages

```
<!ELEMENT Alternative (Page, Page+)>
```

Yet, this definition found in the DTD might be interpreted differently in a metamodel. One possible interpretation is that the first `XMLContentParticle` represents a special page, e.g., a de-

[6] For readability reasons, we do not incorporate "instance-of" relationships from the modeling example part to the metamodel part of the figure.

fault page. The correct interpretation in the context of WebML [CFB$^+$03] is, however, that the first and the second `XMLContentParticle` together represent one set of alternative Pages, i.e., one containment `EReference`, but with special restrictions on their cardinalities, i.e., 2..*. In metamodels, this constraint can be expressed unambiguously, which is shown by the *Alternative.page* reference in Figure 4.17*(a)*.

While Rule 3 already detected, that Pages and Transactions can be contained by either a Siteview or an Area (cf. Listing 4.3), Heuristic 5 identified further possible candidates for XOR constraints in the HypertextOrganization package.

Listing 4.3: Page is Either Placed Within a Siteview or an Area

```
<!ELEMENT Siteview (... Page* ...)>
<!ELEMENT Area (... Page* ...)>
```

In Listing 4.4, an Area can have either a *defaultArea* or a *defaultPage*, but not both at the same time.

Listing 4.4: Area has either a defaultPage or a defaultArea

```
<!ELEMENT Area (...)>
<!ATTLIST Area
    defaultPage IDREF #IMPLIED
    defaultArea IDREF #IMPLIED
    ...>
```

In the DTD the attribute list declaration of Area is not able to ensure this constraint at the instance level. Therefore, an XOR constraint has been introduced to specify that either the *defaultPage* `EReference` or the *defaultArea* `EReferences` occurs at the instance layer:

```
context Area inv:
    defaultArea.oclIsUndefined()<>defaultPage.oclIsUndefined()
```

In the ACME WebML model, separate Siteviews for users and administrators have been designed. The first one, the *Web* Siteview, is depicted in Figure 4.17*(b)*. The *Products* Area groups all Pages presenting some information about products and the *Home* Page (H) acts as the entry point of the Siteview. The default page of an Area (D) such as the *ByCategory* Page of the *Products* Area is the one displayed when the Area is entered. Furthermore, Pages and Areas declared as *landmark* (L) are reachable from all other Pages or Areas within their enclosing Siteview or enclosing Area. In this respect, the *landmark* represents a compact way of specifying a set of links to a Page or Area, respectively.

4.4.4 Hypertext Package

The hypertext layer represents a view on the content layer of a web application, only, and thus, the Hypertext package reuses concepts from the Structure package, namely, Entity, Relationship, and Attribute. The Hypertext package (cf. Figure 4.18*(a)*) summarizes ContentUnits used, for example, to display information from the content layer, which may be connected by Links in a certain way. Based on Heuristic 6, a candidate `EClass` for introducing inheritance has been identified. In WebML, Pages contain different kinds of ContentUnits (cf. Listing 4.5).

Listing 4.5: Page Contains Different Kinds of ContentUnits

```
<!ELEMENT Page (ContentUnits, ...)>
<!ELEMENT ContentUnits ANY>
```

The `XMLAnyET`, however, does not restrict which element types are allowed, i.e., only ContentUnits, and which are not allowed at the instance layer. Again, these constraints have to be ensured by the WebRatio tool. In the metamodel, this problem could be resolved by manually introducing a generalization hierarchy, which includes the additional abstract classes *ContentUnit*, *DisplayUnit*, and *SortableUnit*. This way, it can be ensured that Pages contain sub-classes of ContentUnit, only, and handle the large amount of different kinds of ContentUnits more easily by reducing redundant structural feature definitions.

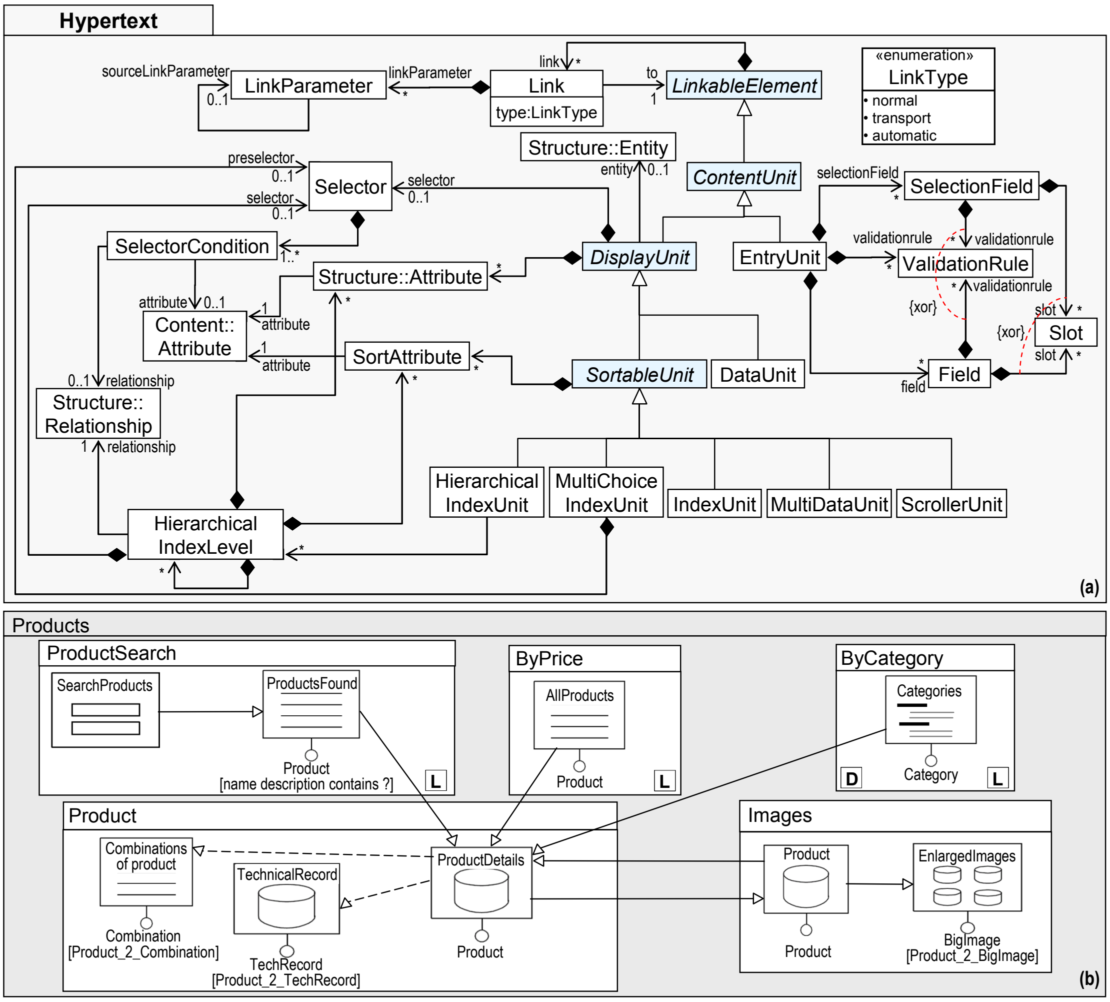


Figure 4.18: Hypertext Package

The abstract class *LinkableElement* has been manually introduced in order to cope with other language concepts that can also be connected by Links. This was necessary, since the `IDREF`-typed *XMLAttribute to* of the Link `XMLElemType` declaration does not restrict the referenced elements to those that the designer originally intended to reference (cf. Listing 4.6).

Listing 4.6: Link Targets are not Specified

```
<!ELEMENT Link (...)>
<!ATTLIST Link
    to IDREF #REQUIRED
    type (normal|automatic|transport) 'normal'
    ...>
```

Furthermore, besides ContentUnits, there are other LinkableElements in the HypertextOrganization package (cf. Section 4.4.3), namely Page and Area, as well as in the ContentManagement package (cf. Section 4.4.5), namely OperationUnits. More specifically, three disjoint LinkTypes are available in WebML, i.e., *normal*, *automatic*, and *transport* (cf. Figure 4.18*(a)*). Besides this Link concept, there are also the OKLink and KOLink modeling concepts from the ContentManagement package, which are specifically used to define Links from OperationUnits to other LinkableElements. Consequently, there are multiple sourceElement-link-targetElement tuples of which some are allowed in WebML, only (cf. Table 4.3).

From\To	Content Unit	Operation Unit	Page	Area
Content Unit	*normal* *automatic* *transport*	*normal* *transport*	✗	✗
Operation Unit	*transport* *OK* *KO*	*transport* *OK* *KO*	*transport* *OK* *KO*	*transport* *OK* *KO*
Page	✗	*normal* *transport*	*normal* *transport*	*normal*

Table 4.3: Linking Possibilities in WebML

These sourceElement-link-target Element tuples, however, are not restricted by the WebML DTD but are implicitly ensured within the WebRatio tool. Aiming at a precise definition of sourceElement-link-targetElement tuples, in the WebML metamodel, the introduction of the LinkableElement concept, which acts as a super-class for all possible sources and targets, is not enough. Consequently, a set of appropriate OCL constraints restricting the tuples to those that are allowed in WebRatio have been introduced (cf. Table 4.3). For example, a Page cannot link ContentUnits, which can be specified with the following OCL constraint:

```
context Page inv:
    self.link->forAll(l | not l.to.oclIsTypeOf(ContentUnit))
```

Figure 4.18*(b)* shows a refined view of the *Web* Siteview presented in Figure 4.17*(b)* depicting in detail the *Products* Area: While the *ByPrice* Page uses the IndexUnit *AllProducts* for listing links to all products in ascending order according to their price, the *ByCategory* Page displays a linked list of all products organized according to their categories using a HierarchicalIndexUnit. The *ProductSearch* Page provides an EntryUnit *SearchProducts* with one Field where the user can enter a keyword and displays the found products as an IndexUnit. A single SelectorCondition of the IndexUnit's Selector defines that only those products are to be retrieved, where the keyword is part of the name or the description of the product. A specific product is shown by the *Product* Page, which is linked by all other Pages via Links of type *normal*. The *ProductDetails* DataUnit represents one product from the content model and displays the specified Attributes, only. Furthermore, an additional DataUnit retrieves the technical record of the product and an additional IndexUnit displays a linked list of combinations where the specific product is part of. The information about what technical records and what combinations to retrieve is transported via LinkParameters of Links of type *transport* (dashed arrows), which are neither navigable by nor visible to users. Finally, the *Images* Page again shows some details of a product using a DataUnit and a set of images of the product using a MultiDataUnit.

4.4.5 ContentManagement Package

The ContentManagement package contains modeling concepts that allow the modification of data from the content layer. Similar to the generalization hierarchy in the Hypertext package, additional abstract classes have been introduced to the ContentManagement package on the basis of `EAnnotations` created by Heuristic 6 (cf. Figure 4.19*(a)*), i.e., *OperationUnit*, *ContentManagementUnit*, *EntityManagementUnit*, and *RelationshipManagementUnit*. In particular, the introduction of the OperationUnit `EClass` ensures that Areas and Siteviews from the HypertextOrganization package contain sub-classes of OperationUnit, only. Since the specific ContentManagementUnits are able to create, modify, and delete Entities as well as establish or delete Relationships between Entities from the content layer, the ContentManagement package reuses concepts from the Structure package, namely Entity and Relationship.

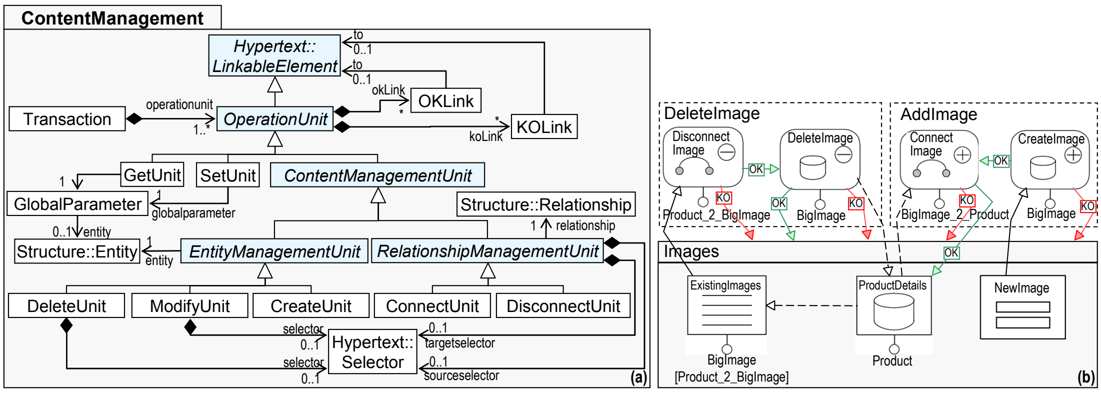


Figure 4.19: ContentManagement Package

Furthermore, redundant definitions of the same concept have been identified, namely Selector. As an example, a RelationshipManagementUnit may have two Selectors, with one being used in the role of a sourceselector and the other one being used as a targetselector. In the WebML DTD, this is expressed as follows (cf. Listing 4.7).

Listing 4.7: Roles of the Selector Concept

```
<!ELEMENT DisconnectUnit (Sourceselector?, Targetselector?,...)>
<!ELEMENT Selector(SelectorCondition+)>
<!ELEMENT Sourceselector (SelectorCondition+)>
<!ELEMENT Targetselector (SelectorCondition+)>
```

Since the *Targetselector* and *Sourceselector* `XMLElemType` declarations are identical to the *Selector* `XMLElemType` declaration, one can conclude that they represent the same concept as the *Selector* but are used in a special context. In contrast, in metamodels this context information can be defined as roles, i.e., incorporated by the `EReferences`' names. Therefore, the WebML metamodel only contains the Selector `EClass`, which is referenced by the RelationshipManagementUnit as a *sourceselector* and *targetselector*, respectively (cf. the `EReferences` role names in Figure 4.19*(a)*). A similar example can be found in the Hypertext package, where a Selector can act as *preselector* for MultiChoiceIndexUnits (cf. Figure 4.18*(a)*).

In the *Administrator* Siteview of the ACME web application, administrators can add, edit, and delete products, combinations, and stores. The *Images* Page in Figure 4.19*(b)* is part of the *ProductEditing* Area and allows adding and deleting images of a specific product. The Page displays product details in a DataUnit, an IndexUnit of existing images for the product, and an EntryUnit allowing the upload of further images. Selecting an image from the IndexUnit activates the Transaction *DeleteImage*, which has similar semantics as typical database transactions. First a DisconnectUnit disconnects the image and the products by deleting the specific instance of the relationship, then the OKLink is followed, the image is deleted using a DeleteUnit, and via a second OKLink the *Images* Page is reached again. In case of an error, the KOLinks are followed to the *Images* Page. The *AddImage* Transaction is activated when the user uploads a new image. It first creates a new image with a CreateUnit and then connects it to the specific product with a ConnectUnit.

4.4.6 AccessControl Package

In Figure 4.20*(a)*, the AccessControl package groups concepts for controlling the access to Siteviews, namely LoginUnit, LogoutUnit, and ChangeGroupUnit.

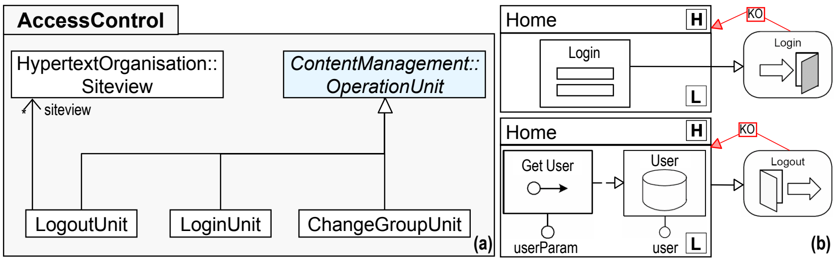


Figure 4.20: AccessControl Package

The example shows the *Web* Siteview, i.e., the *Home* Page of normal users (cf. Figure 4.20*(b)*). Administrators have to log in via the EntryUnit *Login*. The LoginUnit verifies username and password and switches to the user's default Siteview, i.e., the *Administrator* Siteview. In the *Home* Page of the *Administrator* Siteview, user information is displayed with the *User* DataUnit. The respective user is obtained from the session with a GetUnit (cf. Section 4.4.4). A user logs out via LogoutUnit, which forwards the user back to the *Web* Siteview for normal users

4.4.7 Basic Package

The Basic package consists of three abstract concepts, which encompass some features needed by the majority of WebML's modeling constructs. The ModelElement meta-class represents the root element of the WebML language from which all others inherit, either directly or indirectly. The

IdentifiedElement concept encompasses an `EAttribute` *id* as well as containment `EReferences` to the Comment and Property concepts of WebML. Finally NamedElement represents modeling concepts having a *name* `EAttribute`. Since almost consisting of abstract concepts only, no excerpt of the ACME modeling example will be provided for the Basic package.

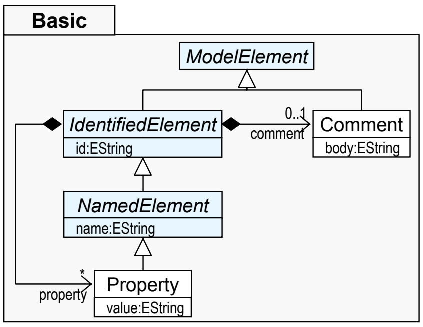


Figure 4.21: Basic Package

4.5 Discussion of the Generated WebML Metamodel

In the following, a discussion on the evaluation of the generated WebML metamodel is provided and shall give an indication on the applicability of the semi-automatic transformation approach. This evaluation is conducted, first, with respect to the metamodel's completeness compared to the language concepts defined in the original WebML DTD (cf. Section 4.5.1) and, second, on the basis of certain quality metrics (cf. Section 4.5.2).

4.5.1 Completeness Criteria

The completeness criteria is fulfilled at the meta-level M2 if the generated WebML metamodel contains all concepts defined within WebML's DTD and the WebRatio tool. At the model level M1 this means that the WebML models can be exchanged in a lossless way, i.e., instances of the WebML metamodel can be exchanged transformed to XML documents conforming to the WebML's DTD and vice versa.

Although WebML does provide a formal definition of the semantics of its concepts [CF01], [BCF02], a formal verification of the completeness criteria is not an option. This is due to the fact that currently within EMF the definition of semantics is not provided without executing the model itself. Nevertheless, a first prerequisite for completeness at the M2 level is provided by the fact that each WebML concept present in the DTD is dealt with by at least one transformation rule of the framework, which in turn assures for each WebML concept that there exists at least one counterpart in the metamodel.

In addition, completeness at the M2 level can be further underpinned by considering the M1 level. Taking a first step towards evaluating completeness at the M1 level, an "example-based" strategy has been followed, i.e., an existing WebML reference example has been remodeled on the basis of the generated metamodel. To do so, a tree-based modeling editor for the metamodel has been generated using EMF in order to completely remodel WebRatio's demo example, the

ACME E-Store. In addition, the example has been extended by those WebML language concepts not covered in the original example[7].

In a second step, the tree-based modeling editor was enhanced with an import/export facility to demonstrate whether models could be exchanged with the WebRatio tool in a lossless way[8]. With that facility it is possible to import the extended ACME example from WebRatio into the modeling editor and subsequently to export the model back into a WebRatio XML document. A comparison of the original XML document defined by WebRatio and the exported XML document from the modeling editor with the XML Differencing facility of StylusStudio[9] demonstrated that both were equivalent.

Admittedly, it has to be noted that this is only a first step towards justifying the semantic equivalence of the WebML metamodel and the original language specification not least since the evaluation shall comprise a larger set of more complex examples.

4.5.2 Quality Metrics

The WebML metamodel and its quality characteristics in terms of expressiveness, accuracy, and understandability have evolved considerably during the three-step transformation process. In order to illustrate this evolution, a set of metrics inspired by [MSZJ04] have been applied to the metamodel versions resulting from each step of the transformation process, i.e., the application of transformation rules, the employment of heuristics, and the manual validation and refactoring. The results of applying these metrics are summarized in Table 4.4.

Interpreting these metrics, one can observe that the introduction of a package structure, inheritance, and roles as well as the resolution of the awkward cardinalities deficiency has had a great impact on the understandability and readability of the metamodel. For the manual refactoring phase, the specific impact of introducing the `Basic` package shall be pointed out. The left-hand side of the manual refactoring phase column in Table 4.4 depicts the metrics of the refactoring actions without considering the introduction of the `Basic` package. The metrics on the right-hand side then depict the numbers for the final metamodel and illustrate the positive effect of the introduction of the `Basic` package. In particular, the introduction of inheritance through 17 abstract `EClasses` has helped to decrease complexity by reducing redundant `EAttributes` and `EReferences`. In this respect, the introduction of the three abstract `EClasses` from the `Basic` package played an important role. The identification of 3 roles has diminished the number of `EClasses`, while the resolution of awkward cardinalities has diminished the number of `EReferences`. All in all, the number of 707 modeling elements in the WebML DTD could be reduced to 487 modeling concepts in Ecore (i.e., counting `EClasses`, `EAttributes`, `EReferences`, and `EEnums`). The application of grouping mechanisms according to Heuristic 3 and further manual refactorings also had a positive effect on the language's readability in terms of an introduced package structure and a reduced ratio of `EClasses` per `EPackage`. After manual refactoring, the maximum number of `EClasses` per `EPackage` decreased from 53 to 26.

Concerning accuracy, the resolution of `IDREF(S)`-typed `XMLAttributes` into `EReferences`, the introduction of `EBoolean`-typed `EAttributes` instead of enumerations and the definition of constraints have considerably contributed to a more precise language. E.g., Heuristic 1 already correctly resolved 39, that is 41% `IDREF`-typed `XMLAttributes` into `EReferences` making the

[7] The modeling editor and the ACME example are available at http://big.tuwien.ac.at/projects/webml/

[8] Note that currently, the import/export facility supports the WebML content model only.

[9] www.stylusstudio.com

Metrics		Phase 1 Automatic Transformation		Phase 2 Manual Refactoring	
		Step 1	Step 2	Step 3	
All Modeling Concepts (EClass, EEnum, EAttribute, EReference)		707	707	580	487
EPackage		1	7	11	12
nested EPackage depth (Heuristic 3)		1	3	3	
EClass		96	96	104	107
abstract		0	0	14	17
inheriting from multiple EClasses		0	0	6	20
maximum inheritance depth		0	0	5	7
average inheritance depth		0	0	1.22115	1.66355
annotated with «Resolve XMLAnyET manually» (Heuristic 6)		-	3	-	
annotated with «Resolve Multiplicity manually» (Heuristic 4)		-	1	-	
EClasses/EPackage	MIN	96	1	1	
	MAX	96	53	26	
	AVG	96	13	9	8
EAttribute		338	338	241	191
EString		278	278	180	150
annotated with «Resolve IDREF manually» (Heuristic 1)		-	51	-	
annotated with «Resolve IDREFS manually» (Heuristic 1)		-	5	-	
annotated with «Resolve XOR manually » (Heuristic 5)		-	17	-	
EBoolean (Heuristic 2)		0	46	41	
EEnum		50	14	16	
EInteger		0	0	4	
EReference	annotated with «Validate IDREF» (Heuristic 1)	-	39	65	
	annotated with «Validate IDREFS» (Heuristic 1)	-	0		
	annotated with «Resolve XOR manually » (Heuristic 5)	-	5	-	
Containment EReference		234	234	159	113
EEnum		12	12	11	
annotated with «Resolve possible Boolean type manually » (Heuristic 2)		-	6	-	
OCL constraints	XOR constraint	5	5	10	
	other constraints	-	-	27	
Identified Roles		-	-	3	

Table 4.4: Metamodel Metrics

relationship between EClasses explicit. Further 56 EStrings had to be resolved manually into EReferences. From 50 enumeration-typed XMLAttributes, 36 could be resolved correctly by Heuristic 2 as EBooleans. Moreover, further 4 EEnums could be eliminated reducing their number to 8, the respective EAttributes could be refactored to EBooleans. Nevertheless, 3 more EEnums were introduced due to domain knowledge obtained from the WebRatio tool. This results in the final number of 11 EEnums, a reduction of EStrings in favor of an increase of EEnum attributes. And finally 32 additional constraints could be defined, thus, achieving a more precise WebML metamodel.

4.6 Introducing Customization into the WebML Metamodel

As already mentioned before, WebML's recently introduced concepts for modeling customization have not been part of WebML's DTD. This section describes how to manually incorporate them into the semi-automatically generated WebML metamodel presented in Section 4.4.

4.6.1 Designing Ubiquitous Web Applications with WebML in a Nutshell

Customization in WebML is based on context-aware Pages, Areas, and Siteviews [CDMF07]. These are tagged with a C-label (cf. Figure 4.22), i.e., a *ContextUnit*, which indicates that some adaptivity actions are associated with the Page, Area, and Siteview, respectively. These adaptivity actions must be evaluated prior to page computation, since they might cause the page content or the predefined navigation flow to be adapted to a certain context. Adaptivity actions defined for container elements such as Areas and Siteviews apply to all context-aware Pages within the container. Computed parameters then can be passed from containers to contained elements via ContextUnits. Furthermore, in the ContextUnit modelers can specify a time interval which triggers the adaptivity actions periodically after the first page access.

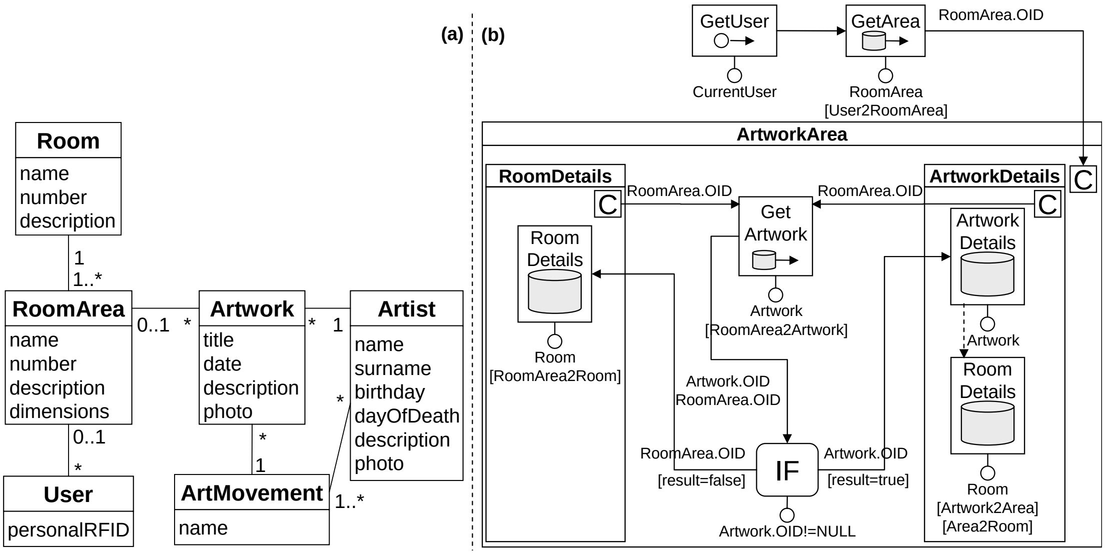


Figure 4.22: A Location-aware Museum Web Application

In general, the WebML extension for customization encompasses five new concepts, namely *ContextUnit*, *GetClientParUnit*, *GetDataUnit*, *ChangeSiteviewUnit*, and *ChangeStyleUnit* (cf. Table 4.5). In addition, two further concepts, i.e., *IfUnit* and *SwitchUnit*, that have already been introduced for supporting workflow-based web applications [BCC$^+$03] are reused for customization purposes. The new concepts' notation is provided in Table 4.5.

In Figure 4.22*(b)*, part of the hypertext model for a location-aware museum web application is shown. The corresponding content model is depicted in Figure 4.22*(a)*. It is assumed that the application relies on an RFID infrastructure and that sensing as well as storing of the users locations is done at server-side. In the example, the context-aware Area *ArtworkArea* computes the location, i.e., the room area, of the current user, which is to be passed to the contained context-aware Pages.

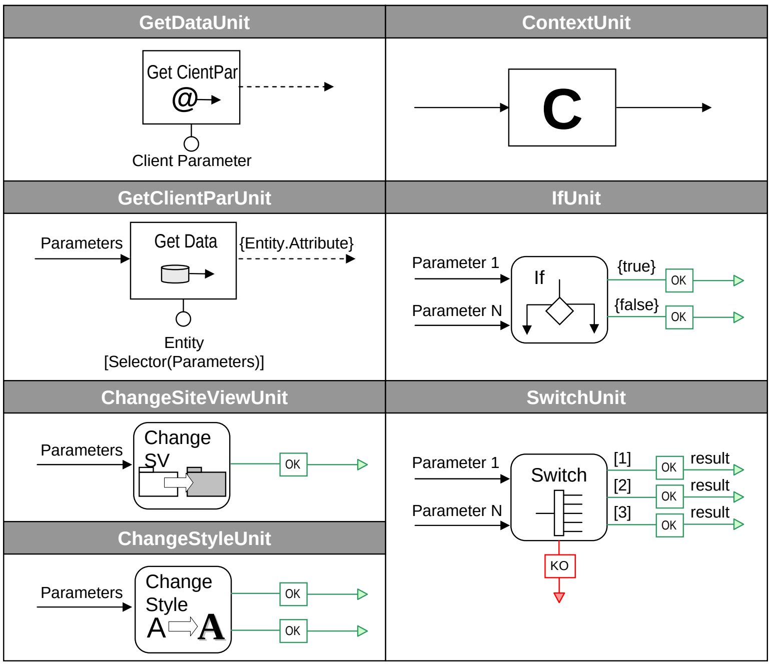


Table 4.5: WebML's Customization Concepts

In the example given in Figure 4.22*(b)*, the location of the user is obtained by the GetDataUnit *GetArea* and is passed to the context-aware Pages contained by the Area *ArtworkArea*. The GetDataUnit retrieves context data stored in the content model according to a SelectorCondition, which then can be passed on for further computations. If a user accesses the Page *ArtworkDetails*, his/her location, i.e., the room area, is used to get the artwork currently exhibited in that location with the GetDataUnit *GetArtwork*. In case no artwork can be found for the room area, the navigation flow will be adapted by an IfUnit and the user will be redirected to the Page *RoomDetails*. Still, if a user accesses the context-aware Page *RoomDetails* first, and the GetDataUnit *GetArtwork* retrieves an artwork for the area of the room the user is currently located, then the user will be redirected to the Page *ArtworkDetails*.

Among the new concepts not shown in the example, the GetClientParUnit retrieves context information sensed at the client side, e.g., longitude and latitude sensed by a GPS module attached to the client, the ChangeSiteviewUnit allows changing the currently displayed Siteview, and the ChangeStyleUnit represents an adaptation operation for adapting the presentation layer by changing the CSS style sheet. Finally, the IfUnit and SwitchUnit have the semantics of typical programming language conditional operators. For more details on those concepts the reader is referred to [CDMF07]. Moreover, the reader will find detailed information on how to model UWAs with WebML in the case study presented in Chapter 6.

4.6.2 The Final WebML Metamodel

The extension of the semi-automatically generated WebML metamodel with concepts for customization modeling has been straight-forward. It was possible to integrated all of the new concept into the WebML metamodel through inheritance in a constructive way. In the following, a discussion of the extensions made to the metamodel as well as the required refactoring actions is provided.[10]

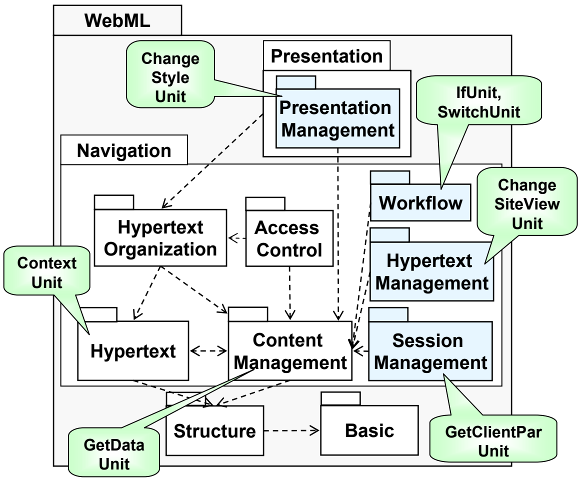


Figure 4.23: WebML Packages View

Figure 4.23 presents an overview of WebML's package structure after having introduced the new concepts, this time ignoring tool related packages. The ContextUnit and the GetDataUnit concepts have been integrated into existing packages. With respect to the rest, four additional packages, i.e., *SessionManagement*, *PresentationManagement*, *HypertextManagement*, and *Workflow*, have been introduced in order to better group concepts that semantically belong together.

The ContextUnit has been introduced into the Hypertext package as sub-class of the LinkableElement meta-class. As depicted in Figure 4.24, the HypertextOrganization package has been extended with appropriate containment `EReferences` in order to allow the ContextUnit to be part of Pages, Areas, and Siteviews.

In the ContentManagement package, the GetDataUnit has been specialized from the EntityManagementUnit in order to inherit the necessary `EReference` to the Entity meta-class from the Structure package. Additionally, a containment `EReference` is established to the Selector meta-class.

Incorporating the GetClientParUnit into the metamodel has been somewhat tricky. It has been put into a separate package together with similar already existing WebML concepts, e.g., GetUnit, SetUnit, GlobalParameter, and ParameterTypes (cf. Figure 4.25). More specifically, a new abstract concept *SessionManagementUnit* has been introduced, from which GetUnit, SetUnit, as well as the new GetClientParUnit inherit. The SessionManagementUnit itself has been derived from the LinkableElement meta-class of the Hypertext package. As a consequence, some refactor-

[10] Please not that the final WebML metamodel including concepts for customization modeling is also available at to *www.wit.at/people/schauerhuber/aspectUWA*.

ing actions where necessary in the HypertextOrganization as well as in the ContentManagement package in order to allow the SessionManagmentUnits to be contained by the correct container elements, i.e., Page, Area, and Siteview.

For both, the ChangeSiteviewUnit and the ChangeStyleUnit, two separate packages have been defined, namely HypertextManagement and PresentationManagement, respectively (cf. Figure 4.25). In both cases, an abstract meta-class has been introduced from which the new concepts shall inherit. More specifically, concerning the HypertextManagement package the *HypertextManagementUnit* has been designed to inherit from the OperationUnit concept from the ContentManagement package. Likewise the *PresentationManagementUnit* in the PresentationManagement package inherits from OperationUnit. This way, the WebML metamodel can be be easily extended with further adaptation operations, both for the hypertext and the presentation level. The ChangeSiteviewUnit then is specialized from the HypertextManagementUnit. It receives as an input parameter the target Siteview. Likewise, the ChangeStyleUnit is specialized from the PresentationManagementUnit and receives as input parameter the CSS style sheet to be used.

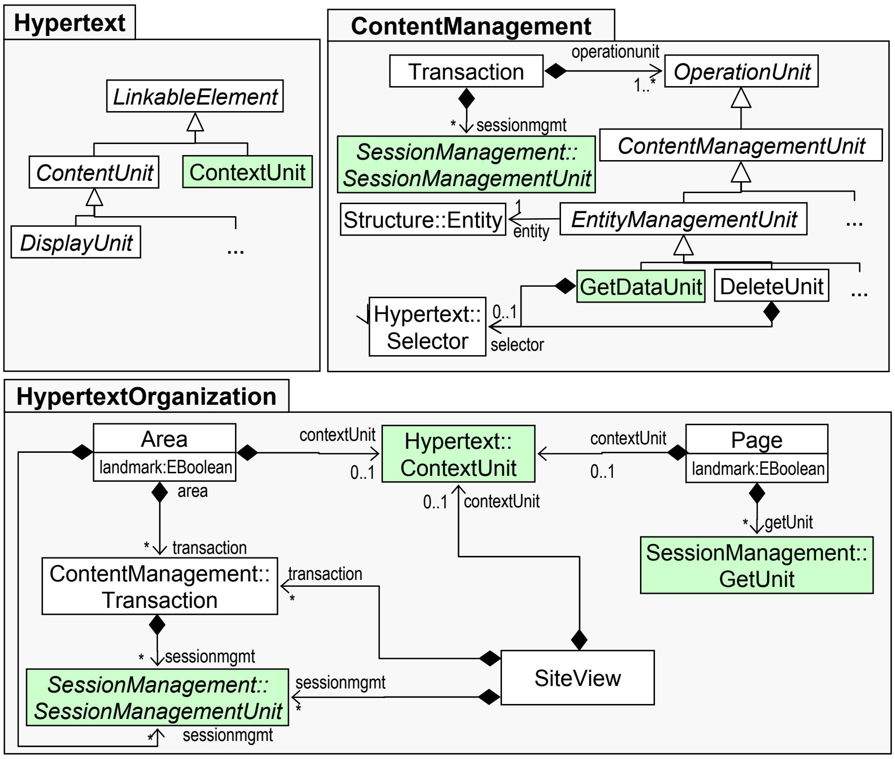


Figure 4.24: The Final WebML Metamodel (1)

Finally, the last two concepts from the domain of workflow-based web applications have been grouped within the new *Workflow* package (cf. Figure 4.25). The abstract concept *ConditionalUnit* inherits from the OperationUnit meta-class and provides the *Expression* to be evaluated to its sub-classes IfUnit and SwitchUnit. Depending on the ConditionalUnit there maybe several *Cases* to which the Expression can be resolved. Each case, however, has an `EReference` to the corresponding OKLink of the ConditionalUnit to be followed.

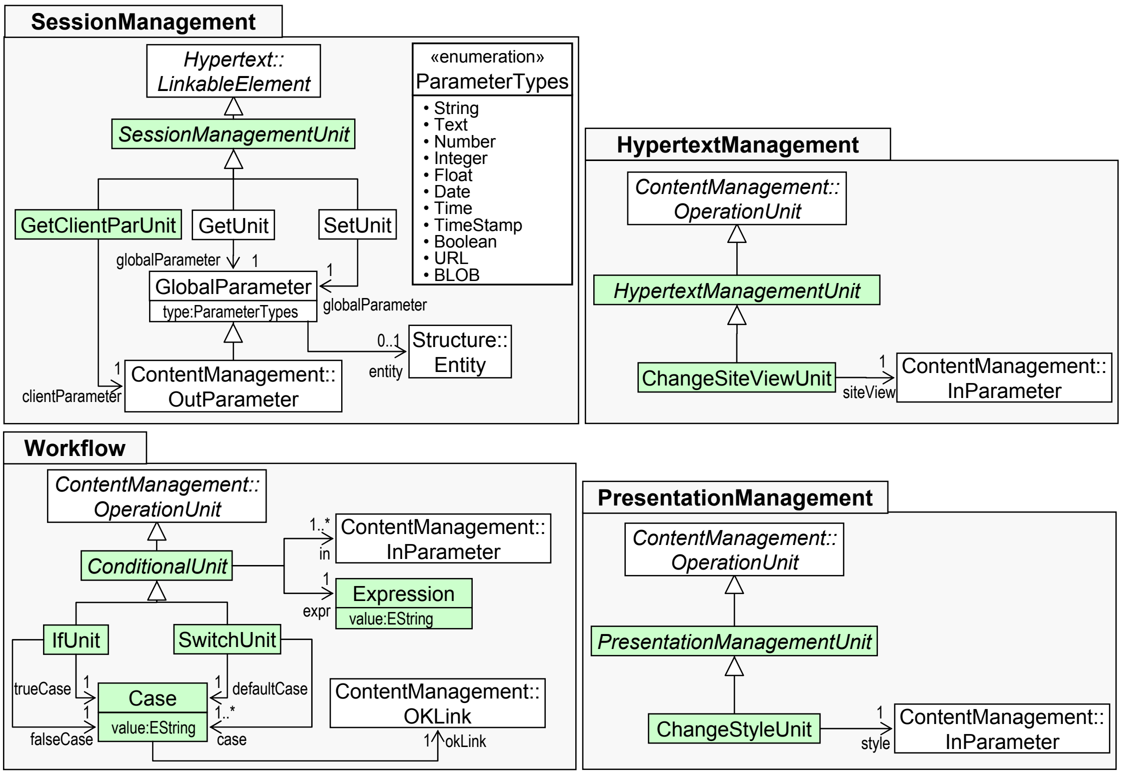


Figure 4.25: The Final WebML Metamodel (2)

4.7 Related Work

With respect to the presented approach of a semi-automatic generation of a MOF-based metamodel for WebML, two areas of related work can be distinguished: First, approaches aiming at the design of metamodels for web modeling languages, and second, approaches dealing with the transformation of DTDs to MOF-based meta-models.

4.7.1 Defining Meta-models for Web Modeling Languages

To the best of or knowledge, there is currently just one closely related approach focusing on the definition of a UML 2.0 Profile for WebML [MFV06]. The motivation of this approach is to facilitate the interoperability of the WebRatio tool with existing UML modeling tools. More specifically, WebML has been manually remodeled using MOF and in a second step a UML profile has been inferred from it. The approach followed in this thesis differs from the approach of Moreno et al. [MFV06] in three ways. First, we strive for a domain-specific language, for which today tool support can easily be provided, e.g., based on the EMF. Second, the presented WebML metamodel has been semi-automatically generated from WebML's DTD-based language specification. Third, since also tool related concepts have been considered in the transformation, this approach provides the prerequisite of migrating existing WebML models to MOF, while the WebML profile requires developers to re-model existing WebML models from scratch. And finally, WebML's new concepts for addressing context-awareness in web applications have also been incorporated.

Besides this closely related work, in the context of WebML, three other web modeling approaches which are currently defined on top of a metamodel need to be mentioned, namely W2000 [BCMM06], UWE [KK03], and Muller *et al.* [MSFB05]. W2000 [BCMM06], originally has been defined as an extension to UML. In [BM02], the provision of a metamodel based on MOF 1.4 [OMG02] has been motivated and adopted as a necessity for providing tool support for an evolving language definition. The metamodel of UWE [KK03] has been designed as a conservative extension to the UML 1.4 metamodel [OMG01], and thus is implicitly based on MOF 1.4. It is intended as a step towards a future common metamodel for the web application domain, which envisions supporting the concepts of all existing web modeling methods. Similar to W2000, a language definition already exists as UML Profile. Muller *et al.* [MSFB05] present a model-driven web application design and development approach through the Netsilon tool. The tool is based on a metamodel specified with MOF 1.4 and the Xion action language. The decision for a metamodel-based approach has been motivated by the fact that in the web application domain the semantic distance between existing modeling elements (e.g., UML) and newly defined modeling elements is becoming too large.

This work is complementary to W2000 and UWE in that a metamodel is proposed for another prominent web modeling language, i.e., WebML. Furthermore, in contrast to these approaches, the WebML metamodel presented in this thesis has been semi-automatically, instead of manually deriving it from an existing language definition. Finally, the resulting WebML metamodel is based on Ecore and thus, basically corresponds to MOF 2.0, while the meta-models of the other approaches are based on MOF 1.4.

4.7.2 Transforming between DTDs and Meta-Models

There have already been several approaches focusing on the transformation between XML and meta-models [WSKK06]. Summarizing these results, the approaches can be classified according to the direction of the transformation and the concrete formalisms used as source/target of the transformation. Considering the direction, one can distinguish between forward and backward approaches, regarding the used formalisms the approaches focus on the XML side either on DTDs or XML Schema and on the model side either on MOF, UML or ER, respectively. In the context of this approach, especially those approaches conducting a forward transformation from DTD to MOF are closely relevant. To the best of our knowledge, currently there is no such approach but there are two approaches [BCFK99] and [Sof00] transforming DTDs into UML models which are also closely related, not least since UML is based on MOF. There are, however, two differences with respect to the presented approach. First, a straight-forward transformation on basis of the correspondences between the two formalisms is extended by employing a set of heuristics dealing with potential ambiguous correspondences, thus facilitating a manual refactoring of the resulting metamodel. Second, the approach is based on a higher level of abstraction, meaning that the WebML DTDs are considered at the meta-level M2 whereas the other approaches relate domain DTDs to the model-level M1. Because of this higher level of abstraction, it is possible to transform WebML models in terms of XML documents conforming to the WebML DTD into instances of the corresponding WebML Ecore metamodel (representing in fact a so called ”linguistic instantiation” according to [AK03] and to validate if these models indeed fully conform to the WebML Ecore metamodel which is not facilitated by the other approaches. Note that, with respect to UML models, an XML document could in principle be mapped onto an object model, which represents an ”ontological instantiation” [AK03] at M1. However, the problem is that the object model must not fulfil the constraints given by the UML model and thus, the ”conforms to”-relationship between the XML document and the DTD is lost.

4.8 Summary

In this chapter, the prominent web modeling language WebML has been bridged to MDE for exploiting MDE benefits such as standardized storage, exchange, and transformation of models. To do so, the WebML language specifications partly available in the form of a DTD and partly hard-coded in WebML's modeling tool has been reused to generated a MOF-based WebML metamodel in terms of EMF's Ecore through a semi-automatic transformation process. As a consequence, this chapter's contributions are as follows:

First, when comparing a language specified in MOF to one specified on the basis of DTDs, it is obvious that DTDs considerably lack extensibility, readability, and understandability for humans, and above all expressiveness. In this respect, a set of eight deficiencies of DTDs when used as a language specification mechanism has been identified.

Second, having elaborated on the concepts of DTDs and MOF as well as their correspondences, a set of rules and heuristics for transforming arbitrary DTDs into MOF-based metamodels has been provided.

Third, a tool, i.e., the MetamodelGenerator for supporting a semi-automatic transformation process from DTD to MOF has been developed. As a consequence, the transformation approach enables the ”visual” representation of any DTD-based language in terms of MOF-based metamodels and thus, enhances the understandability of those languages.

Fourth, the resulting metamodel for WebML represents an important prerequisite and thus, an initial step towards a transition to employ model-driven engineering techniques (e.g., model transformations or language extensions through profiles) within the WebML approach. It also enables interoperability with other MDE tools and furthermore represents another step towards a a common reference metamodel for Web modeling languages.

The resulting WebML metamodel has been evaluated concerning its completeness and quality giving in particular indication on the applicability of the semi-automatic transformation approach. For evaluating completeness, an ”example-based” strategy has been followed in that a WebML reference example has been remodeled on the basis of a tree-based modeling editor supporting the generated metamodel. A prototype of an import/export facility of that editor was used to demonstrate that models could be exchanged with the WebML's modeling tool in a lossless way. For evaluating the quality of the metamodel, a set of quality metrics was applied to show the improvement of the metamodel during the semi-automatic transformation process. In the context of this thesis, it was also necessary to incorporate WebML's recently introduced concepts for modeling customization into the semi-automatically generated metamodel. These concepts have not been part of WebML's DTD and thus, had to be included manually according to the available literature.

5 aspectWebML - Applying aspectUWA to WebML

Contents

This chapter describes how the web modeling language WebML has been extended with modeling concepts from the aspect-orientation paradigm. *aspectWebML* is the result of bridging WebML to aspect-orientation and has been designed in order to better support the development of ubiquitous web applications (UWA). As input to the extension process, two previously presented artifacts have been used, i.e., the Conceptual Reference Model (CRM) for aspect-oriented modeling (AOM) presented in Chapter 3 and the WebML metamodel presented in Chapter 4. More specifically, the CRM has served as a blueprint for designing aspectWebML on top of the WebML metamodel, as is outlined in Section 5.1. The subsequent sections are dedicated to the presentation of the metamodel of aspectWebML focusing particularly on the aspect-oriented extensions made as well as on the discussion of how aspectWebML can be used to model and compose (crosscutting) concerns. In this respect, Section 5.2 explains how exactly the CRM has been applied, i.e., which of its concepts have been adopted and what further extensions have been necessary[1]. While the aspect-oriented concepts introduced in aspectWebML allow for separating (crosscutting) concerns such as customization from the rest of the web application model, a mechanism for integrating the previously decomposed parts is necessary. Only integration allows achieving a ″working″ model of the web application that can be fed to code generation facilities. Therefore, in Section 5.3, an explanation of how (crosscutting) concerns can be modeled with aspectWebML is given and the composition semantics of aspectWebML are explained on the basis of a set of examples. Furthermore, the section reports on the composition algorithm implemented to realize these semantics. Finally, a summary of the chapter is given in Section 5.4

5.1 On Using the Conceptual Reference Model for Bridging WebML to AOM

This section is dedicated to a description of how the web modeling language WebML has been bridged to AOM on the basis of the CRM. The primary goal in designing the CRM in Chapter 3 was to establish a common understanding of aspect-oriented concepts in the field of AOM. At the

[1] A previous version of the aspectWebML language has already been published in [SWS+07]

same time the CRM has been designed to provide a general framework allowing the extension of modeling languages with AOM concepts through a set of extension points.

In Chapter 3, the CRM has been defined in terms of a UML class diagram and identifies the basic ingredients of AOM, abstracted from specific modeling languages as well as from specific composition mechanisms. 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. These relationships to an arbitrary modeling language represent the aforementioned extension points.

Before providing an overview on how these extension points can be used, it has to be noted that the CRM has not been realized as is in the aspectWebML language for the following reasons:

- As already mentioned before, the CRM has been defined in terms of a UML class diagram. In contrast, for designing the WebML language and consequently the aspectWebML language, MDE technologies developed under the hood of the Eclipse Modeling Framework (EMF) have been used, i.e., the aspectWebML metamodel is based on the Ecore language. As a consequence, concepts such as association classes which are available in UML, only, have to be captured differently in the Ecore-based aspectWebML metamodel.

- Having already in mind modeling tool support based on the EMF, some further meta-classes have been introduced, including concepts for modeling repositories for ConcernModules and ConcernCompositionRules. In this respect, the aspectWebML metamodel has been extended with appropriate meta-classes.

- Some concepts of the CRM are not incorporated into aspectWebML, which is due to the WebML language's peculiarities. For example, the WebML language does not distinguish between structural and behavioral modeling elements. Consequently, in the resulting aspectWebML language a refinement of the meta-classes JoinPoint, ComposableElement, and SimpleAdvice in order to distinguish between structural and behavioral modeling elements has not been considered. In the aspectWebML metamodel, therefore, the meta-classes JoinPoint, ComposableElement, and SimpleAdvice adopt the role of an extension point instead of their sub-classes in the CRM.

Figure 5.1 outlines how the WebML language is bridged to AOM. The figure shows the aspectWebML package representing the aspectWebML metamodel. From the CRM, the packages ConcernComposition, AsymmetricConcernComposition, and SymmetricConcernComposition are reused, while the Language package is replaced by the WebML package representing the WebML metamodel. As is outlined in Figure 5.1, the extension of the WebML metamodel with AOM concepts of the CRM is achieved via the extension points JoinPoint, SimpleAdvice, ComposableElement, and ConcernModule. More specifically, the root element of the WebML language, i.e., the ModelElement meta-class, is specialized from the JoinPoint meta-class. In the aspectWebML language, this means any element of a WebML model can serve as a JoinPoint, or put in other words, the join point model of aspectWebML comprises all modeling concepts of the WebML language. In the same way, the ModelElement meta-class is specified as a sub-class of ComposableElement, allowing any modeling concept of WebML to participate in a symmetric composition. Furthermore, the SimpleAdvice meta-class is required to have ModelElements allowing any WebML modeling concept to be used in a SimpleAdvice. Finally, the WebML meta-class which represents a whole

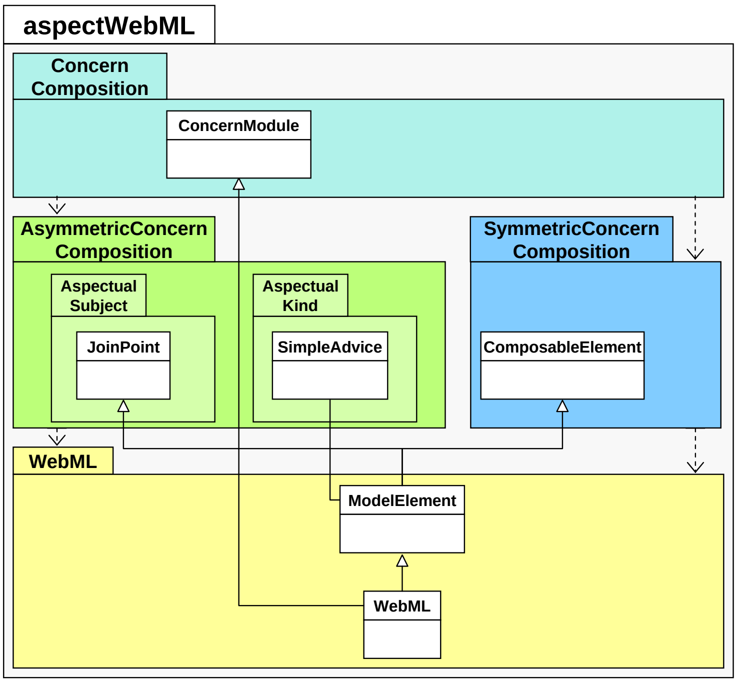


Figure 5.1: aspectWebML: An Overview

WebML model, needs to be specialized from the ConcernModule meta-class so that it can be composed in a CompositionPlan.

As already pointed out before, in Chapter 3 the CRM has been designed such that it abstracts from several composition mechanisms and at the same time specializes the aspect-oriented concepts in order to support the *pointcut-advice* and *open class* asymmetric composition mechanisms as well as the *compositor* symmetric composition mechanism. This particular design allows language designers to decide on implementing both kinds of composition mechanisms or only one of them while being able to easily complete the language with the other one later on. Due to the specific goal of separately modeling customization in UWAs, in this thesis, the focus will be on asymmetric composition mechanisms. In most web modeling languages, modeling customization is treated as a separate step to be performed on top of previous output artifacts in the development process, i.e., the content model, the hypertext model, and the presentation model. Moreover, customization functionality inherently is a crosscutting concern that needs to be defined on top of existing models. In this respect, it represents a concern that cannot exist on its own, meaning that it has to be applied to the functionality of a web application. As a consequence, asymmetric composition mechanisms which distinguish between core concerns and crosscutting concerns are more suitable to support customization modeling in the development of UWAs than symmetric composition mechanisms. Nevertheless, supporting both asymmetric as well as symmetric composition mechanisms allows for a powerful language. Therefore, the technical support for both kinds of composition mechanisms is given in terms of extending the WebML metamodel with aspect-oriented concepts for both mechanism. This allows for modeling (crosscutting) concerns in aspectWebML in either a symmetric or an asymmetric way. Considering the composition semantics of aspectWebML, the focus will be on a discussion of the semantics of asymmetric composition, which is due to the specific goal of this thesis. The discussion of the composition semantics of the symmetric composition mechanism supported in aspectWebML is out of scope and will be subject to future work as well as further research on how both kinds of composition mechanisms can be successfully used in parallel.

5.2 The aspectWebML Metamodel

In the following sub-sections, the aspectWebML metamodel will be described. This description follows a schema similar to the one used in the specification of the UML standard [OMG05d], thus, providing to modelers a well-structured reference. The aspectWebML metamodel will be presented along with its packages. For each package, a brief overview in terms of a class diagram depicting the package's meta-classes as well as a textual description is given. Thereafter, each meta-class is presented along with its *purpose*, possible *generalizations* and *specializations*, *attributes*, *references*, *constraints*, and *notation* in a tabular form. Please note, that the notation has been partly adopted and inspired by the AspectJ's IDE, i.e. the *AspectJ Development Tools*[2].

5.2.1 The ConcernComposition Package

The *ConcernComposition* package abstracts over different composition mechanisms.

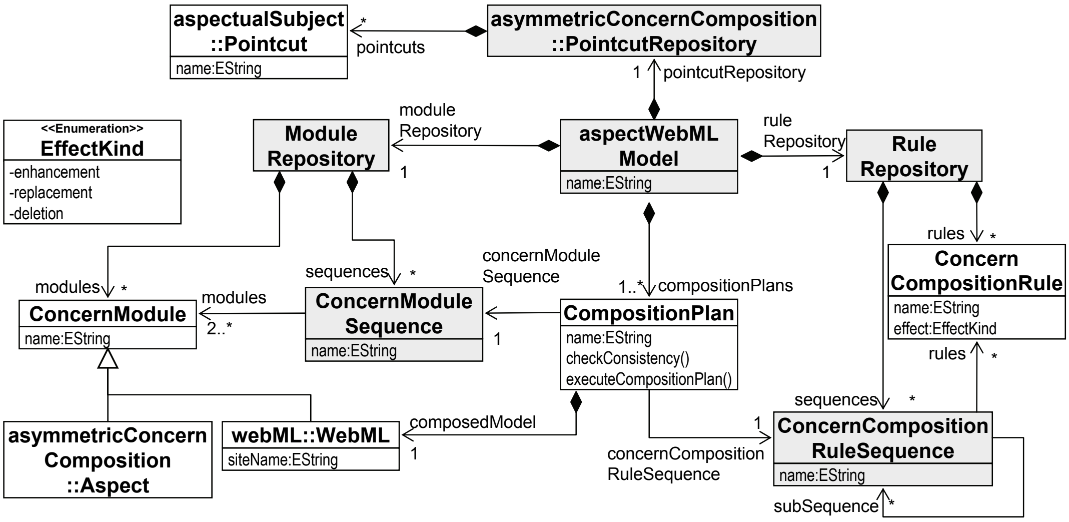


Figure 5.2: aspectWebML: The ConcernComposition Package

It has to be noted that the interaction issue, which is supported by the CRM through the ModuleInteraction and RuleInteraction concepts, is currently not (fully) considered within aspectWebML. This means, that modelers are not able to explicitly indicate interactions between ConcernModules, in their models. Indeed, the issue of *aspect dependencies and interactions* is considered an own sub-research field in the area of aspect-oriented software development. It will be

[2] www.eclipse.org/ajdt/

subject to future work, to find out what kinds of interactions (e.g., dependency, conflict, mutual exclusion) are relevant for the aspectWebML language and thus need to be supported. Nevertheless, within the implementation of the composition semantics in the aspectWebML tool support (cf. Chapter 7), dependencies between Advice as well as between Aspects can be computed and modelers are provided with appropriate warnings if necessary. Furthermore, some form of conflict resolution is provided through the introduction of the meta-classes ConcernModuleSequence (cf. Section 5.2.1.6) and ConcernCompositionRuleSequence (cf. Section 5.2.1.5), which allow for explicitly specifying the execution order of ConcernModules and ConcernCompositionRules, respectively. Having in mind modeling tool support based on the EMF, some further meta-classes have been introduced to the ConcernComposition package, including *aspectWebMLModel* as the the root modeling concept of the aspectWebML language as well as concepts for modeling repositories for ConcernModules and ConcernCompositionRules. An overview of the package is given in Figure 5.2, whereby the concepts specifically introduced for the aspectWebML language have been highlighted in grey.

5.2.1.1 aspectWebMLModel

The *aspectWebMLModel* meta-class has been introduced for tool support purposes. It represents the root modeling element in an aspectWebML project and encompasses a ModuleRepository, a RuleRepository, a PointcutRepository as well as a set of CompositionPlans.

Generalization. None.

Specialization. None.

Attributes.

name This attribute specifies the name of the aspectWebML project.

References.

moduleRepository The *moduleRepository* containment reference specifies the repository for storing the ConcernModules of the project, i.e., WebML models and Aspects, as well as ConcernModuleSequences to be used in a CompositionPlan.

ruleRepository The *ruleRepository* containment reference designates the repository for storing the ConcernCompositionRules of the project, i.e., AsymmetricCompositionRules and SymmetricCompositionRules, as well as ConcernCompositionRuleSequences to be used in a CompositionPlan.

pointcutRepository The *pointcutRepository* containment reference specifies the repository for storing the Pointcuts of the project to be used in AsymmetricCompositionRules.

compositionPlans The *compositionPlans* containment reference represents a set of CompositionPlans denoting different configurations for composing the project's ConcernModules.

Constraints. None.

Notation. There is no separate notational element necessary. Nevertheless, within the aspectWebML tool support (cf. Chapter 7) a special icon representing the aspectWebML language is used:

AW

5.2.1.2 ConcernModule

As is pointed out in Chapter 3, the *ConcernModule* concept encompasses a set of concern elements that together realize a concern or part of a concern. In the aspectWebML language, the *Concern* concept of the CRM has not been incorporated as a separate meta-class, since it is rather a theoretical concept than something modelers need to model explicitly within aspectWebML model. Furthermore, in Chapter 3, the ConcernModule concept has been defined to subsume the notion of aspect and base. In order to support asymmetric and symmetric composition mechanisms in the aspectWebML language, the ConcernModule concept needs to be specialized, however. Thus, the ConcernModule meta-class is declared to be abstract. The meta-class *Aspect* is specialized from ConcernModule and introduced to the AsymmetricConcernComposition package. Moreover, the *WebML* meta-class from the WebML language, which represents a WebML model inherits from ConcernModule as well. For the asymmetric composition mechanism a ConcernModule of type WebML then serves as the base.

Generalization. None.

Specialization. WebML::WebML, AsymmetricConcernComposition::Aspect.

Attributes.

name This attribute specifies the name of the ConcernModule.

References. None.

Constraints. None.

Notation. There is no separate notational element necessary, since the ConcernModule meta-class is declared abstract.

5.2.1.3 ConcernCompositionRule

A *ConcernCompositionRule* defines in detail how the various concern elements are to be composed. According to the CRM, the general concept of concern composition rule is specialized into subclasses supporting different composition mechanisms, i.e., the AsymmetricCompositionRule and the SymmetricCompositionRule.

Generalization. None.

Specialization. AsymmetricConcernComposition::AsymmetricCompositionRule, SymmetricConcernComposition::SymmetricCompositionRule.

Attributes.

name This attribute specifies the name of the ConcernCompositionRule.

effectKind The *effectKind* attribute declares the kind of the effect a rule has in the composed model, i.e., an enhancement effect, a replacement effect, or a deletion effect (cf. Enumeration EffectKind in Figure 5.2).

References. None.

Constraints. None.

Notation. There is no separate notational element necessary, since the ConcernCompositionRule meta-class is declared abstract. Instead, there are notational elements for its concrete sub-classes.

Please note, that the *Effect* concept from the CRM has been incorporated into the ConcernCompositionRule meta-class as an attribute instead of being modeled as a separate one. While the focus of the CRM has been to make explicit each aspect-oriented concept as a first-class citizen in the CRM class diagram, the goal in the aspectWebML language is to not impose too many new modeling elements on the modeler.

5.2.1.4 CompositionPlan

The *CompositionPlan* specifies how to integrate a set of ConcernModules according to a set of ConcernCompositionRules. The execution of a composition plan results in a composed model i.e., a WebML Model. As can be seen in Figure 5.2 the CompositionPlan meta-class has an appropriate operation *executeCompositionPlan()* which is responsible for executing the plan. Since the operational semantics of WebML are not defined in such a way that WebML models can be executed and simulated, static composition of concern modules is currently considered in the aspectWebML language, only. Therefore, the *isDynamic* attribute of the CompositionPlan concept in the CRM has not been adopted in aspectWebML.

Generalization. None.

Specialization. None.

Attributes.

name The *name* attribute declares the name of the CompositionPlan.

References.

concernModuleSequence The *concernModuleSequence* reference specifies the integration order of the ConcernModules to be used in the CompositionPlan by referencing an instance of ConcernModuleSequence.

concernCompositionRuleSequence The *concernCompositionRuleSequence* specifies the order of the ConcernCompositionRules by referencing an instance of ConcernCompositionRuleSequence.

composedModel The result of executing the CompositionPlan is stored in the *composedModel* containment reference. In the modeling environment (cf. Chapter 7), modelers shall be able to export the resulting WebML model to the ModuleRepository, where it can be reused in further CompositionPlans.

Operations.

executeCompositionPlan() The operation *executeCompositionPlan()* executes the CompositionPlan and results in the *composedModel*.

checkConsistency() The operation *checkConsistency()* is to be executed before *executeCompositionPlan*() and provides the modeler with warnings if the CompositionPlan cannot be executed due to errors in the aspectWebML project.

Constraints. None.

Notation. Again, there is no separate notational element necessary, since the CompositionPlan modeling element will not be part of a diagram. Nevertheless, within the aspectWebML tool support (cf. Chapter 7) a special icon representing a CompositionPlan is provided:
🔧

5.2.1.5 ConcernCompositionRuleSequence

The *ConcernCompositionRuleSequence* meta-class has been introduced to the aspectWebML language in order to specify reusable sequences of ConcernCompositionRules to be used in CompositionPlans. A ConcernCompositionRule defines the order of a set of ConcernCompositionRules. Moreover, a ConcernCompositionRuleSequence can be specified to have sub-sequences in order to further improve reuse of existing sequences.

Generalization. None.

Specialization. None.

Attributes.

name The name of the ConcernCompositionRuleSequence.

References.

rules The *rules* reference declares an ordered set of ConcernCompositionRules.

subSequence The *subSequence* reference specifies an ordered set of ConcernCompositionRuleSequences.

Constraints.

At least one of the references *rules* and *subSequence* has to be set:

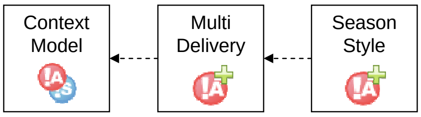
```
context ConcernCompositionRuleSequence
inv: self.rules->notEmpty()
       xor self.subSequence->notEmpty()
```

SymmetricCompositionRules contained in the ConcernCompositionRuleSequence need to specify via the *compElem* reference SymmetricConcernComposition::ComposableElements of type WebML::WebML:

```
context ConcernCompositionRuleSequence
inv: self.rules->
       select(e|e.oclIsTypeOf(SymmetricCompositionRule)).
       oclAsType(SymmetricCompositionRule)->
       forAll(e1|e1.compElem->forAll(e2|e2.oclIsTypeOf(WebML)))
```

Notation. For the ConcernCompositionRuleSequence an icon is used, which is composed of the icons for AsymmetricCompositionRules and SymmetricCompositionRules together depicting a sequence of rules:

Diagram. The *Rule Sequence Diagram* is used to graphically visualize the parts of a ConcernCompositionRuleSequence, i.e. its concrete ConcernCompositionRules and/or its sub-sequences:



In case the *rules* and *subSequence* references are both set, the ConcernCompositionRules from the *rules* reference (e.g. MultiDelivery and SeasonStyle) are to be considered after the *subSequence* reference (e.g. ContextModel) in a CompositionPlan.

5.2.1.6 ConcernModuleSequence

The purpose of the *ConcernModuleSequence* meta-class is similar to that of the ConcernCompositionRuleSequence. It has been introduced to the aspectWebML language in order to specify the order of ConcernModules in a CompositionPlan. During composition, the ConcernModules will be composed one after the other into a composed model.

Generalization. None.

Specialization. None.

Attributes.

name The *name* attribute specifies the name of the ConcernModuleSequence.

References.

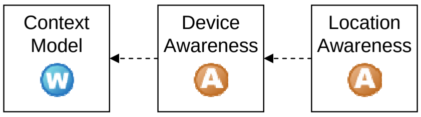
modules The *modules* reference declares an ordered set of ConcernModules.

Constraints. The first ConcernModule needs to be of type WebML::WebML:

```
context ConcernModuleSequence
inv: self.modules->first().oclIsTypeOf(WebML)
```

Notation. For the ConcernModuleSequence an icon is used, which is composed of the icons for AsymmetricConcernComposition::Aspects and WebML::WebML models together depicting a sequence of modules.

Diagram. The *Module Sequence Diagram* is used to graphically visualize the parts of a ConcernModuleSequence, i.e., its concrete modules in terms of WebML models and/or Aspects:



5.2.1.7 ModuleRepository

As already indicated above, some meta-classes have been introduced in order to allow for better modeling tool support. In this respect, the *ModuleRepository* meta-class has been included and subsumes all ConcernModules as well as ConcernModuleSequences defined in an aspectWebML project.

Generalization. None.

Specialization. None.

Attributes. None.

References.

modules This containment reference specifies the set of ConcernModules that can be used in the aspectWebML project.

sequences The *sequences* reference denotes the set of ConcernModuleSequences available in the aspectWebML project.

Constraints. None.

Notation. There is no need for a separate notational element, since the ModuleRepository meta-class is not used in the sense of a modeling element in aspectWebML. Within the aspectWebML tool support (cf. Chapter 7), a typical "folder" icon is used to denote the ModuleRepository:

5.2.1.8 RuleRepository

Analogous to the ModuleRepository meta-class, the *RuleRepository* meta-class which subsumes all ConcernCompositionRules as well as ConcernCompositionRuleSequences defined in an aspectWebML project, has been included.

Generalization. None.

Specialization. None.

Attributes. None.

References.

rules This containment reference specifies the set of ConcernCompositionRules that can be used in the aspectWebML project.

sequences The *sequences* reference declares the set of ConcernCompositionRuleSequences available in the aspectWebML project.

Constraints. None.

Notation. Analogous to the ModuleRepository meta-class, a typical ”folder” icon is used to denote the ModuleRepository within the aspectWebML tool support (cf. Chapter 7):

📁

5.2.2 The AsymmetricConcernComposition Package

In the *AsymmetricConcernComposition* package, the ConcernCompositionRule is specialized for covering asymmetric composition mechanisms. Analogous to the CRM, the package is organized into two sub-packages, namely *AspectualSubject* and *AspectualKind*. Besides the *Aspect* meta-class the *PointcutRepository* meta-class has been introduced. An overview of the package is given in Figure 5.3. Deviations from the CRM are highlighted in grey and are explained with the corresponding meta-class description.

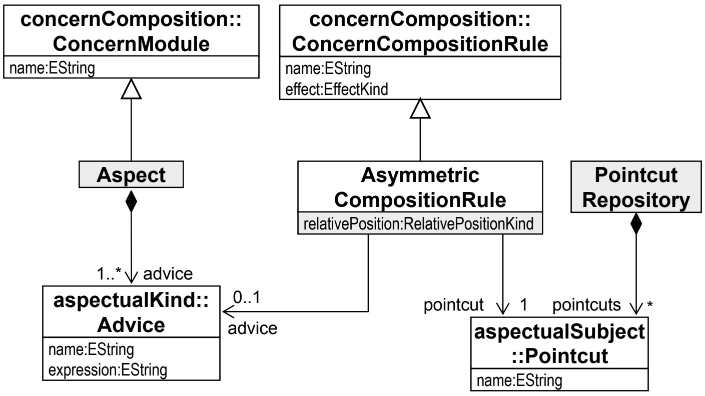


Figure 5.3: aspectWebML: The AsymmetricConcernComposition Package

5.2.2.1 AsymmetricCompositionRule

The *AsymmetricCompositionRule* is a specialization of the ConcernCompositionRule for asymmetric composition mechanisms. The AsymmetricCompositionRule meta-class has a reference to a Pointcut (cf. AspectualSubject::Pointcut) and an Advice (cf. AspectualKind::Advice). In contrast to the CRM, a separate meta-class for the CRM's RelativePosition concept is not provided. Again, the goal is not to impose on the modeler too many new modeling concepts. Therefore, the `relPos` attribute of type RelativePositionKind is directly incorporated into the AsymmetricCompositionRule meta-class.

Generalization. ConcernComposition::ConcernCompositionRule.

Specialization. None.

Attributes.

relPos This attribute denotes the relative position with respect to the JoinPoints where the Advice shall be introduced. Besides, *before*, *after*, and *around*, the *into* RelativePositionKind (cf. Enumeration RelativePositionKind in Figure 5.4) is supported.

References.

pointcut The *pointcut* reference declares the Pointcut where the Advice of the rule shall be applied.

advice The *advice* reference designates the Advice to be used in the rule.

Constraints.

In case the modeler specifies the rule to have deletion effect, the Advice must not be set:

```
context AsymmetricCompositionRule
inv: self.effect = EffectKind::deletion implies
     self.advice.oclIsUndefined()
```

In case the modeler specifies the rule to have an enhancement or replacement effect, the Advice is mandatory:

```
context AsymmetricCompositionRule
inv: self.effect <> EffectKind::deletion implies
     not self.advice.oclIsUndefined()
```

Notation. Depending on the EffectKind and the RelativePositionKind, different icons for the AsymmetricCompositionRule are available. The default case assumes an *enhancement* EffectKind and an *into* RelativePositionKind:

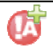
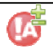
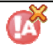
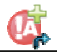
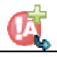
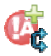
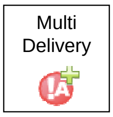
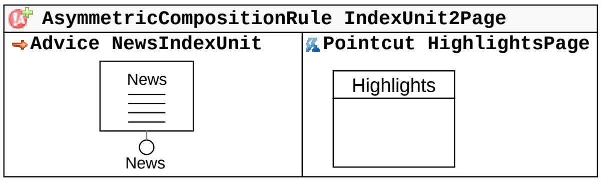
		EffectKind		
		Enhancement	Replacement	Deletion
RelativePositionKind	Into			
	Before		n/a	n/a
	After		n/a	n/a
	Around		n/a	n/a

Diagram. The notational element for AsymmetricCompositionRules to be used in the *Rule Sequence Diagram* is as follows:



Furthermore, in order to illustrate the details of AsymmetricCompositionRules, the *AsymmetricCompositionRule Diagram* depicts the rule's Pointcut and Advice in separate compartments and the rule's RelativePositionKind and EffectKind with one of the above icons.



5.2.2.2 Aspect

The *Aspect* meta-class has been introduced as a sub-class of ConcernModule in order to support asymmetric composition mechanisms. Aspects are used to encapsulate a set of Advice that together contribute to a certain concern.

Generalization. ConcernComposition::ConcernModule.

Specialization. None.

Attributes. None.

References.

advice This containment reference specifies the set of Advice that belongs to the Aspect.

Constraints. None.

Notation. The notational element for Aspect to be used in the *Module Sequence Diagram* is as follows:

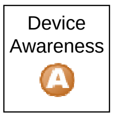
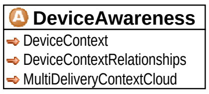


Diagram. In addition, Aspects are visualized within the so-called *Aspect Diagram*, which contains one or more Aspects together with their Advice.



5.2.2.3 PointcutRepository

In order to be able to reuse Pointcuts within an aspectWebML project, they are not defined as parts of Aspects but as parts of a general *PointcutRepository* in the project. This even allows some kinds of Pointcuts to be exported to repositories within other aspectWebML projects.

Generalization. None.

Specialization. None.

Attributes. None.

References.

pointcuts This containment reference specifies the set of Pointcuts that can be used in the aspectWebML project.

Constraints. None.

Notation. There is no need for a separate notational element, since the PointcutRepository meta-class is not used in the sense of a modeling element in aspectWebML. Again within the aspectWebML tool support the PointcutRepository is illustrated with a typical "folder" icon:
📁

5.2.3 The AspectualSubject Package

The *AspectualSubject* package encompasses the concepts required for identifying where to apply an Advice. As already explained before, during the application of the CRM to the WebML language, some of the package's concepts have not been incorporated into the aspectWebML language including StructuralJoinPoint and BehavioralJoinPoint. Further deviations from the CRM can be found in the following class descriptions. An overview on the AspectualSubject package is given in Figure 5.4, whereby deviations from the CRM are again highlighted in grey.

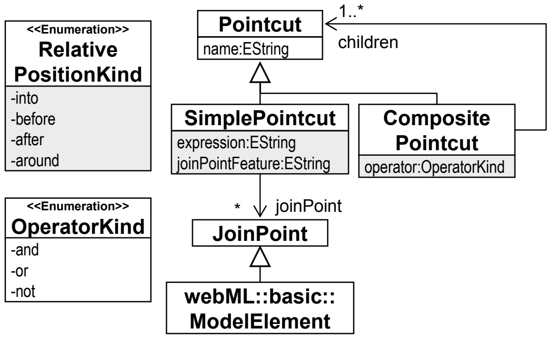


Figure 5.4: aspectWebML: The AspectualSubject Package

5.2.3.1 JoinPoint

In aspectWebML, a *JoinPoint* is a well-defined place in a ConcernModule, or rather an instance of a modeling concept belonging to the WebML language. This means, that any instance of a modeling element from the WebML language, whether it is defined in a WebML model or an Advice, can serve as a JoinPoint at some point in the CompositionPlan. In contrast, an Advice or any other aspect-oriented concept from the CRM cannot serve as JoinPoint. This is ensured in aspectWebML by letting the WebML's root modeling element (cf. WebML::Basic::ModelElement) inherit from the JoinPoint meta-class. Please note that the CRM's JoinPointModel concept has not been included into the aspectWebML language, since it is rather a theoretical concept than something to be modeled in UWAs. Furthermore, the *isDynamic* attribute of the CRM's JoinPoint concept is not incorporated into the aspectWebML language, since the operational semantics of WebML have not been specified and thus WebML models currently cannot be executed and simulated. Consequently, a distinction between dynamic JoinPoints and static JoinPoints is not required.

Generalization. None.
Specialization. WebML::Basic::ModelElement.
Attributes. None.
References. None.
Constraints. None.
Notation. Since the JoinPoint meta-class is defined to be abstract, there is no separate notational element required.

5.2.3.2 Pointcut

The *Pointcut* describes a set of JoinPoints selected for the purpose of introducing certain augmentations or constraints (cf. AspectualKind::Advice). The Pointcut meta-class is defined abstract and is specialized into the SimplePointcut meta-class and the CompositePointcut meta-class. It is important, that modelers define Pointcuts in a way such that they can be used together with an Advice in an AsymmetricCompositionRule. This means, that the Pointcut needs to specify a set of JoinPoints so that an Advice can be applied to all of them. The set of JoinPoints selected by the Pointcut has to be of the same type or there must be some model element type that can be introduced to all of them via an Advice. For example,it is not possible to define a Pointcut referencing instances of Entity and Page, since according to WebML's metamodel no Advice can be specified which is applicable to both of them.

Generalization. None.
Specialization. SimplePointcut, CompositePointcut.
Attributes.
name The name of the Pointcut.
References.
Constraints. None.
Notation. There is no separate notational element necessary, since the Pointcut meta-class is declared abstract. Instead, there are notational elements for its concrete sub-classes.

5.2.3.3 SimplePointcut

In the CRM, the *SimplePointcut* concept has been defined to represent a set of JoinPoints which are selected according to a certain quantification method (cf. Chapter 3). Please note, that the quantification method concept has been incorporated as a meta-attribute into the SimplePointcut meta-class instead of imposing to the modeler a separate QuantificationMethod meta-class as is suggested in the CRM. While the goal of the CRM has been explicitness by modeling all aspect-oriented concepts as first-class citizens, this is not necessary in the aspectWebML metamodel. Currently, the aspectWebML language allows for an enumeration-based way of explicitly selecting the JoinPoints (cf. *joinPoints* reference) and a declarative way of specifying JoinPoints by using OCL queries (cf. *expression* attribute). When modeling a SimplePointcut the modeler can choose one way or the other. While the enumeration-based way of specifying a SimplePointcut is straightforward and can be used in case the modeler is not acquainted with OCL, the use of OCL-based SimplePointcuts has two advantages: First, it allows for reusing Pointcuts within other *aspectWebML* projects. And second, pointcuts are more robust to changes of the web application model. This means that new model elements that match the query are automatically selected by the Pointcut. In an enumeration-based Pointcut, each new model element would have to be specified explicitly. Following, a description is given on how enumeration-based as well as OCL-based Pointcuts can be specified.

Defining Enumeration-based SimplePointcuts. In order to specify SimplePointcuts as an enumeration, the modeler simply defines the references to existing modeling elements in the *joinPoints* reference. In some of these cases, however, the modeler is required to provide more information on where an Advice needs to be applied. This information can be provided within the *joinPointFeature* attribute, where the modeler specifies the name of the JoinPoint's meta-attribute or meta-reference. For example, a modeler might wish to change the *name* attribute of an Entity in the content model. Let's say the Entity *User* shall be renamed to 'Person'. In this case, the modeler needs to specify the *User* Entity instance in the *joinPoints* reference of the Pointcut. In addition, the name of the meta-class feature, i.e., the *Entity.name* attribute, needs to be defined in the *SimplePointcut.joinPointFeature* attribute. Likewise, a modeler might want to set the *Entity.superentity* reference of the *User* Entity to point to some other entity (cf. Section 4.4.2).
Another reason for specifying a JoinPoint's feature in *SimplePointcut.joinPointFeature*, is the possibility of a meta-class having more than one (containment) reference to one and the same type in the metamodel. Imagine the content model would capture persistent and transient entities in separate references named *persistentEntity* and *transientEntity*. In this case, modelers need to explicitly specify the intended reference, since during composition a default can be chosen, only.

Defining OCL-based SimplePointcuts. Allowing for OCL-based SimplePointcuts, modelers are able to define a repository of reusable Pointcuts which can be imported into other aspectWebML projects. Two levels of reusability can be distinguished: *First-level Pointcuts* are defined independently from a WebML model meaning they are defined according to properties from the WebML language. As an example consider a Pointcut selecting all home pages of a web application. This SimplePointcut is specified on the basis of metamodel information, only. Consequently, it is reusable for all aspectWebML projects:

```
context WebML
inv:  self.webModel.siteview.homepage
```

Second-level Pointcuts are defined on the basis of information from a WebML model and can be reused in another aspectWebML project if its WebML model possesses the same information. As an example consider a SimplePointcut selecting an Entity with name *User*:

```
context WebML
inv:  self.dataModel.entity->select(e|e.name='User')
```

When compared to an enumeration-based SimplePointcut, which specifies the *User* Entity as a reference within *SimplePointcut.joinPoints*, the advantage of the OCL-based SimplePointcut is that it is reusable in other aspectWebML projects, given that the content model also has an entity named 'User'. The actual JoinPoint of the SimplePointcut is not computed until executing the CompositionPlan.
Unfortunately, the modeler is required to have a good understanding of the WebML language in order to define OCL-based Pointcuts. Still, this problem can be sufficiently solved with appropriate tool support, e.g., in terms of an OCL console including code completion functionality on the basis of the WebML metamodel (cf. Section 7.1.2). Unlike the above examples, please note that in the aspectWebML tool support, the OCL queries need to be specified starting navigation from the

the *aspectWebMLModel* element.

Generalization. Pointcut.

Specialization. None.

Attributes.

expression The *expression* attribute represents the modeler's OCL query used to determine the set of JoinPoints.

joinPointFeature The *joinPointFeature* attribute specifies the name of the structural feature of the selected JoinPoints.

References.

joinPoints The *joinPoints* reference specifies the enumeration-based selection of JoinPoints.

Constraints. Either the expression attribute or the joinPoints reference (possibly in combination with the joinPointFeature attribute) is set:

```
context SimplePointcut
inv1: not (self.expression.oclIsUndefined() or self.expression='') xor
      joinPoint->notEmpty()
inv2: not (self.expression.oclIsUndefined() or self.expression='')
      implies joinPointFeature.oclIsUndefined()
inv3: not (self.joinPointFeature.oclIsUndefined() or
      self.joinPointFeature='') implies self.joinPoint->notEmpty()
```

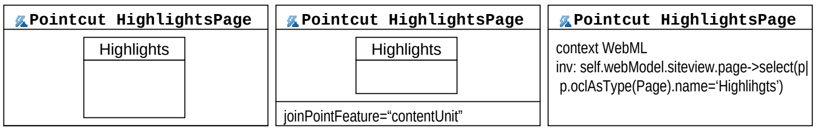
If *joinPointFeature* is set, then all JoinPoints specified in *joinPoint* need to have the feature specified in *joinPointFeature*:

```
context SimplePointcut
inv: if (not self.joinPointFeature.oclIsUndefined() or
          not self.joinPointFeature='')
      then self.joinPoint->forAll(j|j.eStructuralFeatures->exists(f1|
        f1.oclAsType(EReference).name = self.joinPointFeature)) or
        self.joinPoint->forAll(j|j.eStructuralFeatures->exists(f2|
        f2.oclAsType(EAttribute).name = self.joinPointFeature))
      else true
      endif
```

Please note that this invariant can actually not be realized in OCL, since OCL does not allow to reflexively work on the meta-level, e.g., using the *eStructuralFeatures* reference from the Ecore language is not possible. Such a constraint consequently needs to be realized within the tool support.

Notation. SimplePointcuts are depicted with the following icon:

Diagram. Furthermore, they can be visualized in the *Pointcut Diagram*. Depending on how the SimplePointcut is defined, three variants for the Pointcut Diagram can be distinguished.


Pointcut HighlightsPage
Highlights
Pointcut HighlightsPage
Highlights
joinPointFeature="contentUnit"
Pointcut HighlightsPage
context WebML
inv: self.webModel.siteview.page->select(p|
p.oclAsType(Page).name='Highlihgts')

5.2.3.4 CompositePointcut

For reuse purposes, modelers are allowed to define *CompositePointcuts* which are composed of other SimplePointcuts and/or CompositePointcuts by means of logical operators, e.g., AND, OR, NOT. Since Ecore does not support association classes, the `operator` used is now defined as an attribute within the CompositePointcut meta-class instead of the CRM's *Composition* association class.

Generalization. Pointcut.

Specialization. None.

Attributes.

operator This attribute specifies the kind of the operator used to compose the CompositePointcuts children, i.e., AND, OR, and NOT (cf. Enumeration OperatorKind in Figure 5.4).

References.

children The *children* reference specifies the Pointcuts, i.e., SimplePointcuts and CompositePointcuts, to make up the CompositePointcut.

Constraints. A CompositePointcut cannot be composed of itself:

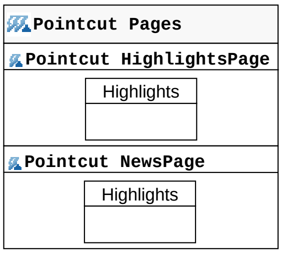
```
context CompositePointcut
inv: not self.getAllSimplePointcuts()->includes(self)
```

The *getAllSimplePointcuts()* operation is defined in order to be able to reflexively compute all SimplePointcuts of a CompositePointcut. The operation is used for the previous constraint:

```
context CompositePointcut def:
getAllSimplePointcuts():Bag(SimplePointcut)=
 self.children->iterate(
  pc:Pointcut;
  allSimplePCs:Bag(SimplePointcut) = Bag{} |
  if pc.oclIsTypeOf(SimplePointcut)
    then allSimplePCs.including(pc)
    else allSimplePCs.union(pc.getAllSimplePointcut())
  endif)
```

Notation. Like SimplePointcuts, CompositePointcuts are supported with an icon:

Diagram. The Pointcut Diagram visualizes CompositePointcuts in separate compartments, whereby the contents of the compartments can be hidden.



5.2.4 The AspectualKind Package

The *AspectualKind* package comprises the concepts necessary to describe how to augment or constrain other ConcernModules. In contrast to the CRM, in aspectWebML, the *SimpleAdvice* is not further specialized into *StructuralAdvice* and *BehavioralAdvice*, since such a distinction is not supported by the underlying WebML language (cf. Figure 5.5). Again, further deviations from the CRM are explained in the following meta-class descriptions.

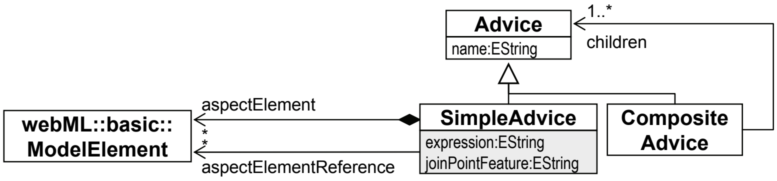


Figure 5.5: aspectWebML: The AspectualKind Package

5.2.4.1 Advice

The *Advice* specifies how to augment or constrain other ConcernModules at JoinPoints matched by a Pointcut. Similar to the Pointcut of the AspectualSubject package, the Advice is specialized into SimpleAdvice and CompositeAdvice.

Generalization.	None.
Specialization.	SimpleAdvice, CompositeAdvice.
Attributes.	
name	The name of the Advice.
References.	None.
Constraints.	None.
Notation.	The Advice meta-class is declared as abstract. Notational elements are available for its concrete sub-classes, only.

5.2.4.2 SimpleAdvice

Depending on what the modeler wishes to do, there are three different ways of specifying a *SimpleAdvice*, namely to ”change” modeling elements, their attributes, or their references:

- In case, the web application model shall be extended with a set of modeling elements, the modeler will need to specify them with the *aspectElement* containment reference of the SimpleAdvice. These modeling elements together have to be modeled such that they can be introduced to the same types of JoinPoints. For example, the SimpleAdvice may contain an Entity *Product* and a Domain *AgeClass* (cf. Section 4.4.2). According to the WebML metamodel, both have the same container, i.e., the content model. In the *Diagram* section below the graphical representation of this kind of SimpleAdvice is indicated with the *NewsIndexUnit* SimpleAdvice containing an IndexUnit.
- If the modeler wishes to change a modeling element’s property, the *expression* attribute of the SimpleAdvice meta-class needs to be used. For example, the name of the Entity *User*

shall be changed into 'Person'. The *expression* attribute of the SimpleAdvice meta-class allows modelers to specify the new value of the property to be changed in terms of a String. For composition purposes, it has to be noted, that in the *Pointcut.joinPointFeature* attribute, additional information is needed in order to select the right property of the modeling element. Furthermore, in case the meta-attribute specified in the *Pointcut.joinPointFeature* is of type Integer, this has to be considered during the execution of the CompositionPlan as well, meaning that the value of *SimpleAdvice.expression* needs to be casted accordingly. In the *Diagram* section below the graphical representation of this kind of SimpleAdvice is indicated with the *NameValue* SimpleAdvice.

- Finally, it might be necessary to change a modeling element's references. For example, the super-entity of an entity is specified in the WebML metamodel with a reflexive reference of the Entity meta-class. If set, the *Entity.superentity* reference points to an entity's super-entity. In order to set the reference to another entity instance, this instance needs to be captured in the *aspectWebMLReference* reference. In the *Diagram* section below the graphical representation of this kind of Advice is indicated with the *NewsIndexUnit2* Advice, referencing an existing IndexUnit (highlighted in grey).

Generalization. Advice.
Specialization. None.
Attributes.
expression This attribute represents the value used to set a modeling element's attribute.

References.
aspectElement This containment reference specifies the set of WebML modeling elements that make up the Advice.
aspectElementReference The *aspectElementReference* reference captures the references to a set of existing WebML modeling elements, which is used to (re)set the references selected by a Pointcut.

Constraints. Either *aspectElement*, *aspectElementReference*, or *expression* is needs to be set.

```
context SimpleAdvice
inv: self.aspectElement->notEmpty() xor
     self.aspectElementReference->notEmpty() xor
     not (self.expression.oclIsUndefined() or self.expression='')
```

Notation. A SimpleAdvice is represented with the following icon:
➪

A SimpleAdvice can be visualized within the so-called *Advice Diagram*. Depending on how the SimpleAdvice has been defined, three variants of the Advice Diagram are distinguished:

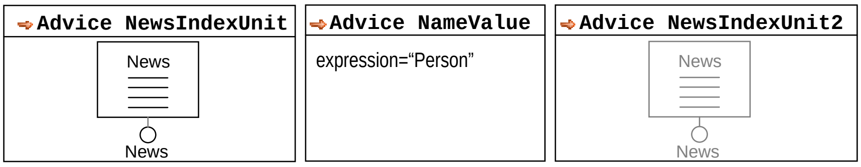
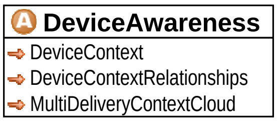


Diagram. In addition, a SimpleAdvice can be visualized within the so-called *Aspect Diagram*, which contains one or more Aspects together with their Advice:



5.2.4.3 CompositeAdvice

For reuse purposes, an Advice can be composed of a coherent set of SimpleAdvice, to form a *CompositeAdvice*. In aspectWebML a CompositeAdvice will be interpreted as the sum of the modeling elements specified by its children. Consequently, a coherent set of SimpleAdvice means, that together they have to be applicable to one Pointcut. For example, it is not possible to compose a SimpleAdvice specifying Entities with another one specifying Pages.

Generalization. Advice.

Specialization. None.

Attributes. None.

References.

children The *children* reference specifies the Advice, i.e., SimpleAdvice and CompositeAdvice, to make up the CompositeAdvice.

Constraints. All children have either *aspectElement* or *aspectElementReference* set. SimpleAdvice with the *expression* attribute set are not allowed to be children of a CompositeAdvice:

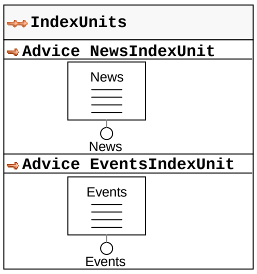
```
context CompositeAdvice
inv: not self.getAllChildren()->forall(e|e.aspectElement->notEmpty())
or forall(e|e.aspectElementReference->notEmpty())

context CompositeAdvice
inv: self.getAllChildren -> forAll(e| e.expression.oclIsUndefined
or e.expression = '')
```

The *getAllSimpleAdvice()* operation is defined in order to be able to reflexively compute all SimpleAdvice of a CompositeAdvice. The operation is used for the previous constraints:

```
context CompositeAdvice def: getAllSimpleAdvice():Set(SimpleAdvice)=
 self.children->iterate(
  a:Advice;
  allSimpleAdvice:Set(SimpleAdvice) = Set{} |
  if pc.oclIsTypeOf(SimpleAdvice)
    then allSimpleAdvice.including(a)
    else allSimpleAdvice.union(a.getAllSimpleAdvice())
  endif)
```

Notation. A CompositeAdvice is represented with the following icon:

Diagram. Similar to the Pointcut Diagram, a CompositeAdvice is visualized within the *Advice Diagram* using a compartment for each of the CompositeAdvice's children:



Again, a CompositeAdvice is also visualized within the so-called *Aspect Diagram* using its special icon.

5.2.5 The SymmetricConcernComposition Package

As already explained before, the symmetric concern composition is not the focus of this thesis. Nevertheless, full support of symmetric concern composition is planned for aspectWebML in the future. Consequently, in this thesis, the concepts specified in the CRM to support symmetric composition mechanisms are incorporated into the aspectWebML metamodel as far as it is necessary to guarantee an easy extension with symmetric composition semantics at a later date. This means, symmetric concern composition is also considered within the composition algorithm but still needs to be implemented. Following, a brief overview on the symmetric concern composition's current support within aspectWebML will be provided.

In the *SymmetricConcernComposition* package, the concern composition rule is specialized according to the compositor composition mechanism. The concepts specified in the CRM have been included as is into the aspectWebML language with two exceptions (cf. Figure 5.6): A distinction of the *ComposableElement* into *ComposableStructuralElement* and *ComposableBehavioralElement* is not made. For the asymmetric concern composition mechanism, the ModelElement meta-class of WebML (cf. WebML::Basic::ModelElement) has been specialized from JoinPoint, thus allowing all of WebML's modeling elements to become the subject of an Advice. Likewise, in the symmetric composition mechanism all of WebML's modeling elements are composable by specifying an inheritance relationship between ModelElement and ComposableElement.

For the ConcernCompositionRuleSequence, only those rules are allowed that reference ComposableElements of type WebML (cf. the previously specified OCL constraints). If more SymmetricCompositionRules are necessary to specify the composition of two WebML models, they need to be defined as sub-rules of this SymmetricCompositionRule. As an example for an extra rule, imagine the content models of the two WebML models need to be merged but the Entity *User* in one content model needs to be overridden by the Entity *Person* of the other content model. Consequently, in the aspectWebML metamodel, the SymmetricCompositionRule meta-class is specified to have *subRules*.

It is obvious, that the aspectWebML metamodel will need to be refined to fully support the symmetric composition mechanism. We intend to base this refinement on the work of *Clarke's* Theme/UML approach [Cla02]. Furthermore, we are currently also investigating the suitability of the compositor composition mechanism in the case of the WebML language. Compared to Clark's work which is based on UML class diagrams, in aspectWebML the compositor mechanism is currently applied to the whole WebML language, i.e., the content model and the hypertext model, which is richer in terms of modeling concepts. Thus, merging two WebML models can involve a lot of rules. Again appropriate means for visualization and tool support have to be designed as it

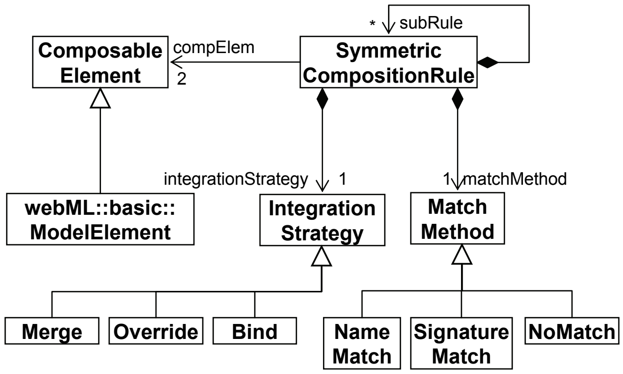


Figure 5.6: aspectWebML: The SymmetricConcernComposition Package

is done for the asymmetric composition mechanisms in this thesis (cf. Chapter 5 to 7). In future work, the compositor composition mechanism for aspectWebML shall be fully realized as well.

5.3 Modeling and Composing Crosscutting Concerns with aspectWebML

"Having divided to conquer, we must reunite to rule" [Jac90]. Or put in other words, decomposition requires corresponding composition support. In this section, the focus is on the (asymmetric) composition semantics of the aspectWebML language, which heavily rely on the WebML language, i.e., its metamodel. The section will start with an introduction into how aspectWebML is to be used considering WebML's peculiarities in Section 5.3.1. More specifically, a simple example is provided in order to illustrate, how Aspects can be defined as well as how they can be composed within a CompositionPlan. Thereby, an overview on the main parts of the composition algorithm - as it has been implemented - is given. What follows next is a detailed, examples-based discussion of aspectWebML's composition semantics in Section 5.3.2 in terms of an explanation of how the composition algorithm deals with different kinds of AsymmetricCompositionRules.

The last part of this section is dedicated to a discussion on the limitations of the aspectWebML language with respect to supporting the asymmetric pointcut-advice composition mechanism (cf. Section 5.3.3).

5.3.1 An aspectWebML Primer

5.3.1.1 Modeling Concerns with aspectWebML

In the following example, an arbitrary web application is extended in order to provide the user with information on news and events. Therefore, the content model and the hypertext model shall be extended with appropriate Entities and Pages, respectively.

The web application is represented by the *ArbitraryWebApplication* WebML model (cf. Figure 5.7*(a)*) and the required extensions will be captured in the *NewsAndEvents* Aspect. The *NewsAndEvents* Aspect encapsulates all Advice needed to extend the *ArbitraryWebApplication* WebML model as is illustrated in the *Aspect Diagram* of Figure 5.7*(b)*. In order to extend the content model and the hypertext model with appropriate Entities and Pages, two Advice need to be specified. First, a *News* Entity and an *Event* Entity are modeled in the Advice *NewEntities*. The corresponding *Advice Diagram* is shown in Figure 5.7*(c)*. Users of the web application shall be provided with two further pages which are accessible from anywhere in the web application. One will present a list of news and the other will provide a list of events. In the Advice *NewLandmarkPages*, this is realized with two landmark Pages, each having an IndexUnit to display instances of either the *News* Entity or the *Events* Entity (cf. Figure 5.7*(c)*).

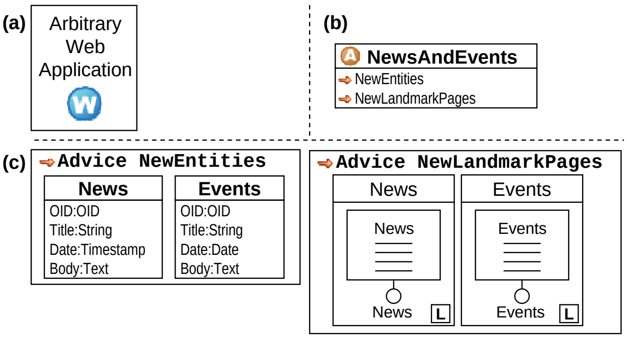


Figure 5.7: The *NewsAndEvents* Aspect

Having specified all necessary Advice, the next thing to do is to define the Pointcuts where the Advice shall be applied in the *ArbitraryWebApplication* WebML model and then to combine the Advice with an appropriate Pointcut in an AsymmetricCompositionRule. As will be explained in detail in Section 5.3.2, an AsymmetricCompositionRule is a combination of an Advice a Pointcut as well as a RelativePositionKind and an EffectKind. As a result there will be alternatives for specifying the same thing with different configurations of an AsymmetricCompositionRule.

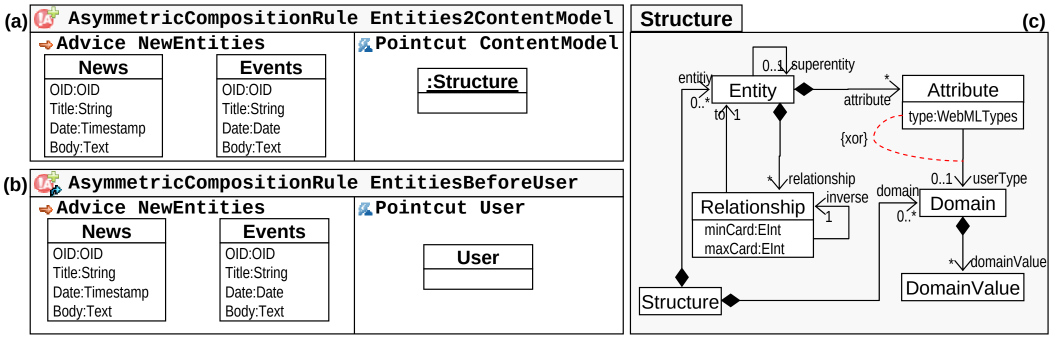


Figure 5.8: The *Open Class* Composition Mechanism in aspectWebML

In the following, two different ways of how the *NewEntities* Advice can be applied to the *ArbitraryWebApplication* WebML model will be discussed. These two examples shall illustrated that the WebML language mainly focuses on the structural features of a web application. Thus, with respect to using asymmetric composition mechanisms, in aspectWebML, modelers will actually use the open class composition mechanism in order to compose Aspects with a WebML model (cf.

Chapter 3). A discussion on using the pointcut-advice composition mechanism in aspectWebML is provided in Section 5.3.3.

1. The first way of extending the *ArbitraryWebApplication* WebML model with the new Entities is illustrated in Figure 5.8*(a)* with the AsymmetricCompositionRule *Entities2ContentModel*. The rule has an *enhancement* EffectKind on the WebML model as is denoted by the icon of the rule. This means, that the modeling elements specified in the Advice *NewEntities* are added to the JoinPoints defined by the Pointcut *ContentModel*. Per default, the RelativePositionKind is *into*, meaning that the elements of the Advice are added into the container specified in the Pointcut. More specifically, the Entities are inserted into the *entity* containment reference of the WebML meta-class *Structure* (cf. the corresponding WebML metamodel excerpt in Figure 5.8*(c)*). Please note that, in the absence of a notational element in the WebML language, the abstract syntax in the style of UML object diagrams is used, as is shown with the Pointcut *ContentModel*.

2. The second way of extending the *ArbitraryWebApplication* WebML model with the new Entities is illustrated in Figure 5.8*(b)* with the AsymmetricCompositionRule *EntitiesBeforeUser*. In some cases, where the order of elements in a reference is important, the modeler might want to insert the modeling elements specified in the Advice at a specific point within a reference. For example, the modeler might want to insert an additional SortAttribute to an IndexUnit so that the entity instances displayed are sorted according to this specification. In this context, the order in which the attributes shall be considered during the sorting process is important (cf. Section 5.3.2.1). While the order of entities typically is not significant for a web application, the previous example shall be reused for illustration purposes and show how to model these requirements in Figure 5.8*(b)*. The Entities *News* and *Events* specified within the Advice *NewEntities* need to be inserted into the content model. More specifically, they need to be inserted into the *entity* containment reference *before* the Entity *User* of the content model (cf. Pointcut *User*). The RelativePositionKind *before* of the AsymmetricCompositionRule is again indicated by the icon of the rule.

Finally, in the AsymmetricCompositionRule *Pages2Siteview*, the landmark Pages defined in the Advice *NewLandmarkPages* are to be added to the *Public* Siteview of the WebML model, i.e., the Siteview which is available for all users of the web application. The rule is illustrated in Figure 5.9.

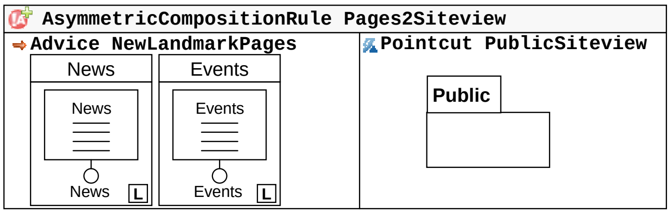


Figure 5.9: The *Pages2Siteview* AsymmetricCompositionRule

5.3.1.2 Composing Concerns with the aspectWebML Composition Algorithm

The aspectWebML composition semantics have been implemented within the aspectWebML tool support. Since the aspectWebML language has been defined in Ecore, it is possible to profit from EMF's code generation facilities to automatically produce a Java-based API for the aspectWebML language and furthermore to automatically produce modeling support on the basis of a tree-based modeling editor. The aspectWebML composition algorithm thus has been implemented in Java and is integrated within the modeling environment of aspectWebML (cf. Chapter 7), which is available for downloaded[3]. More importantly, EMF comes with a full-fledged reflection mechanism which is vital when it comes to operating at different meta-levels as was specifically necessary within the composition algorithm when processing AsymmetricCompositionRules (cf. Section 5.3.2). What follows is an explanation of how the previously separated concerns available in terms of ConcernModules can be composed with the composition algorithm.

The prerequisite for composition in aspectWebML is a CompositionPlan, which specifies how the concerns shall be composed. As already indicated in Section 5.2, the composition algorithm is implemented within the *executeCompositionPlan()* operation. Consequently, having defined all ConcernModules, i.e., the WebML model and all Aspects to be applied to the WebML model, as well as the required AsymmetricCompositionRules, the CompositionPlan can be configured. The CompositionPlan requires a ConcernModuleSequence and a ConcernCompositionRuleSequence to be defined. The first specifies the order of how the ConcernModules shall be composed. The second, allows to specify the order of the ConcernCompositionRules, allowing to fine-tune the order of an Aspect's Advice.

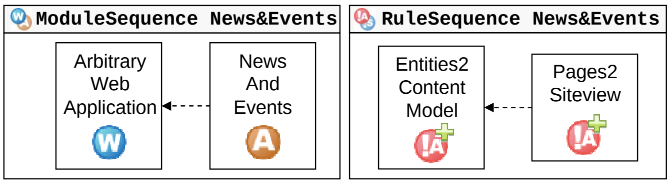


Figure 5.10: Configuration of the *ArbitraryWebApplication* CompositionPlan

With respect to the example, both sequences are depicted in Figure 5.10: The Aspect *NewsAndEvents* is applied to the *ArbitraryWebApplication* WebML model, while the Advice *Entities2ContentModel* is to be processed before the Advice *Pages2Siteview*.

In specifying both sequences, the modeler needs to pay particular attention to possible interactions between the ConcernModules. In case of dependencies between Aspects, the Aspect being dependent on another needs to be composed after the one it depends on. Furthermore, the modeler needs to pay particular attention to the order of the AsymmetricCompositionRules in case one of the rule has an EffectKind other than *enhancement*. For example, the modeler might specify an AsymmetricCompositionRule which causes a modeling element to be deleted. It is crucial, that this modeling element is not used in the Pointcut of another AsymmetricCompositionRule which follows the previous rule. In the *consistencyCheck()* operation of the CompositionPlan these kind of situations are detected and the modeler is warned.

The actual composition is carried out in two separate steps as is outlined within the UML activity diagram of Figure 5.11*(a)*:

[3] *www.wit.at/people/schauerhuber/aspectUWA*

1. Consistency Check
2. Composition of ConcernModules

In the first step, the CompositionPlan is validated. The algorithm checks for possible interactions, i.e., dependencies between ConcernModules and between ConcernCompositionRules. In case, the *CheckConsistency* action detects errors in the CompositionPlan, the algorithm produces appropriate warnings (cf. *ProduceWarnings* action) and terminates. For example, the algorithm might detect that an Aspect of the ConcernModuleSequence is dependent on another Aspect which is not part of the ConcernModuleSequence. Accordingly, the algorithm produces an appropriate warning requesting the user to include the missing Aspect.

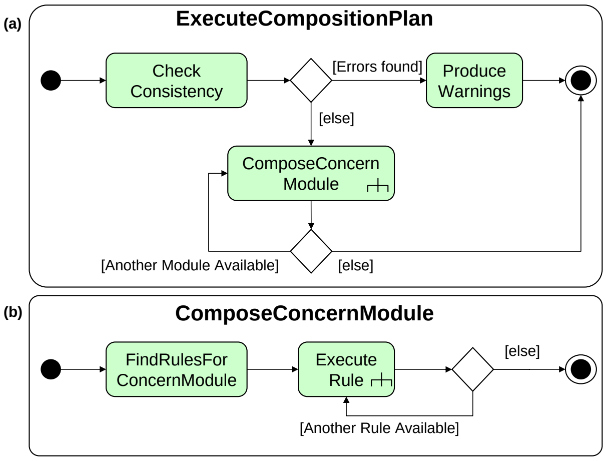


Figure 5.11: Overview on the aspectWebML Composition Algorithm

Given there are no errors, in the second step, the composition algorithm starts with the actual composition. The algorithm iterates over the ConcernModules in order to compose them one after the other (cf. *ComposeConcernModule* action). The composition algorithm starts composing the first two ConcernModules. The result of their composition is considered when composing the third ConcernModule in the sequence, etc.

As is specified by the activity *ComposeConcernModule* in Figure 5.11*(b)*, when composing a ConcernModule, the composition algorithm finds the corresponding ConcernCompositionRules in the given sequence (cf. action *FindRulesForConcernModule*). At this point, the composition algorithm will provide a separate control flow for asymmetric concern composition as well as for symmetric concern composition, which is not shown due to the focus on asymmetric concern composition. After that, the composition algorithm will continue with processing each rule one after the other. The details of the *ExecuteRule* action will be discussed in the following section.

Considering existing composition algorithms supporting asymmetric composition, two implementations are currently available, i.e., the weaving algorithm of Klein et al. and the Motorola Weavr of Cottenier et al. already discussed in Section 3.3.4 and Section 3.3.5, respectively. Similar to these implementations, one can distinguish between a *detection* and a *composition* phase as will be explained in Section 5.3.2.4. Since based on the UML for composing sequence diagrams and state machines, respectively, these two approaches obviously are not reusable within this thesis.

5.3.2 The aspectWebML Composition Semantics in Detail

When introducing the aspectWebML metamodel in Section 5.2, several ways of specifying Pointcuts and Advice have been pointed out. Furthermore, when modeling AsymmetricCompositionRules, modelers need to specify the EffectKind as well as the RelativePositionKind. In Table 5.1, an overview on the possible combinations of EffectKind, RelativePositionKind, Pointcut, and Advice is provided, resulting in thirteen different kinds of AsymmetricCompositionRules to be considered in the composition algorithm. What follows next is a description of each of the thirteen cases grouped according to their EffectKind (cf. Section 5.3.2.1 to 5.3.2.3). For each case, a modeling example as well as discussion on the issues to be considered in the composition algorithm is provided. Finally, an outline of how the composition algorithm processes these different kinds of AsymmetricCompositionRules is given in Section 5.3.2.4. For a discussion of the implementation details of the composition algorithm, the interested reader is referred to [Tom07].

	AsymmetricCompositionRule							
		relative	Pointcut			Advice		
Case	effect	Position	joinPoint	joinPointFeature	expression	aspectElement	aspectElementReference	expression
(1)	Enhancement	🟦	ModelElement(s) e.g. ENTITYs	Optional Name of Containment Reference e.g. attribute	🟪	Contained ModelElement e.g. ATTRIBUTEs	🟦	🟦
(2)		Before/ After	ModelElement(s) e.g. SORTATTRIBUTE	🟦	🟪	ModelElement(s) of same Type as joinPoint e.g. SORTATTRIBUTE	🟦	🟦
(3)		Around	ModelElement(s) e.g. PAGE	🟦	🟪	Container ModelElement of joinPoint e.g. AREA	🟦	🟦
(4)		🟦	ModelElement(s) e.g. AREA	Optional Name of Reference e.g. defaultPage	🟪	🟦	Reference to existing ModelElement(s) e.g. PAGE	🟦
(5)		🟦	ModelElement(s) e.g. ATTRITBUE Price	Name of Attribute e.g. value	🟪	🟦	🟦	String value e.g. "Self.WholesalePrice*1,2"
(6)		Before/ After	ModelElement(s) e.g. INDEXUNIT	Name of Reference plus Index e.g. displayattribute[i]	🟪	🟦	Reference to existing ModelElement e.g. ATTRIBUTE	🟦
(7)	Replacement	🟦	ModelElement(s) to be replaced e.g. ENTITY	🟦	🟪	ModelElement(s) used for Replacement e.g. ENTITY	🟦	🟦
(8)		🟦	ModelElement(s) to be replaced e.g. AREA	Optional Name of Reference e.g. defaultPage	🟪	🟦	Reference to existing ModelElement(s) e.g. PAGE	🟦
(9)		🟦	ModelElement(s) to be replaced e.g. ENTITY User	Name of Attribute e.g. name	🟪	🟦	🟦	String value e.g. "Person"
(10)		🟦	ModelElement(s) to be replaced e.g. INDEXUNIT	Name of Reference plus Index e.g. displayattribute[i]	🟪	🟦	Reference to existing ModelElement(s) e.g. ATTRIBUTE	🟦
(11)	Deletion	🟦	ModelElement(s) to be deleted e.g. ENTITY	🟦	🟪	🟦	🟦	🟦
(12)		🟦	ModelElement(s) to be deleted e.g. ENTITY	Name of Feature E.g., name, superEntity	🟪	🟦	🟦	🟦
(13)		🟦	ModelElement(s) to be deleted e.g. INDEXUNIT	Name of Reference plus Index e.g. displayattribute[i]	🟪	🟦	🟦	🟦

Legend: □ Part of Rule Configuration 🟦 Not Applicable 🟩 Optional Part of Rule 🟪 Alternative to joinPoint

Table 5.1: The 13 Kinds of AsymmetricCompositionRules in *aspectWebML*

5.3.2.1 Enhancement AsymmetricCompositionRules

Case 1 - Extending a Web Application Model With Further ModelElements

The majority of AsymmetricCompositionRules will be of this kind. The developer might want to extend a model with some further modeling elements. Depending on the modeling elements s/he will group them within several Advice, i.e., the modeling elements are grouped according to their container so that they can be applied to the WebML model within different AsymmetricCompositionRules having appropriate Pointcuts.

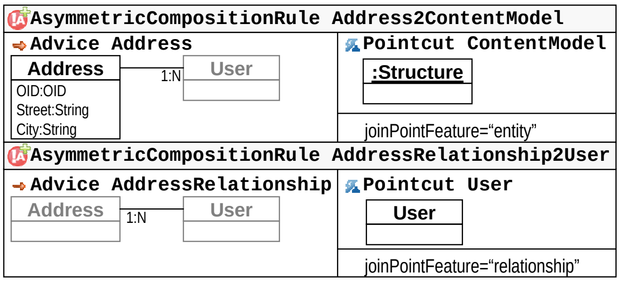

AsymmetricCompositionRule Address2ContentModel
Advice Address
Address
OID:OID
Street:String
City:String
1:N
User
Pointcut ContentModel
:Structure
joinPointFeature="entity"
AsymmetricCompositionRule AddressRelationship2User
Advice AddressRelationship
Address
1:N
User
Pointcut User
User
joinPointFeature="relationship"

Figure 5.12: AsymmetricCompositionRule: Case 1

In Figure 5.12, the content model needs to be extended with an Entity *Address*, having a Relationship to the existing Entity *User* in the content model. In the AsymmetricCompositionRule *Address2ContentModel*, the Advice *Address* specifies the Entity to be inserted together with its Attributes and its Relationship to the Entity *User* (cf. *Advice.aspectElement* containment reference). The Pointcut *ContentModel* specifies the Structure meta-class of WebML to be the JoinPoint (cf. *SimplePointcut.joinPoint* reference). The modeler can optionally specify the containment reference of the Pointcut (cf. *joinPointFeature* attribute) to which the elements in the Advice are to be added, i.e., *entity* in the example (cf. Section 4.4.2). Please note, that modeling elements are highlighted in grey if they are defined either in the WebML model or within another Advice. This notation shall indicate that modeling elements in the Advice have pointers to modeling elements outside the Advice. Again, in the absence of a notational element in the WebML language, UML object diagrams shall be used, as is shown with the Pointcut *ContentModel*. A second AsymmetricCompositionRule *AddressRelationship2User* is required to model the inverse relationship from the *User* Entity to the *Address* Entity. This time, the Pointcut specifies the *User* Entity to be the JoinPoint and the Advice consists of the Relationship pointing to the *Address* Entity (highlighted in grey) previously defined within the Advice *Address*.

When processing this kind of rule during composition, the modeling elements of the Advice (cf. *Advice.aspectElement* containment reference) need to be added to the correct containment reference of the specified Pointcut. This containment reference is either provided by the user or needs to be computed at runtime. With this particular example, two peculiarities of the aspectWebML language shall be pointed out:

- Due to the structure of the WebML metamodel, one has to define the extension of the content model within two separate Advice to be applied to two different Pointcuts. For the human eye, such an extension is often considered as a whole, while for the composition algorithm it has to be disassembled into several parts, i.e., several Advice.

- Typically, these Advice are coupled in terms of having modeling elements pointing to modeling elements within other Advice. For the above example, the Advice are applied only once to the WebML model, i.e., each Advice is applied in one AsymmetricCompositionRule having one JoinPoint, only. Still, the more common case for a crosscutting concern is that coupled Advice are to be applied to several JoinPoints. For example, an IndexUnit providing recommendations to the user of web application might be inserted to a large set of Pages. Some similar examples can be found in the case study of Chapter 6. In order to support these cases the composition algorithm gets quite complex, since all the couplings between Advice need to be traced during composition. For an in-depth discussion of the required trace model in the composition algorithm's implementation the reader is referred to [Tom07].

CASE 2 - EXTENDING A WEB APPLICATION MODEL WITH FURTHER MODELELEMENTS IN A SPECIFIC ORDER.

At some time, it might be necessary to introduce modeling elements to the web application model while considering a specific order. For example, a SortAttribute needs to be introduced before/after another one in order to influence how the entity instances of an IndexUnit shall be sorted. Consequently, this kind of AsymmetricCompositionRule configuration is to be used where the order of modeling elements is important.

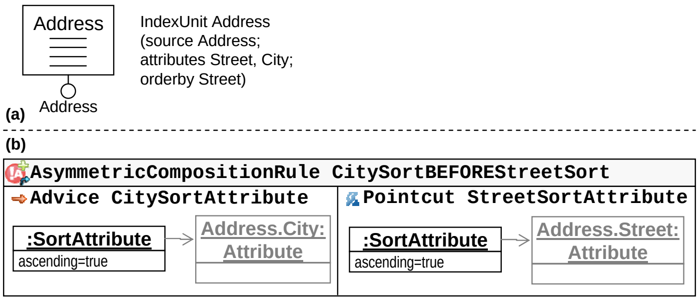


Figure 5.13: AsymmetricCompositionRule: Case 2

The example given in Figure 5.13*(a)* shows an IndexUnit displaying instances of Entity *Address*. WebML's textual representation furthermore reveals, that the *Street* and *City* Attributes are to be displayed and that the instances are sorted by the *Street* Attribute. Figure 5.13*(b)* presents the AsymmetricCompositionRule *CitySortBEFOREStreetSort* which shall introduce to the IndexUnit another SortAttribute, i.e., the instances of the Entity *Address* shall be sorted in ascending order according to the *City* Attribute and then according to the *Street* Attribute. Thus, in the Advice *CitySortAttribute*, a new SortAttribute is specified having a reference to the *City* Attribute of Entity *Address*. In the rule, this Advice is added *before* the existing SortAttribute, which is identified by the Pointcut *StreetSortAttribute*.

When processing this kind of rule, the modeling elements defined in *SimpleAdvice.aspectElement* need to be added to the container of the Pointcut, i.e., to the *sortattribute* containment reference of the IndexUnit in the example (cf. Section 4.4.4). More specifically, the modeling elements are added to the containment reference before or after the specified Pointcut. Consequently, it is important that they have the same type or super-type as the JoinPoints specified in the Pointcut.

CASE 3 - EXTENDING A WEB APPLICATION MODEL WITH A MODELELEMENT PLACED AROUND A SET OF OTHERS.

Case 3 represents the only kind of AsymmetricCompositionRule that uses the RelativePositionKind *around*. This kind of rule is used, if the modeler wants to place a model element around a set of other model elements, e.g., the modeler might want to draw a Transaction around a set of OperationUnits. Please note, that in the open class composition mechanism (originally for introducing aspectual structure at programming level such as additional attributes to a class) the RelativePositionKind is interpreted differently than in the pointcut-advice composition mechanism (originally for introducing aspectual behavior at programming level such as intercepting method calls). In the example, the modeling element defined in the SimpleAdvice is placed around the JoinPoints specified in the Pointcut. This requires, that the JoinPoints can be contained by the modeling element of the Advice.

In the example given in Figure 5.14, the WebML model has a landmark page named *News* and a Page *News Details* as is specified in the Pointcut of the AsymmetricCompositionRule *NewsArea-AROUNDNewsPages*. In the composed WebML model, there shall be a landmark Area *News*, which contains the existing *News* Page and *News Details* Page as well as a default Page *Highlights*. Thus, in the Advice *NewsArea*, the Area *News* is modeled having as default page the *Highlights* Page. After composition, all three pages will be contained by the Area *News*.

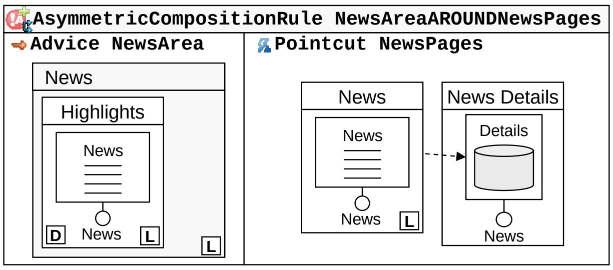


Figure 5.14: AsymmetricCompositionRule: Case 3

When processing this kind of rule, the modeling element of the SimpleAdvice (cf. containment reference *SimpleAdvice.aspectElement*) needs to be added to the container of the Pointcut in a first step. In the example, this container is the home Siteview of the WebML model which contains the pages specified in the Pointcut *NewsPages*. Consequently, the Area *News* including the *Highlights* Page needs to be added to the *area* containment reference of the Siteview (cf. Section 4.4.3). In a second step, the JoinPoints defined within the Pointcut need to be moved to the modeling element of the SimpleAdvice. In the example, this means that the *News* Page and the *News Details* Page will be moved from the home Siteview to the *News* Area.

CASE 4 - SETTING A MODELELEMENT'S REFERENCE.

This AsymmetricCompositionRule kind is used in case a modeler wishes to set a modeling element's reference to an existing modeling element. This means in the Advice, no new modeling element needs to be specified within the *SimpleAdvice.aspectElement* containment reference. Instead, the Advice needs to capture the reference to the existing modeling element within *SimpleAdvice.aspectElementReference*. In the example, the News *Area* has no default page set. Thus, the AsymmetricCompositionRule *AddDefaultPage2Area* in Figure 5.15 provides the reference to the existing Page *Highlights* of the *News* Area in the Advice *HighlightsPage*. In the rule's Pointcut *NewsAreaDefaultPage* the *News* Area is specified as well as optionally the *Pointcut.joinPointFeature* of the Area meta-class, i.e., *defaultPage*, to which the Advice shall be applied.

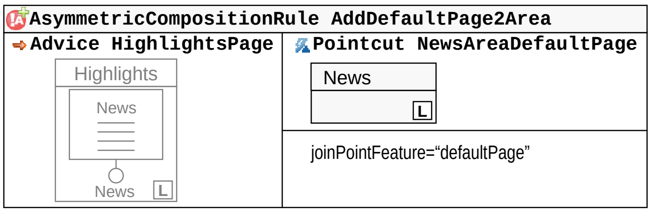

AsymmetricCompositionRule AddDefaultPage2Area
Advice HighlightsPage
Pointcut NewsAreaDefaultPage
Highlights
News
News
L
News
L
joinPointFeature="defaultPage"

Figure 5.15: AsymmetricCompositionRule: Case 4

During composition, the *SimpleAdvice.aspectElementReference* needs to be added to the given *Pointcut.joinPointFeature*. In the example, the *defaultPage* of the *News* Area will point to the *Highlights* Page after composition.

CASE 5 - SETTING A MODELELEMENT'S ATTRIBUTE.

If the modeler needs to set an attribute of a modeling element, s/he will use the AsymmetricCompositionRule of Case 5. For example, the modeler might want to define a derived Attribute, i.e., to set the *value* meta-attribute of the Attribute meta-class. In this case, the modeler needs to provide the new value of the attribute to be set within the Advice (cf. *SimpleAdvice.expression* attribute). In addition, in the *Pointcut.joinPointFeature* of the Pointcut, the modeler has to specify which attribute needs to be set. In the example given in Figure 5.16, the Entity *Product* is assumed to have two Attributes, namely *Price* and *WholesalePrice*. In the AsymmetricCompositionRule *AddValue2PriceAttribute.Value*, the *Price* Attribute shall be defined to be derived from the *WholesalePrice* Attribute, i.e., the *value* of the *Price* Attribute needs to be set. More specifically, the *Price* is equal to the *WholesalePrice* plus 20% tax. Consequently, in the Advice *Value*, the *SimpleAdvice.expression* needs to be set to the required derivation String *Self.WholesalePrice*1,2* (cf. WebML's derivation language [CFB$^+$03]). The Pointcut *PriceAttributeValue* specifies the modeling element, i.e., the *Price* Attribute, and in the *joinPointFeature* the modeling element's meta-attribute that needs to be set, i.e., the *value*.

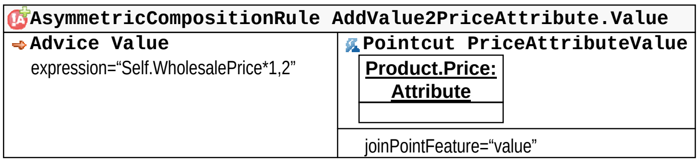

AsymmetricCompositionRule AddValue2PriceAttribute.Value
Advice Value
expression="Self.WholesalePrice*1,2"
Pointcut PriceAttributeValue
Product.Price:
Attribute
joinPointFeature="value"

Figure 5.16: AsymmetricCompositionRule: Case 5

During composition, the *expression* of the Advice needs to be set to the given *joinPointFeature* of the Pointcut. In the example, the *value* of the *Price* Attribute will be set to *Self.WholesalePrice*1,2* after composition.

CASE 6 - EXTENDING A MODELELEMENT'S REFERENCE IN A SPECIFIC ORDER.

At some time, it might be necessary to introduce a reference instance to a modeling element while considering a specific order. For example, the IndexUnit meta-class has a reference pointing to the Attributes of the IndexUnit's Entity (cf. *IndexUnit.displayAttribute*). In a web application model, the order of the reference instances to the Attribute will determine their display order in the web application. For example, a reference instance to an Attribute might be inserted before/after an existing reference instance in order to change the display order. Consequently, this kind of AsymmetricCompositionRule configuration is to be used where the order of references to modeling elements is important. In contrast to Case 2, this kind of AsymmetricCompositionRule does not operate on a containment reference but on a normal reference. This means in the Advice, references to existing modeling elements (cf. *SimpleAdvice.aspectElementReference* reference) are provided instead of modeling new elements.

The example given in Figure 5.17*(a)* again shows an IndexUnit displaying instances of Entity *Address*. WebML's textual representation furthermore reveals, that the *City* Attribute is to be displayed. In this case, the IndexUnit shall also display the *Street* Attribute *before* the *City* Attribute. This is realized in Figure 5.17*(b)* with the AsymmetricCompositionRule *DisplayStreetBEFORECity*: In the rule's Advice *StreetAttribute*, the reference to the *Street* Attribute of Entity *Address* is provided and needs to be inserted. The rule's Pointcut *CityDisplayAttribute* specifies the IndexUnit *Address*. Furthermore the Pointcut's *joinPointFeature* denotes the reference where the *Street* Attribute reference needs to be inserted. Please note that, the modeler furthermore needs to specify the index of the *City* Attribute reference. This is necessary, in order to be able to insert the reference to the *Street* Attribute reference before the one of the *City* Attribute.

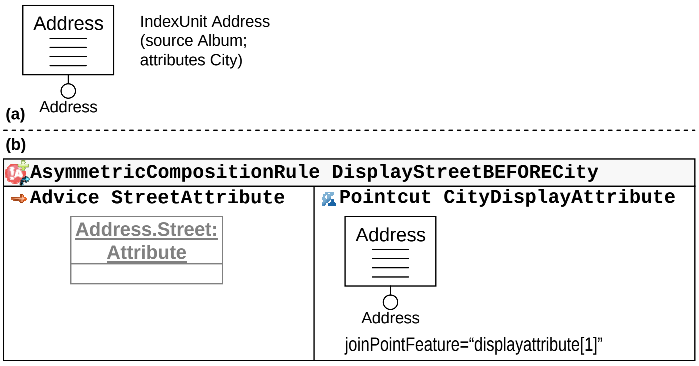


Figure 5.17: AsymmetricCompositionRule: Case 6

When processing this kind of rule, the references specified in *SimpleAdvice.aspectElementReference* need to be added to the *joinPointFeature* of the Pointcut, i.e., to the *displayattribute* reference of the IndexUnit in the example (cf. Section 4.4.4). More specifically, the references are added to the *displayattribute* reference before or after the specified index in the *joinPointFeature*. Consequently, it is important that the references specified in *SimpleAdvice.aspectElementReference* have the same type or super-type as the reference specified in the Pointcut's *joinPointFeature*.

5.3.2.2 Replacement AsymmetricCompositionRules

CASE 7 - REPLACING MODELELEMENTS OF THE WEB APPLICATION MODEL WITH NEW ONES.
Sometimes, the modeler needs to replace previously defined modeling elements with new ones. The AsymmetricCompositionRule kind of Case 7 represents the necessary configurations of the rule. Figure 5.18*(a)* provides an example, where the *User* Entity specified in the Pointcut *User* of the AsymmetricCompositionRule *ReplaceUserWITHPerson* needs to be replaced with a *Person* Entity specified in the Advice *Person*.

While processing this rule, the modeling element specified by the Pointcut is deleted and instead, the modeling element specified within the Advice is inserted into the container of the Pointcut, i.e., the *entity* containment reference of the Structure meta-class in the example (cf. Section 4.4.2). Consequently, it is important that the modeling element defined in *SimpleAdvice.aspectElement* has the same type or super-type as the JoinPoint specified in the Pointcut.

Furthermore, the following issues need to be considered:

- Since the replaced modeling element actually is deleted, all pointers to it as well as to its children (e.g., the *User* Entity's Attributes and Relationships to other Entities) are lost. The current implementation of the composition algorithm, keeps track of the pointers to the modeling element of the Pointcut and lets them point to the modeling element of the Advice after composition. Thus, in the example, DataUnits defined for the *User* Entity will point to the *Person* Entity after composition. Still, the *User* Entity's Attributes and Relationships have been deleted as well, meaning that pointers to them are now dangling. Consequently, all these pointers need to be unset. For example, after composition, the DataUnits that will then be defined for the *Person* Entity will have dangling pointers to the *User* Entity's Attributes in their *displayattribute* reference. The composition algorithm will unset the dangling references and the modeler will have to set the Attributes to be displayed in a separate AsymmetricCompositionRule.
- Keeping track of pointers is always done for the first *aspectElement* of an Advice and the first JoinPoint of the Pointcut, only. Thus, in case the modeler replaces the *User* Entity with a *Person* Entity and a *Student* Entity, all pointers will be resolved to the *Person* Entity instead of the *Student* Entity.

The example shows, that replacing an Entity possibly causes some problems in the composed model. This is due to possible dependencies between the Entity and modeling elements within the content model and the hypertext model. In this case, the modeler needs to balance the trade off between defining a replacement rule or just renaming the Entity *User* and defining some more rules for changing Attributes and Relationship of the previous Entity *User*.

CASE 8 - RE-SETTING A MODELELEMENT'S REFERENCE.
This AsymmetricCompositionRule kind is similar to Case 4, with the difference that it is used to re-set a modeling element's reference to an existing modeling element instead of initializing it. This means in the Advice, the reference to the existing modeling element(s) within *aspectElementReference* is captured. In the example, the News *Area* is assumed to have its *defaultPage* set to the *News* page (not shown). Thus, the AsymmetricCompositionRule *ReplaceDefaultPageOFArea* in Figure 5.18*(b)* provides the reference to the existing Page *Highlights* of the *News* Area in the Advice *HighlightsPage*. In the rule's Pointcut *NewsAreaDefaultPage*, the *News* Area is specified as well as

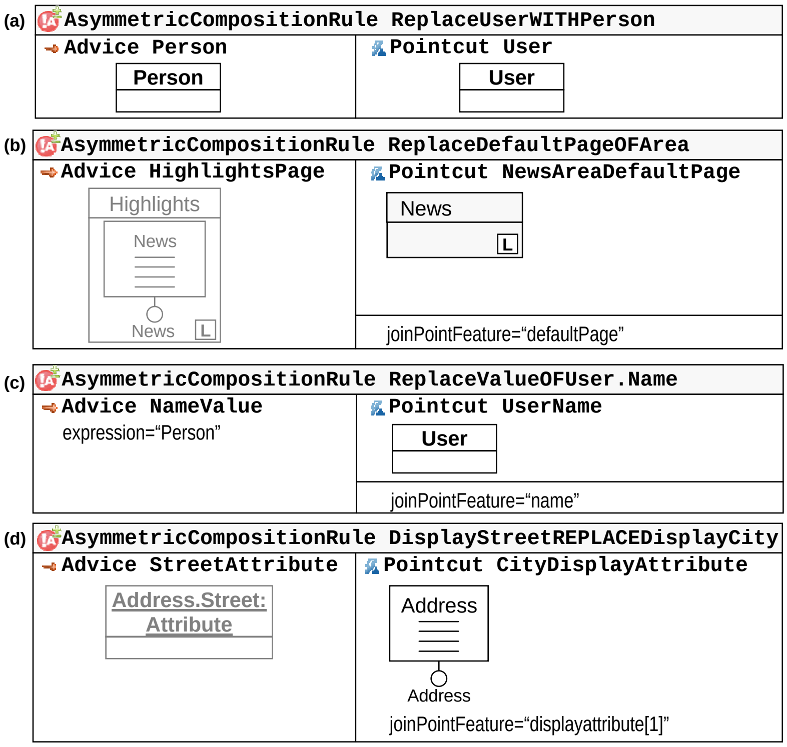

(a)
AsymmetricCompositionRule ReplaceUserWITHPerson
Advice Person
Person
Pointcut User
User
(b)
AsymmetricCompositionRule ReplaceDefaultPageOFArea
Advice HighlightsPage
Highlights
News
News
L
Pointcut NewsAreaDefaultPage
News
L
joinPointFeature="defaultPage"
(c)
AsymmetricCompositionRule ReplaceValueOFUser.Name
Advice NameValue
expression="Person"
Pointcut UserName
User
joinPointFeature="name"
(d)
AsymmetricCompositionRule DisplayStreetREPLACEDisplayCity
Advice StreetAttribute
Address.Street: Attribute
Pointcut CityDisplayAttribute
Address
Address
joinPointFeature="displayattribute[1]"

Figure 5.18: AsymmetricCompositionRule: Case 7-10

optionally the *joinPointFeature* of the Area meta-class, i.e., *defaultPage*, to which the Advice shall be applied.

During composition, the *aspectElementReference* of the Advice needs to replace the previous reference given by *joinPointFeature* of the Pointcut. In the example, the *defaultPage* of the *News* Area will point to the *Highlights* Page instead of to the *News* Page after composition.

Case 9 - Re-Setting a ModelElement's Attribute.

If the modeler needs to change an attribute of a modeling element, s/he will use the AsymmetricCompositionRule of Case 9. This is similar to Case 5, which is used to initialize a model element's attribute. For example, the modeler might want change the *name* of an Entity. In this case, the modeler needs to provide the new value of the attribute to be set within the Advice (cf. *SimpleAdvice.expression* attribute). In addition, in the *joinPointFeature* of the Pointcut, the modeler has to specify which attribute needs to be set. In the example given in Figure 5.18*(c)*, the Entity *User* shall be renamed to *Person*. Consequently, in the Advice *NameValue*, the *expression* is set to the required String 'Person'. In the Pointcut *UserName*, the modeling element is specified, i.e., the *User* Entity. Furthermore, the *joinPointFeature* specifies the modeling element's meta-attribute that needs to be changed, i.e., the *name*.

During composition, the *expression* of the Advice needs to replace the value of the given *joinPointFeature* of the Pointcut. In the example, the *name* of the *User* Entity will have changed to *Person* after composition.

Case 10 - Replacing a ModelElement's Reference Instance

This kind of AsymmetricCompositionRule is to be used where a specific instance of a model element's reference needs to be replaced. Consider an IndexUnit and the sub-set of its Entity's Attributes that shall be displayed. A modeler might want to replace one of the Attributes with another one, e.g., a *ShortDescription* Attribute with a *LongDescription* Attribute. In the metamodel, the relationship to the Attributes is specified as a reference of the IndexUnit's meta-class (cf. *displayattribute*). Consequently, similar to the previously explained Case 6, this kind of rule operates on a the instances of a model element's reference.

In the example of Figure 5.18*(d)*, the IndexUnit *Address* is assumed to have one Attribute to be displayed, i.e., the *City* Attribute of Entity *Address*. In the composed model, however, the *Street* Attribute of Entity *Address* shall be displayed instead, as is specified in the AsymmetricCompositionRule *DisplayStreetREPLACEDisplayCity*: In the Advice *StreetAttribute* the reference to the *Street* Attribute is specified. The Pointcut specifies the IndexUnit *Address* as well as its *joinPointFeature*, denoting the reference where the *Street* Attribute reference needs to be inserted. Again, the modeler furthermore needs to specify the index of the *City* Attribute reference. This is necessary, in order to be able to replace the reference to the *City* Attribute with the one of the *Street* Attribute.

When processing this kind of rule, the *aspectElementReference* of the Advice needs to be inserted into the *joinPointFeature* of the Pointcut, i.e., to the *displayattribute* reference of the IndexUnit in the example (cf. Section 4.4.4). More specifically, the *aspectElementReference* is inserted into the *displayattribute* reference at the specified index instead of the previous reference. Consequently, it is important that the *aspectElementReferences* of the Advice have the same type or super-type as the reference specified in the Pointcut's *joinPointFeature*.

5.3.2.3 Deletion AsymmetricCompositionRules

For AsymmetricCompositionRules having a *deletion* effect, modelers need to specify a Pointcut, only. The Pointcut identifies the JoinPoints in the WebML model to be deleted. Following, three kinds of AsymmetricCompositionRules having a deletion effect can be distinguished.

Case 11 - Deleting ModelElements from the Web Application Model.

In order to delete modeling elements from a WebML model, the modeler specifies in the Pointcut all JoinPoints to be deleted. In Figure 5.19*(a)*, the Pointcut *User* of the AsymmetricCompositionRule *DeleteUser* specifies the *User* Entity to be deleted from the content model.

When processing the rule, the modeling elements are simply deleted from the WebML model. Still, the modeler needs to consider dependencies between the deleted modeling elements and others. After deletion, pointers to the deleted modeling elements will be dangling and need to be dealt with in other rules.

Case 12 - Unsetting a ModelElement's Attribute or Reference .

Likewise, modelers might need to delete the value of a modeling element's attribute or reference. The AsymmetricCompositionRule *DeletePriceAttribute.Value* in Figure 5.19*(b)* illustrates how a meta-attribute of *Price* Attribute of the *Product* Entity shall be deleted. The derivation string stored in the *value* attribute of the Attribute meta-class shall be deleted. Consequently, in the Pointcut, the *Price* Attribute as well as the *joinPointFeature value* needs to be specified. During composition, the value of the meta-attribute, i.e., the value of *Attribute.value*, needs to be unset.

In Figure 5.19*(c)*, an example for deleting a modeling element's reference is shown. In the example, the inheritance relationship of the *User* Entity to its super-entity shall be deleted. Thus, in the AsymmetricCompositionRule *DeleteUser.superentity*, the Pointcut specifies the *User* Entity as JoinPoint as well as the the *superentity* reference that needs to be deleted in the *joinPointFeature*. During composition, the meta-reference, i.e., *Entity.superentity*, needs to be unset.

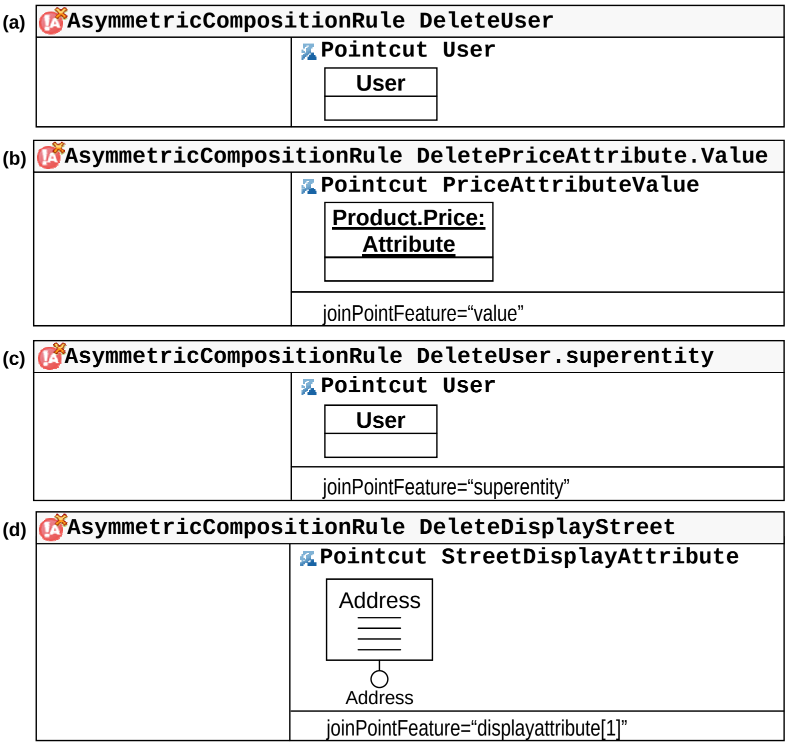

(a)
AsymmetricCompositionRule DeleteUser
Pointcut User
User
(b)
AsymmetricCompositionRule DeletePriceAttribute.Value
Pointcut PriceAttributeValue
Product.Price:
Attribute
joinPointFeature="value"
(c)
AsymmetricCompositionRule DeleteUser.superentity
Pointcut User
User
joinPointFeature="superentity"
(d)
AsymmetricCompositionRule DeleteDisplayStreet
Pointcut StreetDisplayAttribute
Address
Address
joinPointFeature="displayattribute[1]"

Figure 5.19: AsymmetricCompositionRule: Case 11-13

CASE 13 - DELETING A MODELELEMENT'S REFERENCE INSTANCE.
Finally, modelers might wish to delete one specific reference instance of a ModelElement. In the example, the IndexUnit *Address* displays two Attributes of the Entity *Address*, namely *Street* and *City*. The AsymmetricCompositionRule *DeleteDisplayStreet* specifies how the *Street* Attribute reference shall be deleted. In the Pointcut, the IndexUnit *Address* is specified as the JoinPoint. Furthermore, the *joinPointFeature* of IndexUnit is given, i.e., the *displayattribute* reference which stores the reference instances to the Entity's Attributes that shall be displayed. The actual reference to be deleted from the *displayattribute* sequence is specified via an index in the *joinPointFeature*. After composition, the IndexUnit *Address* will display the *Street* Attribute, only.

5.3.2.4 Processing AsymmetricCompositionRules

This section outlines how the composition algorithm processes the previously identified kinds of AsymmetricCompositionRules. Consequently, the *ExecuteRule* Action of Figure 5.11 of the composition algorithm will be refined in a separate activity diagram which is depicted in Figure 5.20.

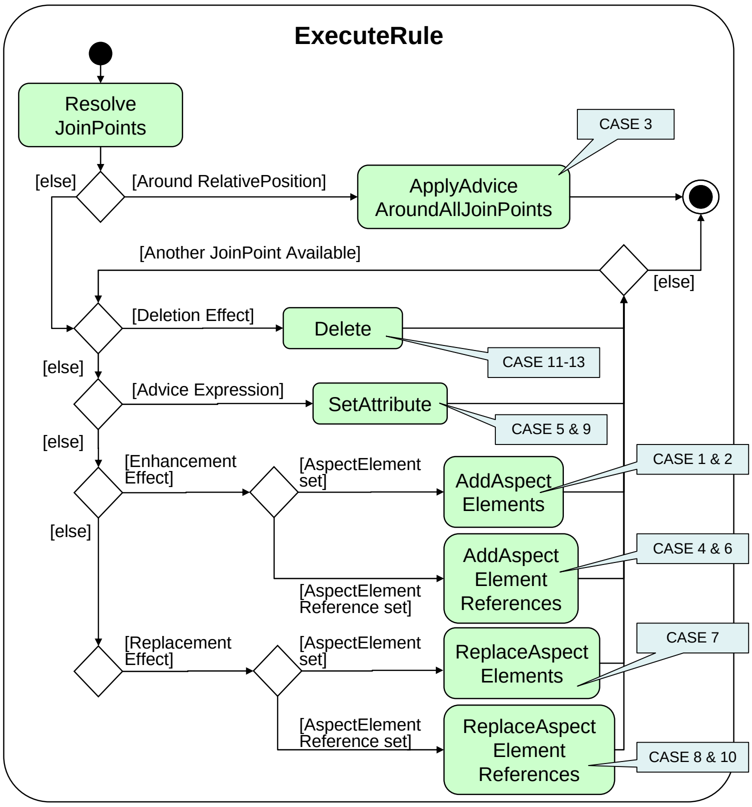


Figure 5.20: Processing Rules in the aspectWebML Composition Algorithm

In the figure, the parts of the activity realizing one or more of the previously defined cases are indicated.

In a first step, in the activity *ExecuteRule*, the JoinPoints of the rule's Pointcut need to be resolved. The *ResolveJoinPoints* action is responsible for establishing a list of JoinPoints specified in terms of an OCL query or an enumeration within a SimplePointcut or a CompositePointcut. This step corresponds to the previously mentioned *detection* phase of a composition algorithm. After that, the different kinds of AsymmetricCompositionRules are processed in separate control flows, representing the *composition* phase.

First of all, the algorithm deals with rules specified in Case 3. Rules having an *around* RelativePositionKind are applied to all JoinPoints at once in that the *aspectElement* of the Advice is placed around the JoinPoints (cf. *ApplyAdviceAroundAllJoinPoints* action). For all other rules, the list of JoinPoints is iterated and the Advice is applied to each JoinPoint separately.

Then the rules having a *deletion* EffectKind are considered. The *Delete* action in the activity covers Case 11 to Case 13 and is not further refined. Depending on the specific rule, the composition semantics defined in the previous sections have to be ensured.

Case 5 and 9 are captured by the *SetAttribute* action. Independent from the EffectKind, the attribute specified in the Pointcut's *joinPointFeature* can be set to the *expression* of the Advice.

Last but not least, the last major decision distinguishes between rules having an *enhancement* EffectKind and rules having a *replacement* EffectKind. In both cases, a further distinction with respect to how the Advice is defined is required, i.e., either through the *aspectElement* containment reference or the *aspectElementReference* reference. The *AddAspectElement* action thus is executed in case of an *enhancement* effect and in case the *aspectElement* reference is set. Within the action, a further distinction according to the rule's RelativePositionKind is necessary in order to support Case 1 (RelativePositionKind *into*) and Case 2 (RelativePositionKind *before/after*). If the *aspectElementReference* reference is set instead, the *AddAspectElementReferences* action is executed. Again a further distinction according to the rule's RelativePositionKind is necessary in order to support both Case 4 (RelativePositionKind *into*) and Case 6 (RelativePositionKind *before/after*).

Considering a replacement effect, the action *ReplaceAspectElements* is executed, if the *aspectElement* reference of the Advice is set thereby realizing Case 7. Finally, if no other action applies beforehand, the *ReplaceAspectElementReferences* action is executed. The action realizes Case 8 or Case 10 depending on the possibility of specifying an index within the Pointcut's *joinPointFeature*.

After the Advice has been applied to the JoinPoint, either the Advice is again applied to the next JoinPoint specified by the Pointcut or the rule has fully been applied to all JoinPoints and the composition algorithm proceeds to the next rule.

5.3.3 Supporting the Pointcut-Advice Composition Mechanism in aspectWebML

While the WebML language mainly supports modeling the structural features of a web application, some kind of behavior can be modeled by introducing content management functionality e.g., in terms of CreateUnits, ModifyUnits, and DeleteUnits, to a WebML model or can be found in the context clouds of context-aware Pages, Areas and Siteviews. In these cases, some flow of actions in terms of OperationUnits is available. Consequently, applying an Advice to such a flow of actions, e.g., before or after an OperationUnit, corresponds to using the pointcut-advice asymmetric composition mechanism. In aspectWebML, allowing for such composition semantics is not considered, since they would cause some problems which are explained in the following.

Imagine the context cloud of the context-aware Page *A* in Figure 5.21*(a)* needs to be extended. Currently, the context cloud ensures that the page is accessible for users only if they have already successfully logged in to the web application. The IfUnit of the context cloud checks if the user is known in the session by querying the *UserCtxParam* Parameter with a GetUnit. In case the user has not yet logged in to the web application, s/he is redirected to the Page *Login*. In the example, the context cloud shall be extended so that after composition, the context cloud resembles the one depicted in Figure 5.21*(c)*. This context cloud ensures that the user is able to visit Page *A* only if the user's location is known, or the user has already successfully logged in to the web application. The *Location.OID* is assumed to have been calculated by some outer context cloud in the web application and thus is available in the ContextUnit of Page *A*. The extension of the context cloud would require the IfUnit that checks the *Location.OID* to be inserted *before* the existing IfUnit of the context cloud. 'Before' means, that an OKLink of the inserted IfUnit needs to point to the existing IfUnit. The existing IfUnit will serve as the Pointcut. In case the condition of the inserted IfUnit evaluates to true, an OKLink to the ContextUnit shall be followed, else an OKLink to the existing IfUnit shall be followed. The outgoing Link from the ContextUnit needs to be redirected to the just inserted IfUnit and a LinkParameter needs to transport the *Location.OID* to the IfUnit.

This extension seems to be quite easy at first sight. In a composition algorithm, this use case has to be generalized, however, in order to be applicable to other OperationUnits such as ModifyUnits, a ChangeSiteviewUnit, and a SwitchUnit. The issues to be considered in a composition algorithm are the following:

- **Identification of the Outgoing Links' Targets**. The outgoing links, i.e., the OKLinks of the IfUnit or any OperationUnit, need to be specified in the Advice. Still, the link's targets must not be specified. These are to be computed by the composition algorithm which needs to decide which link will point to the Pointcut, i.e., the existing IfUnit in the example. The targets of the remaining outgoing links will have to be specified by the modeler after composition.
- **Identification of the Incoming Links' Targets.** The incoming links of the Pointcut, i.e., the existing IfUnit in the example, have to be redirected to the OperationUnit that is inserted. However, such a redirection might have unexpected side-effects on the flow of actions and might change the semantics of the context cloud.

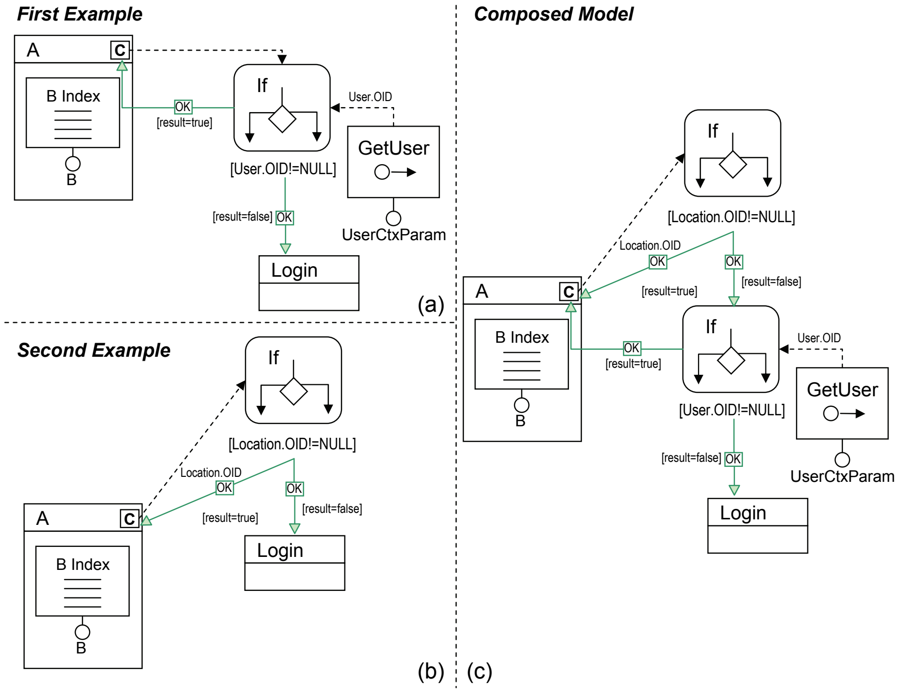


Figure 5.21: The *Pointcut-Advice* Composition Mechanism in aspectWebML

Now let's consider a second example, in Figure 5.21*(b)*, where the same context-aware Page *A* is depicted. In this case, however, the context cloud ensures that the page is accessible for users only if their location is available. Similar to the previous example, the context cloud shall be extended with a further check with respect to the login-status of the user. The extension of the context cloud would require the IfUnit that checks the *User.OID* to be inserted *after* the existing IfUnit of the context cloud, i.e., again the existing IfUnit will serve as the Pointcut. More specifically, the OKLink, which is followed if the condition *Location.OID!=NULL* resolves to false, shall point to the inserted IfUnit instead of the *Login* Page. The outgoing OKLinks of the inserted IfUnit shall point to the *Login* Page and the *A* Page, respectively as is depicted in Figure 5.21*(c)*.

In the composition algorithm, the major problem to resolve is how the IfUnit of the Advice can be inserted *after* the Pointcut, i.e., an existing OperationUnit. It is not clear if it shall be inserted for one outgoing link of the Pointcut or if it shall be inserted for all of them. In case the Advice shall be applied to one of the outgoing links, i.e., one of the OKLinks in the example, it is neither clear which of them shall be chosen.

For these reasons, behavioral flows in WebML are ignored and the WebML language is considered from a structural view point, only. This means in the specification of aspectWebML's composition semantics, the RelativePositionKind of an AsymmetricCompositionRule will always be interpreted as a relative position in the structure of a WebML model. Consequently, in order to realize the above extensions, the modeler will need to specify them with the open class composition mechanism as already explained before.

5.4 Summary

This chapter has been dedicated to the design of the *aspectWebML* web modeling language, i.e., an extension of the existing WebML language with concepts from the aspect-orientation paradigm in order to better support modeling of crosscutting concerns, i.e., in particular customization. The major contributions of this chapter are as follows:

First, the aspectWebML metamodel has been designed on the basis of the Conceptual Reference Model for aspect-oriented modeling of Chapter 3. In this respect, the Conceptual Reference Model has been used as a blueprint for bridging an existing web modeling language to the aspect-orientation paradigm.

Second, several ways of manipulating WebML models with Aspects have been provided going beyond their enhancement with modeling elements and the replacement as well as the deletion of existing modeling elements. More specifically, modelers are also provided with the possibility of changing existing modeling elements' attributes and references.

Third, allowing for OCL-based Pointcuts, modelers are enabled to define a repository of reusable Pointcuts which can be imported into other aspectWebML projects.

Fourth, the composition semantics of the aspectWebML language have been specified in detail. On the basis of examples, all possible ways of defining AsymmetricCompositionRules have been discussed as well as the issues to be considered during composition. Furthermore, an explanation of the mode of operation of the composition algorithm, which has been implemented in Java and is integrated within the modeling environment for aspectWebML, has been given

Due to the focus on supporting the development of UWAs, this thesis specializes in supporting asymmetric composition mechanisms which are more suitable to support customization modeling than symmetric composition mechanisms. Nevertheless, the concepts specified in the CRM to support symmetric composition mechanisms are incorporated into the aspectWebML metamodel as far as it is necessary to guarantee an easy extension with symmetric composition semantics at a later date. In future work, the compositor composition mechanism for aspectWebML shall be fully realized as well.

6 The Context-Aware Museum Case Study

Contents

This chapter presents the case study used to compare the WebML approach for modeling UWAs with the aspectWebML approach presented in this thesis. The *Context-Aware Museum* (CAM) web application represents a UWA that provides visitors of the museum with a customized access to general information as well as information on its indoor and outdoor exhibits, considering the user's context, his/her current location, and the device used. In the particular example, a non-ubiquitous museum web application shall be made context-aware by extending it with customization functionality. This kind of extension is of a constructive nature. As a consequence, for the aspectWebML part in the case study, the focus will be on defining AsymmetricCompositionRules having an enhancement effect on the museum web application. Following, the CAM web application and its (customization) functionalities is presented in Section 6.1. To allow for a comparison, the UWA is first developed with WebML's modeling means. Thus, the relevant parts of the resulting WebML model are presented in Section 6.2. In a second step, the CAM web application or rather its customization functionality is re-modeled using the aspectWebML approach (cf. Section 6.3). In doing so, the customization functionality extensions made to the museum web application will be extracted from the WebML model presented in Section 6.2 and encapsulated within a separate *Customization* Aspect. The *Customization* Aspect shall be designed in such a way, that after weaving it "back" into the museum web application model, the composed model is equivalent to the WebML solution. In this respect, the goal is to show that aspectWebML can be used to model the same CAM web application, while keeping all customization functionality separate from the rest of the web application. Thereafter, Section 6.4 is dedicated to a comparative discussion on aspectWebML's strengths and shortcomings. Finally, the chapter is closed with a summary in Section 6.5. The case study is available online together with the aspectWebML Modeling Environment (cf. Chapter 7), which allows testing the aspectWebML solution by composing the *Customization* Aspect with the core museum web application model.

6.1 Introducing the Context-Aware Museum

6.1.1 Motivation

As already discussed in Chapter 2, there are no generally acknowledged modeling examples for specific types of web applications including UWAs. In order to "assess" several web modeling approaches and their applicability to model customization functionality, in Chapter 2 a *Tourism Information Web Application* example has been introduced. This example has been chosen in order to not create biases in the evaluation, since it has not yet been presented in any of the investigated approaches. In contrast, for the case study in this chapter, the *Context-Aware Museum* modeling example has been chosen, which has already been used for introducing WebML's new modeling concepts for supporting customization modeling [CDMF07] including a demo[1]. The motivation in not reusing the running example of Chapter 2 is based on the assumption that WebML's developers will have chosen a modeling example which particulary points out WebML's strengths with respect to its new customization modeling concepts. The goal of this chapter is to compare the WebML approach to modeling UWAs with the aspectWebML approach and specifically point out aspectWebML's strengths with respect to the WebML approach. Consequently, the CAM web application example can be considered as a kind of "benchmark" and thus, represents the ideal modeling example candidate for the case study.

Nevertheless, for a better demonstration of aspectWebML's strengths, the original CAM web application presented in [CDMF07], has been considerably extended. More specifically, the set of user groups considered by the CAM web application has been extended in order to serve not only museum visitors with customized services, but also external users of the web site, registered general users, academic users, as well as curators of the museum. In contrast to the original CAM web application example, in this case study the set of supported context properties goes beyond location, including also information on the user, the device, the time as well as the weather context. As a consequence, the original customization scenario of the CAM web application already presented in Section 4.6 for introducing WebML's customization concepts, has been extended with *seven* further customization scenarios. With respect to the core functionality of the CAM web application, the example has been inspired by the *MAK Austrian Museum of Applied Arts / Contemporary Art*[2].

The example, i.e., the core and customization functionalities of the CAM web application, will be described in terms of a subset of artifacts produced in the *requirements specification* phase of WebML's development process [CFB$^+$03], as well as the output of the subsequent *data design* and *hypertext design* phases (cf. Section 6.2). It has to be noted, that the CAM web application presented in the following has not been implemented. In the context of this case study, the goal is to demonstrate two properties of the aspectWebML approach: First, the aspectWebML approach allows modeling customization functionality separately from the rest of the web application model. And second, after composition of the *Customization* Aspect with the core web application model, the composed model is equivalent to the WebML solution of the CAM web application model. Since the aspectWebML approach provides composition of concerns at modeling level, the model to be used to automatically generate a running web application, will be a WebML model without Aspects. As already mentioned before, WebRatio currently does not support developing UWAs, since the extensions to the WebML language proposed in [CDMF07] have not yet been incorporated into the tool support. Consequently, it was not possible to implement the application on the basis of WebRatio's code generation facilities.

In the following sub-sections, an overview on the CAM web application as well as a discussion on its functional requirements is provided.

[1] http://dblambs.elet.polimi.it/Demos/Museo/device.html

[2] http://www.mak.at/e/

6.1.2 Overview

In the case study, the fictitious *Museum of Applied and Contemporary Arts* requires a relaunch of its current website. More specifically, the functionality of the existing web application shall be extended with customization functionality. Users of the future CAM web application shall be able to access its services tailored according to their context of use, e.g., tailored to their age, their interests, their specific user role and access rights, their location, their device, as well as the current time and weather situation (due to the museum's outdoor areas).

The museum will be equipped with a wireless network infrastructure which will allow geopositioning users accessing the CAM web application with their laptops and PDAs using current wireless network standards. Geopositioning will be realized with a browser plugin based on existing geopositioning software such as *Placelab*[3], which users need to install in their Web browsers. Furthermore, the museum will set up information terminals - hiding ordinary PCs - across the museum's premises allowing museum visitors to access the CAM web application and gain more information on certain exhibits. Geopositioning of these users will be realized via IP resolution of the information terminals.

The target users of the CAM web application can be categorized into non-registered users and registered users. Non-registered users are able to access general information on the museum and its services as well as some general information on the museum's collections and exhibitions. Registered users will have access to more details and services. Depending on their role they will have different access rights. The group of registered users can be further specialized into the following categories:

- *Registered general users* will be able to browse the exhibits of the museum's collections and exhibitions instead of receiving general information, only.
- *Academic users* like teachers and students can obtain even more information on the museum's exhibits for research purposes. They will be provided with dedicated materials and links to further information on the web, the museum's library, etc. They will also be allowed to contribute content in terms of making comments.
- *Curators* are in charge of managing collections, exhibitions, and exhibits.
- *Organizational managers* will use the application to administer marketing and communication materials like news, events, etc.
- *Administrators* are in general responsible for user management tasks and for maintaining the geopositioning infrastructure.

Moreover, the group of *Museum visitors* can be distinguished, which possibly overlaps with the other groups. For example, an academic user is considered a museum visitor when accessing the web application while visiting the museum.

[3] www.placelab.org

For designing UWAs, knowing the context properties to which the services of the UWA shall adapt is essential. The CAM web application will cover the following context properties:

- **User.** Customization will be done on the basis of the user's interest, the user's age as well as the user's role.
- **Location.** The web application will provide users with different contents and services according to the location from which it is accessed, e.g., inside or outside the museum.
- **Device.** Depending on the device used to access the CAM web application, users will be presented with a different interface.
- **Time.** The contents as well as their presentation will be delivered depending on the current time.
- **Weather.** Since the museum encompasses indoor as well as outdoor areas, the weather is considered in providing the user with customized content and services as well.

6.1.3 The Context-Aware Museum Web Application's Functional Requirements

Having provided a brief overview on the CAM web application, this section shall present the web application's functional requirements in detail. The WebML development process [CFB$^+$03] follows a user-driven approach of requirements elicitation. Consequently, the core as well as the customization functionality of the CAM web application shall be described on the basis of the different user groups, their use cases and the customization scenarios they participate in.

When identifying the user groups of a web application in the WebML development process, users are typically clustered according to their goals and behaviors and will be associated to distinct siteviews or areas which allow fulfilling the group's requirements. Amongst others, it is suggested to start by distinguishing between internal and external users and an administrative role. Thus, the main discriminator used is the user groups access rights to content and services of the web application. Concerning UWAs, however, the mere consideration of access rights is not enough, since they might depend on the context in which the user is accessing the web application. For instance, users might be provided with different content and services because of their current location, i.e., access rights may change because of the current context. Consequently, during the identification of user groups and their possible hierarchical relationships, the different contexts in which the UWA will be used need to be considered as well.

For the CAM web application, particular attention needs to be payed to the location from which the user requests the web application's services. As already mentioned, it will be accessible from different locations, i.e., from outside the museum boundaries as well as from the inside, via the new information terminals and wireless network infrastructure. The location context has an effect on the user groups, since the specific role of the user as well as her/his location has to be considered with respect to what information is to be made accessible. Figure 6.1, depicts the specialization hierarchy of the CAM user groups. The non-registered *Website User* is distinguished from all other user groups as being the only one exclusively accessing the web application from outside the museum boundaries. Thus, the *Website User* will have access to general information on the museum, its collections, exhibitions, events, opening hours, etc. Due to the possibility of accessing the CAM web application via the given infrastructure from within the museum, the

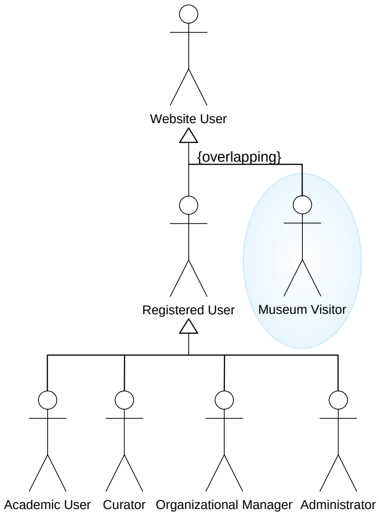


Figure 6.1: The Context-Aware Museum User Groups

Museum Visitors group is introduced and will be able to access more information on the exhibits. If using their own devices (i.e., a Laptop or PDA), they will be able to access the web application via the wireless network. Customization for Museum Visitors mainly will be based on their location context as information on the user is not available. *Registered Users* and their sub-groups need credentials for authentication. They will have access to the museums details also from outside the museum boundaries. The CAM web applications services will be additionally adapted according to their user profile, their interests etc. The Registered User and Museum Visitor groups are defined to be *overlapping* in order to denote that users at the same time can have the role of a Registered User and a Museum Visitor. Consequently, also the sub-types of Registered User will be able to have the role of a Museum Visitor. *Academic Users* such as teachers and students will have access to even more details on the exhibits for research purposes and the possibility to contribute comments. Finally, compared to the others, internal users such as *Curators*, *Organizational managers*, and *Administrators* will have content management access rights for their specific responsibilities and tasks. Thus, internal users typically are not provided with a customized view of the web application. In the CAM web application, however, *Curators* are provided with a context-aware exhibit management and thus, are allowed to maintain the information on exhibits on site with their laptops or PDAs.

The different user groups and their access rights are summarized in the following *group description sheets*. We have altered the schema of WebML's group description sheets [CFB^{+}03] to explicitly capture customization scenarios of the CAM web application that support a specific user group. Therefore, besides relevant use cases, the *customization scenarios* in which the given user group is involved are listed and the usual *context* in which the user group is accessing the web application are described.

Group name	**Website User**
Description	External visitor interested in accessing general content published on the website.
Profile data	No profile required. Users do not need credentials for authentication.
Relevant use cases	Browse News, Browse Collections, Browse Collection's Highlights, Browse Exhibitions, Browse Exhibition's Highlights, Browse Events, Browse Buildings, Browse Rooms, Browse Outdoor Areas, Browse Guided Tours, Access Museum Information
Context	Website users access the museum web application from outside the museum boundaries.
Customization scenarios	**Season's Style:** The presentation style of the web application changes due to the current season. **Multi-Delivery:** Depending on the device used to access the web application, the user is redirected to an appropriate siteview. **Current News and Upcoming Events**: At the home page, users will be informed about the museum's current news and upcoming events.
Objects accessed in read-only mode	Collections, Events, Exhibitions, News, Exhibits (Highlights), Buildings, Rooms, Museum Areas (Outdoor)
Objects accessed in content management mode	None.

Group name	**Museum Visitor** inherits from *Website User*
Description	Museum visitors are actually visiting the museum and potentially accessing the web application for further information on the exhibits via the museum's information terminals or via their proprietary devices (laptop, PDA). If using their own devices, they will receive a temporary account to access the museum's wireless network.
Profile data	from Website User
Relevant use cases	from Website User
Context	Museum visitors access the web application from within the museum boundaries. Their location, i.e., the museum area, needs to be temporarily stored.
Customization scenarios	**Context-Dependent Access to Information:** When browsing a collection or exhibition, the users will have access to all exhibits being part of the collection or exhibition, respectively, instead of having access to a selection of the highlights, only. Likewise, when browsing the museum's outdoor areas, buildings, and rooms, users will also have access to the room's areas and exhibits. **Special Exhibits Recommendation:** Depending on the time and weather, users are recommended special indoor and outdoor exhibits. **Exhibits in Vicinity:** Depending on the user's location, the exhibits in the user's vicinity will be displayed. **Location-Aware Tour:** The Location-Aware Tour is a separate area dedicated to present information of the current location, only. E.g., if the

	user is located near an exhibit, information on the exhibit will be displayed. Otherwise, the user will be presented information on the current room.
Objects accessed in read-only mode	Collections, Events, Exhibitions, News, Exhibits, Buildings, Rooms, Museum Areas
Objects accessed in content management mode	None.

Group name	**Registered User** inherits from *Website User*
Description	Registered users are regular customers of the museum having acquired an account allowing them to gain access to exhibits of the museum's collections and exhibitions.
Profile data	Title, First Name, Last Name, E-Mail, Phone, Fax, Street, Number, Zip Code, City, Country, Login, Password, Birthday, Age Class.
Relevant use cases	Login, Logout, Manage Profile
Context	Registered Users access the CAM web application from outside the museum's boundaries as well as from inside taking the role of a *Museum Visitor*.
Customization scenarios	**In the role of a Museum Visitor:** Special Exhibits Recommendation, Exhibits in Vicinity, and Location-Aware Tour **Context-Dependent Access to Information:** If successfully logged in to the web application and browsing a collection or exhibition, the users will have access to all exhibits being part of the collection or exhibition, respectively, instead of having access to a selection of the highlights, only. Likewise, when browsing the museum's outdoor areas, buildings, and rooms, users will also have access to the room's areas and exhibits.
Objects accessed in read-only mode	Collections, Events, Exhibitions, Exhibits, News, Buildings, Rooms, Museum Areas
Objects accessed in content management mode	Profile data.

Group name	**Academic User** inherits from *Registered User*
Description	Academic users such as teachers and students will have full access to all scientific information on collections, exhibitions, and exhibits.
Profile data	Institution Name, Logo, URL.
Relevant use cases	Access Collection Research Material, Access Exhibition Research Material, Access Exhibit Research Material, Add Collection Comment, Add Exhibition Comment, Add Exhibit Comment
Context	from Registered User
Customization scenarios	from Registered User
Objects accessed in read-only mode	Collections, Events, Exhibitions, Exhibits, News

Objects accessed in content management mode	Comments

Group name	**Curator** inherits from *Registered User*
Description	Personnel in charge of maintaining the information on collections, exhibitions, and exhibits.
Profile data	from Registered User
Relevant use cases	Collection Management (Add, Modify, Delete Collection, Add Research Material to Collection), Exhibition Management (Add, Modify, Delete Exhibition, Add Research Material to Exhibition), Exhibit Management (Add, Modify, Delete Exhibit, Add Research Material to Exhibit, Add Artist to Exhibit), Category Management (Add, Modify, Delete Category)
Context	Users will typically access the web application from desktop PC's in their offices. For managing exhibits, however, users shall also be able to update information on an exhibit using a mobile device, e.g., when re-arranging some exhibits in a room.
Customization scenarios	**Context-Aware Exhibition Management:** Users will be displayed the nearby exhibits for which the content needs to be maintained. Depending on the device, the web application provides the user with an appropriate interface.
Objects accessed in read-only mode	Access Events related to Category, Access Prices related to Collections and Exhibitions
Objects accessed in content management mode	Categories, Collections, Exhibitions, Exhibits, Artist, Material, Profile data

Group name	**Organizational Manager** inherits from *Registered User*
Description	Internal personnel that will manage information related to new collections, exhibitions, and events as well as organizational information including opening hours, etc.
Profile data	from Registered User
Relevant use cases	Event Management (Add, Modify, Delete Events, Add Price), News Management (Add Modify, Delete News, Add Price) Add Price to Collection, Add Price to Exhibition
Context	Users will typically access the web application from desktop PC's in their offices.
Customization scenarios	None.
Objects accessed in read-only mode	Collections, Exhibitions, Exhibits
Objects accessed in content management mode	Events, News, Prices

Group name	**Administrator** inherits from *Registered User*
Description	Technical personnel in charge of managing application users and user groups as well as maintaining the museum's information terminals and wireless network infrastructure
Profile data	from Registered User
Relevant use cases	Typical content management tasks related to the below listed objects.
Context	Users will typically access the web application from desktop PC's in their offices.
Customization scenarios	None.
Objects accessed in read-only mode	Categories, Events, and Exhibits associated to Museum Areas
Objects accessed in content management mode	User, Groups, Modules, Building, Room, Museum Area, InfoTerminals, Museum IPs, User Agent, Device Type, User Location, Weather

6.2 Designing the Context-Aware Museum with WebML

In this section, parts of the CAM web application design models will be discussed, i.e., the content model and the hypertext model consisting of five siteviews. The *Public* siteview serves the Website User group, the Museum Visitor group, the Registered User group, and the Academic User group. The *Public_PDA* siteview basically represents the same content and services to the user but is specifically designed for PDA devices. And finally, the Curator User group the Organizational Manager group as well as the Administrator group will each be supported with a dedicated siteview to fulfill their specific content management tasks, i.e., *Curator* siteview, *Organization* siteview, and *Administrator* siteview.

In the following, the focus will be on presenting eight different customization scenarios, most of them affecting the *Public* siteview as well as the *Curator* siteview. In this respect, the subsequent sub-sections will present the content model, provide an overview on the *Public* siteview and the *Curator* siteview by graphically discussing the parts relevant to the set of customization scenarios.

6.2.1 Content Model

For modeling UWAs, the WebML approach suggests modeling context information in a *context model* [CDMF07]. The context model is not a separate model in the WebML language but can be seen as a sub-schema of the content model. More specifically, the approach provides developers with guidelines to modeling context information on the basis of several sub-schemata. Besides the *Context Sub-Schema*, the Entities of the content model can be grouped into the *Basic User Sub-Schema* and the *Personalization Sub-Schema*, which are typically overlapping. It is important to note, that the proposed sub-schemata are not modeling concepts of the WebML language but guidelines intended to help modelers in finding appropriate entities for modeling context information and how they should be connected with the application data [CDMF07]. These guidelines will be presented in the following before, a concrete example in terms of the CAM web application's content model together with its sub-schemata is given.

The *Basic User Sub-Schema* consists of the *User*, *Group*, and *Module* Entities. It is part of every WebML model and allows to associate users to user groups which have access to certain modules of the web application, i.e., Pages, Areas, and Siteviews. The Entities of this sub-schema allow for adaptation of a web application's services according to the user, the user's role and access rights. The *User* Entity furthermore represents a basic profile of a web application's users.

The *Personalization Sub-Schema* is used to specify relationships between the *User* entity and entities from the web application in order to express that the user ″owns″ or ″prefers″ instances of these entities. Possibly, new entities are introduced into the sub-schema to better express such relationships, e.g., an entity named 'Interest' could be introduced and store the interest level of the user with respect to other entities. This kind of relationship allows personalizing the content and services of a web application according to the user's identity.

The *Context Model Sub-Schema* stores information on the context in which the web application is accessed. For instance, it could include entities such as *Device* and *Location*. The entities of the context model are directly or indirectly connected to the *User* entity via relationships in order to associate the user with his/her current context.

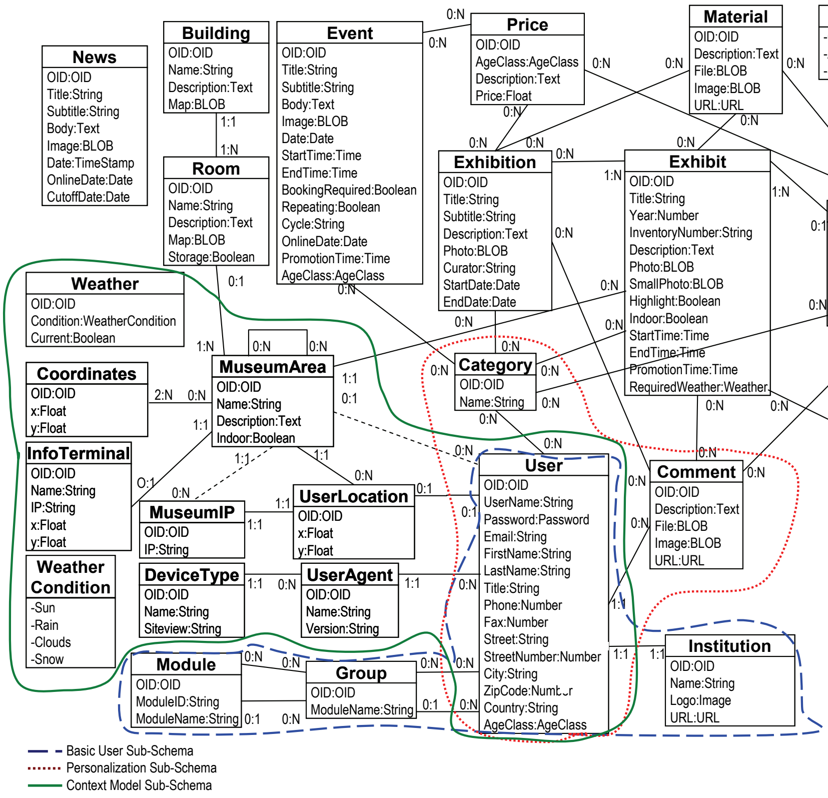


Figure 6.2: The Context-Aware Museum Content Model

In Figure 6.2, the content model of the CAM web application together with the three sub-schemata is presented. As can be seen, the core entities of the museum web application comprise *Exhibits* which are part of a *Collection* as well as can be part of a special *Exhibition*. All of these entities can be associated with *Comments* from academic users (cf. *User* entity) as well as with the *Material* entity, which allows storing additional information in terms of files, images, and links to external web sites. Furthermore, Exhibits can be associated with their *Artists*. The *News* entity typically is used to store the museum's news. For storing information on guided tours, workshops, and other events the *Event* entity is used. Events, Exhibitions, and Collections are also associated with the *Price* entity, allowing to define the prices for tickets of different price classes. Furthermore, information on the museum's *Buildings* and *Rooms* are stored.

The *Basic User Sub-Schema* including the *User*, *Group*, and *Module* entities also includes the *Institution* entity in order to be able to store information on the institution of academic users.

In the *Personalization Sub-Schema*, the entity *Category* is introduced and represents the categories a user is interested in. The necessary relationships from Category to Events, Exhibition, Exhibit, and Collection are included in order to be able to provide the user with a personalized view. Furthermore, the relationship to the *Comments* entity is declared to be part of the sub-schema, since it expresses that a user "owns" comments.

Finally, in the *Context Model Sub-Schema* information on the weather context, the device context, and the location context is stored. With respect to the *Weather*, general information on the current weather condition is of interest, only. The *condition* attribute's type is of the domain *WeatherCondition*. For the CAM web application it is assumed that the current weather (see *current* attribute) is updated by some external service. The *DeviceType* used will be derived from the *UserAgent*, i.e., the Web browser used to access the CAM web application. The UserAgent entity will store a list of browsers which are associated to a specific device type. For example, the *Minimo* or *Pocket Internet Explorer* browsers will be associated to a PDA device type, while *Firefox* or *Internet Explorer* browsers will be associated to a PC device type. Furthermore, in the DeviceType entity, the siteview to be used for the specific device type is stored. With respect to location context, several entities are relevant. Exhibits are to be displayed in certain places in the museum, i.e., in a *MuseumArea*. The *MuseumArea* entity is used to declare indoor as well as outdoor areas of the museum (see attribute *indoor*). Indoor areas typically are part of a Room. Moreover, the *Coordinates* entity is used to specify the size of the MuseumAreas, Rooms, and Buildings, and enables locating users in any of the museum's places. Users accessing the CAM web application from within the museum boundaries will need to install a browser plugin. This will allow determining their location based on the museum's wireless network infrastructure. At the server side, a user's location (cf. *UserLocation*) in terms of GPS coordinates will be calculated and the corresponding MuseumArea will be derived. The *MuseumIP* entity stores the IP range of the wireless network infrastructure and is used to find out if a user is accessing the CAM web application from outside the museum boundaries or is using the museum's wireless infrastructure. Users can also access the CAM web application from one of the museum's information terminals. The *InfoTerminal* entity holds the static IP address of the information terminal and is associated to the MuseumArea entity to denote the terminals location.

6.2.2 Public Siteview

The *Public* siteview is dedicated to all external user groups, i.e., Website User, Museum Visitor, Registered User, and Academic User. In Figure 6.3, an overview of the siteview is given. Users will find three public landmark pages in the *Public* siteview, while protected pages are indicated by the key icon. Public pages are accessible from the main menu of the CAM web application to all users. Amongst them, the *Home* page provides the user with general information and the possibility to log in to the application. The *News* page is specifically dedicated to the museum's news and will show a news list as well as a link to show the details for a news entry. On the *Info & Contact* page, information on opening hours, services, staff, etc. is displayed. The *Logout* page

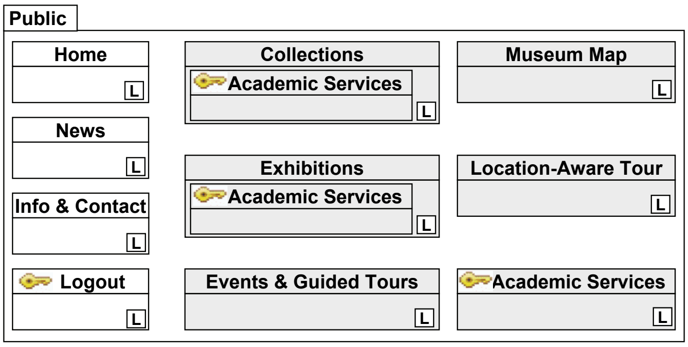


Figure 6.3: The *Public* Siteview: An Overview

is declared to be protected and will be shown to registered users that have successfully logged in to the web application, only.
Besides, the *Public* siteview is organized into five public landmark areas. The *Collections* area provides the user with access pages to the museums *Permanent Collections* and *Study Collections*. Details on a collection such as its exhibits highlights is given in the *Collection Details* page. The area also contains the sub-area *Academic Services*, which is accessible for the Academic User group, only, and provides more information on a collection and its exhibits for research purposes as well as allows users to post comments on the collection and exhibits, respectively. The *Exhibition* area is organized in the same way as the *Collections* area but provides users with information on exhibitions and their exhibits and also includes a sub-area dedicated to *Academic Users*. The *Events* area contains pages that give information on the museum's events, workshops, and guided tours, while the *Museum Map* area allows users browsing the museum's buildings, rooms and their areas, as well as outdoor areas. In the *Location-Aware Tour* area the respective customization scenario is realized, i.e., users will be presented information on rooms or exhibits depending on their current location. Finally, Academic Users will have a dedicated area to manage their profile as well as their comments in the *Academic Services* area. Following, this siteview's customization scenarios are presented.

6.2.2.1 Customization Scenario Multi-Delivery

The *Multi-Delivery* customization scenario allows users to be provided with a customized siteview according to the device they are using to access the CAM web application. Currently, users shall be supported with a separate siteview, i.e., the *Public* siteview, when accessing the application via a PC or laptop and be presented another siteview when accessing the application via a PDA, i.e., the *Public_PDA* siteview.

In order to provide the correct siteview, the web application needs to compute the current user's device type on the first page access. Therefore, this computation needs to be done for every possible page from the *Public* siteview the user might access. In WebML this is realized by defining the *Public* siteview to be context-aware (cf. C-label in Figure 6.4, denoting that a ContextUnit has been added to the siteview) and provide in the context cloud appropriate means for computing the user's device type. As depicted in Figure 6.4, in the context cloud, the user agent string identifying the browser is received via the GetClientParameterUnit *GetUserAgent*. According to this

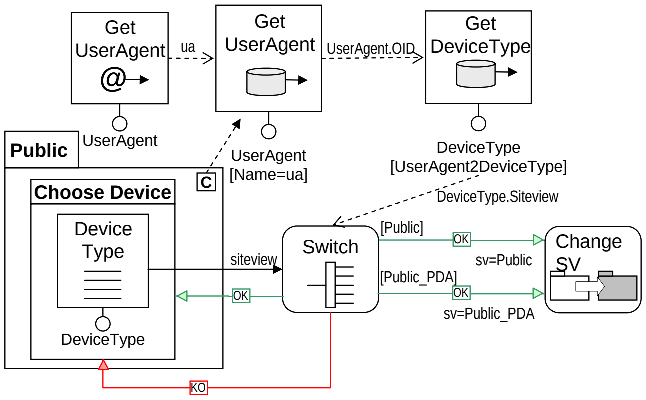


Figure 6.4: Customization Scenario *Multi-Delivery*

string, the corresponding user agent as well as the device type can be extracted from the application data with the GetDataUnit *GetUserAgent* and the GetDataUnit *GetDeviceType*, respectively. From the GetDataUnit *GetDeviceType* the corresponding siteview is sent to a SwitchUnit. In case the transported parameter *DeviceType.Siteview* resolves to *Public_PDA*, the ChangeSiteviewUnit is called in order to switch to the more suitable siteview for PDAs (cf. *sv=Public_PDA*). In case the parameter resolves to *Public*, the OKLink leads back to the ContextUnit of the *Public* siteview as no further adaptation is needed. Still, in the default case and on errors, the user is redirected to the *Choose Device* page in the siteview, which allows users to explicitly select an appropriate siteview.

6.2.2.2 Customization Scenario Season's Style

Similar to the *Multi-Delivery* scenario, the *Season's Style* scenario is to be applied to all pages of the *Public* siteview. The *Season's Style* scenario allows the presentation of the CAM web application to be adapted according to the time context, i.e., the current season. More specifically, the CAM web application shall have a different style in the summer season than in the winter season. Figure 6.5 depicts how this customization functionality can be modeled in WebML. Again, the *Public* siteview is declared to be context-aware having a link from the siteview's ContextUnit to the context-cloud, i.e., starting with the IfUnit in this case. In the example, it is assumed that the current date is received from the client. Consequently, a GetClientParameterUnit *GetDate* is used to feed an IfUnit with the current date. The IfUnit then evaluates the current date and computes which OKLink shall be followed to the ChangeStyleUnit. If the IfUnit's SelectorConditions evaluate to true, then the summer style is needed and the name of the CSS style sheet is transported via the OKLink as input to the ChangeStyleUnit (cf. *css=summer*). Otherwise the winter style is required. Furthermore, the *Season's Style* scenario shall also be applied to the *Public_PDA* siteview.

It has to be noted that in literature on customization with WebML [CDMF07] it has not explicitly been stated if a ContextUnit for a siteview, an area, or a page can have more than one outgoing link to separate context clouds being processed one after the other. Nevertheless, the computation algorithm for context-aware pages can be easily adapted to iterate over the outgoing links and perform the computations sequentially. Otherwise, the two customization scenarios *Multi-Delivery* and *Season's Style* would have to be merged - in contrast to the separation of concerns principle - into one context cloud. Still, the modeler needs to carefully design the order of the

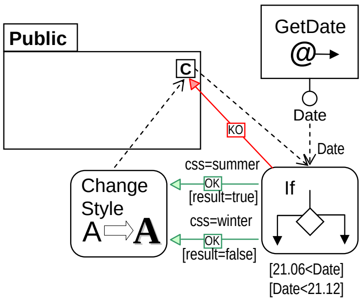


Figure 6.5: Customization Scenario *Season's Style*

ContextUnit's outgoing links. If the customization scenario changes the navigation flow, such as in the *Multi-Delivery* scenario, this might bypass important computations. Still, for the current scenarios the order is not important, since both apply to all pages of the *Public* siteview. This means, having specified the *Season's Style* scenario after the *Multi-Delivery* scenario, the outgoing link to determine the user's device type will be followed first in the computation of the context cloud. In case of a redirect of the user to the *Choose Device* page, the computation of the required style will be omitted for the accessed page but be performed for *Choose Device* page at last.

6.2.2.3 Customization Scenario Current News & Upcoming Events

In the *Current News & Upcoming Events* customization scenario, the *Home* page shall provide the user with a customized view on the museum news and events (cf. Figure 6.6). In the example, the current date and time are assumed to be provided by the client-side via the GetClientParameterUnits *GetDate* and *GetTime*. In the context cloud, the current date and time are used to set two GlobalParameters *CurrentDate* and *CurrentTime*, specifically introduced for this scenario. These GlobalParameters then can be used in the *Home* Page to select news and events according to the time context. A news item will be displayed if the current date is between the date when the news item shall go online (`OnlineDate <= curDate`) and the date when the news item shall go offline (`CutoffDate >= curDate`). Likewise, only today's events (`Date <= curDate`) are displayed if the current time is in between the time when the event shall be promoted (`PromotionTime <= curTime`) and the time when the event actually starts (`StartTime <= curTime`). In addition, if the current user is known, the events will be filtered according to the user's age class (`AgeClass <= User.AgeClass`). This last SelectorCondition is declared to be 'implied' and will be ignored if the age class of the user is not available.

6.2.2.4 Customization Scenario Location-Aware Tour

In the CAM web application, the *Location-Aware Tour* scenario is realized as a separate landmark area dedicated to present the user information on the current location, only. Thus if available, the exhibits of the current location are displayed. Otherwise, the user is presented with information on the room s/he is currently visiting or general information on the museum's buildings and outdoor areas in case the user is visiting an outdoor area.

Figure 6.7 illustrates how this scenario can be realized in WebML. Note that KOLinks have been omitted for readability purposes. First, the location of the user needs to be gathered, which

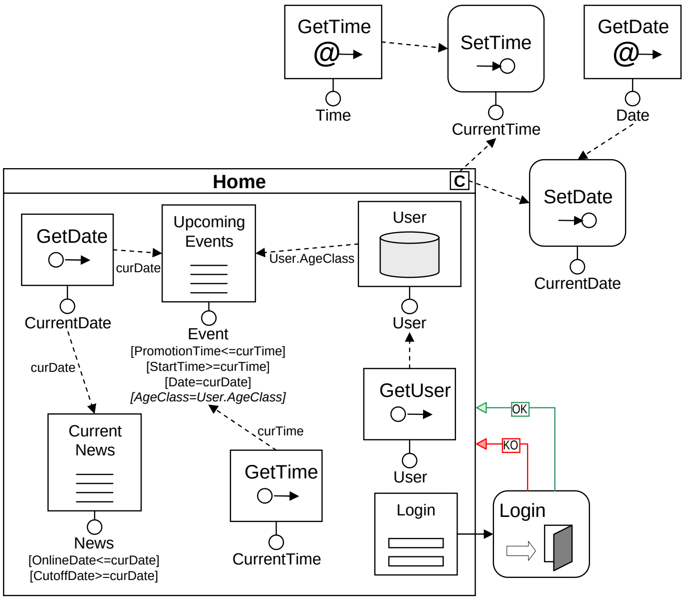


Figure 6.6: Customization Scenario *Current News & Upcoming Events*

is done in the context cloud of the area *Location-Aware Tour*. The IP of the user is gathered using a GetClientParameterUnit in order to find out if the user is accessing the CAM web application from one of the museum's information terminals (cf. *GetInfoTerminal* GetDataUnit) or from her/his own mobile device (cf. *GetMuseumIP* GetDataUnit). The location of the user can be determined by getting the corresponding museum area with the GetDataUnit *GetMuseumArea* for either an InfoTerminal or a MuseumIP (cf. SelectorConditions of *GetMuseumArea*). The museum area then is stored in the ContextUnit to be available for the area's contained context-aware pages. If the IP is neither contained in the set of information terminal IPs nor in the set of IPs for accessing the museum's wireless network, no museum area can be obtained for the user and stored in the ContextUnit. The contained context-aware pages will have to handle this case appropriately, as is explained next.

When a user accesses the *Location-Aware Tour* area, s/he will be displayed the *Default* page. More specifically, the *Default* page is context-aware and its context cloud checks if the location of the user is known, i.e., the *MuseumArea.OID* must not be `NULL`. If the user location is not known, the *Default* page is displayed and will inform the user, that the location-aware tour is not available due to missing location information. If the location is known, the exhibits for the current museum area are computed (cf. *GetExhibit* GetDataUnit). In case exhibits are available for the current museum area, the user is forwarded to the *Exhibit Details* page. Otherwise, it has to be determined if the current museum area is an outdoor area or an area of a room (cf. *GetRoom* GetDataUnit) in order to forward the user to the *Museum Map* page or the *Room Details* Page, respectively. As the user walks trough the museum, the current page will update itself according to the specified time interval for the ContextUnit and possibly redirect the user to another of the *Location-Aware Tour* area's pages.

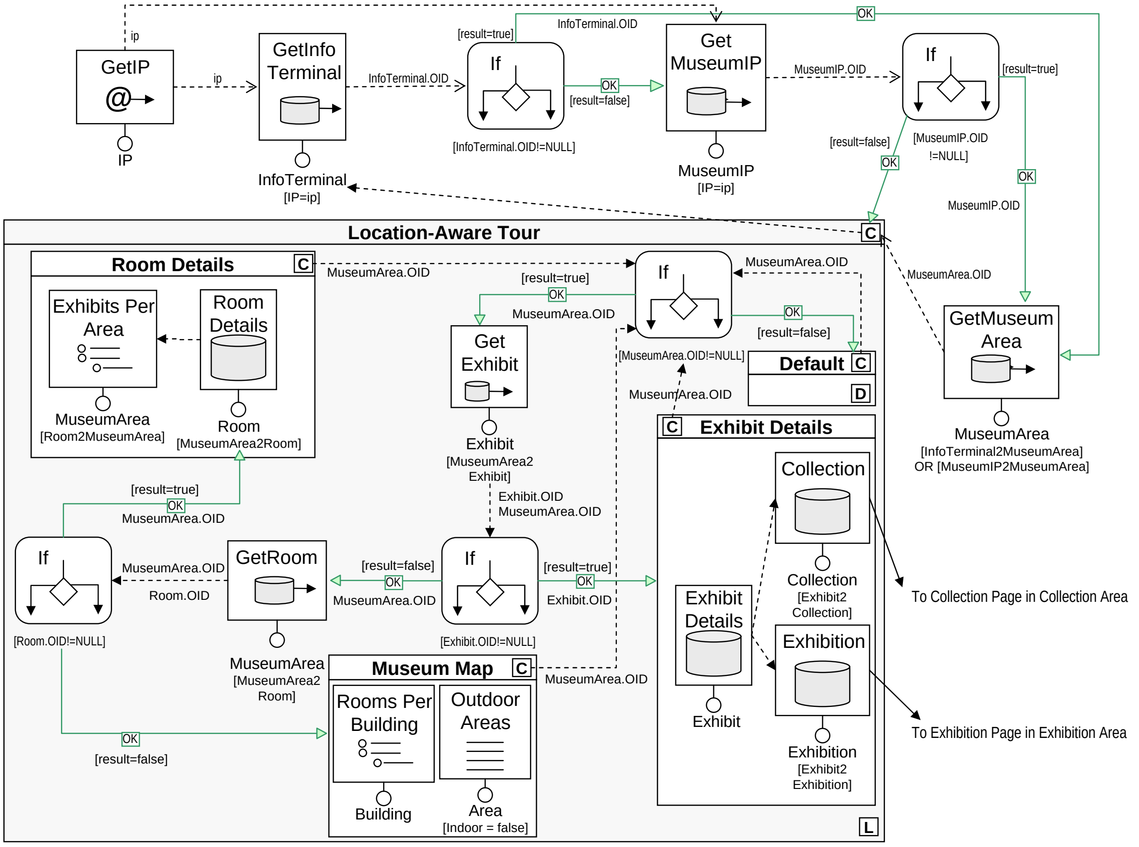


Figure 6.7: Customization Scenario *Location-Aware Tour*

6.2.2.5 Customization Scenario Context-Dependent Access to Information

In this scenario some information of the CAM web application is available to users either if they are located in the museum, i.e., the museum area they are currently in is known, or if they have successfully logged in to the web application. When viewing a collection or exhibition, these users will have access to all exhibits being part of the collection or exhibition, respectively, instead of having access to a selection of the highlights, only. Likewise, when browsing the museum's outdoor areas, buildings, and rooms, users will also have access to the room's details and its exhibits. Information on collections, exhibitions, as well as the museum's buildings, rooms, and areas is provided by three separate areas, namely the *Collections* area, the *Exhibitions* area, and the *Museum Map* area. Since according to the customization scenario, the location of the user needs to be available within each of the three areas, the areas need to be context-aware and gather the respective information for their contained pages. Analogous to the *Location-Aware Tour* customization scenario (cf. Section 6.2.2.4), each of the areas has to be extended with the functionality for computing the current museum area of the user via the user's IP. The context cloud to be attached to the *Collections* area (and others) is illustrated in Figure 6.8. The *MuseumArea.OID* then is available for the areas' contained context-aware pages via their ContextUnits. In the following, the scenario shall be discussed for the *Collections* area, only.

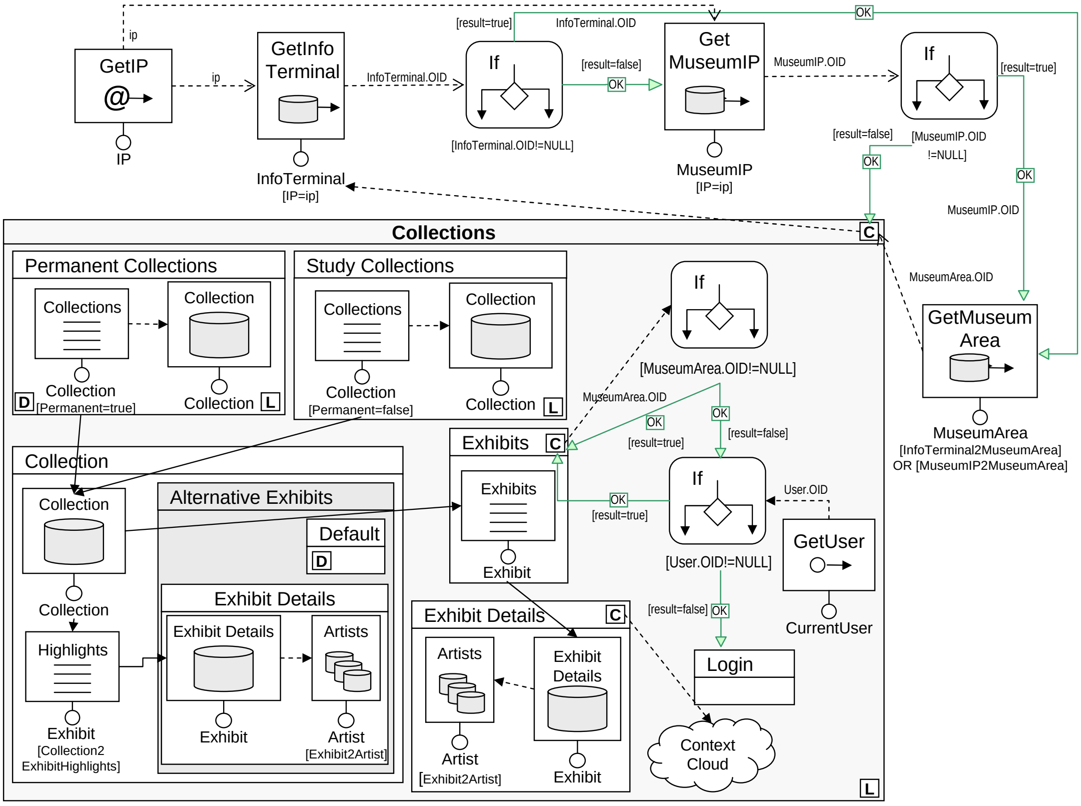


Figure 6.8: Customization Scenario *Context-Dependent Access to Information*

The *Collection* area contains information to be displayed to registered users and/or users accessing the CAM web application from within the museum. In Figure 6.8, parts of the *Collections* area are shown. When accessing the area, users are displayed per default the *Permanent Collections* page (cf. D-label). On the *Collection* page, the user gets information on the selected collection. In particular the user will be able to view the details of some of the collection's highlights but not for all of its exhibits (cf. IndexUnit *Highlights* and Alternative *Alternative Exhibits*). A separate link leads the user to the *Exhibits* page presenting an index to all exhibits and further to the *Exhibits Details* page. These two pages, however, are only accessible if the user has successfully logged in to the web application and/or accesses the application from within the museum. Consequently, both pages are designed to be context-aware. Figure 6.8 only shows the context cloud ensuring the above stated conditions for one of the pages, i.e., the *Exhibits* page. The MuseumArea.OID is provided to the IfUnit in the context cloud to find out if the user is currently located in the museum. If true, the user is allowed to view the page and the OKLink leads back to the page's ContextUnit. Otherwise, the next IfUnit checks if the current user is known. If true, the user is allowed to view the page and again the OKLink leads back to the page's ContextUnit. In case the current museum area and the current user are unknown, the user is redirected to the page *Login*, where s/he is able to log in to the application and is provided with information on how to download the browser plugin for geopositioning.

6.2.2.6 Customization Scenario Special Exhibits Recommendation

In the customization scenario *Special Exhibits Recommendation*, users are recommended special indoor and outdoor exhibits depending on the current time and weather, given that they access the CAM web application from within the museum boundaries, i.e., they are actually visiting the museum. For example some outdoor exhibits might be of interest at a certain time of day and/or under certain weather conditions, only. The CAM web application shall call the users' attention to these exhibits accordingly in a separate part of certain web pages. For the CAM web application, this information shall be added to the entry pages of the *Collections* and *Exhibitions* areas, i.e., the *Permanent Collections* page, the *Study Collections* page, and the *Exhibitions* page. Figure 6.9 shows how this customization scenario is added to the *Permanent Collections* page. For other pages the customization scenario can be attached likewise.

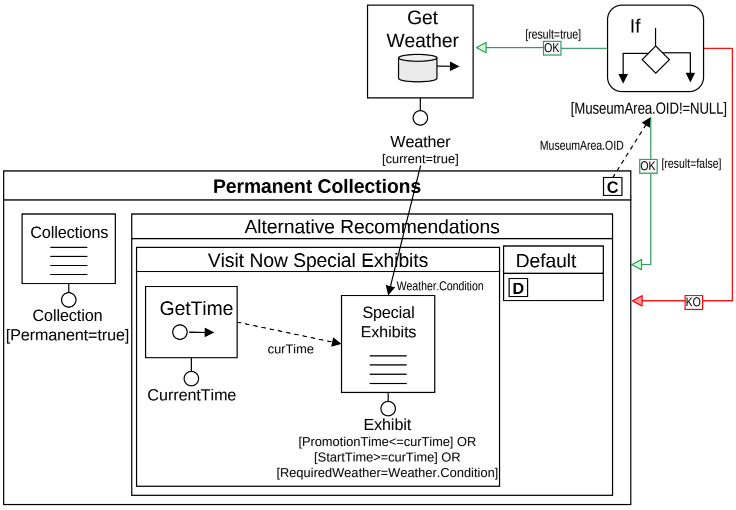


Figure 6.9: Customization Scenario *Special Exhibits Recommendation*

To the *Permanent Collections* page, which shows an index of all the museum's permanent collections, an Alternative *Recommendations* is added. The Alternative contains the *Visit New Special Exhibits* page and the *Default* page, the latter being displayed by default. The *Visit Now Special Exhibits* page shall be displayed, only, if the user accesses the CAM web application from within the museum boundaries. Therefore, the *Permanent Collections* page is made context-aware and in the context cloud the location of the user is checked. If the user's museum area is known (cf. IfUnit), the current weather information is collected with the GetDataUnit *GetWeather* and the *Weather.Condition* is transported to the SelectorConditions of the *Special Exhibits* IndexUnit. This will activate the *Visit Now Special Exhibits* page. Please note, that for this customization scenario to work, the museum area transported to the context cloud (cf. *MuseumArea.OID*) has to be available for the ContextUnit. This means, that the museum area has to be determined by some other context cloud of one of the page's container. In this case, the *Collections* area has already been made context-aware in the *Context-Dependent Access to Information* scenario in order to compute the current museum area (cf Section 6.2.2.5).

6.2.2.7 Customization Scenario Exhibits in Vicinity

The customization scenario *Exhibits in Vicinity* is similar to the *Special Exhibits Recommendation* scenario (cf. Section 6.2.2.6). Depending on the user's location, s/he will be provided with a list of nearby exhibits. The CAM web application shall call the users' attention to these exhibits accordingly in a separate part of certain web pages. For the CAM web application, this information shall be added to the pages providing details of a certain exhibit, i.e., the *Exhibit Details* pages in the *Collections* area and the *Exhibitions* area. Figure 6.10 shows how this customization scenario is added to the *Exhibit Details* page in the *Collections* Area. For other pages the customization scenario can be attached likewise.

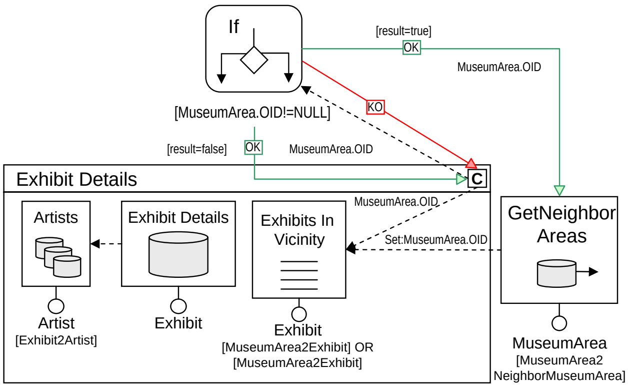


Figure 6.10: Customization Scenario *Exhibits in Vicinity*

To the *Exhibit Details* page, which shows information on the exhibit and its artists, an IndexUnit *Exhibits In Vicinity* is added. The IndexUnit shall display all exhibits of the current museum area as well as those of neighboring museum areas. Thus, the IndexUnit has two SelectorConditions, whereby one needs to be filled with the current museum area provided by the ContextUnit of the page and the other one with the set of neighboring museum areas provided by the context cloud using the *GetNeighborAreas* GetDataUnit. In case the museum area is not known, the neighboring museum areas cannot be calculated either (cf. IfUnit) and no exhibits will be displayed, since both SelectorConditions fail. Again, the museum area needs to be made available to the page's ContextUnit through some context-aware container of the *Exhibit Details* page. In this case, the *Collections* area has already been made context-aware in the *Context-Dependent Access to Information* scenario in order to compute the current museum area (cf Section 6.2.2.5). Please note, that in the customization scenario *Special Exhibits Recommendation*, the use of an Alternative was necessary, since some exhibits fulfilling the time constraints might have been displayed independently from the user's location. In contrast to this scenario, the GetUnits and IndexUnit could not be added to the page directly.

6.2.3 Curator Siteview

The *Curator* siteview is dedicated to the *Curator User* group. According to the group's use cases, the siteview provides the user with separate areas for the different content management tasks including the *Collection Management* area, the *Exhibition Management* area, the *Exhibit Management* area, and the *Category Management* area. In the following scenario, the *Exhibit Management* area shall be made context-aware with respect to the location and device context. Figure 6.11*(a)* shows a simplified version of the original *Exhibit Management* area.

6.2.3.1 Customization Scenario Context-Aware Exhibit Management

In the *Context-Aware Exhibit Management* customization scenario, users will be displayed the nearby exhibits for which the content needs to be maintained. Furthermore, depending on the device, the web application provides the user with an appropriate interface, i.e., for PDAs the *Exhibits* page (cf. Figure 6.11*(a)*) shall be split into two pages, whereby in the first one, the index of exhibits and in the second one, the actual form for modifying a single exhibit shall be displayed.

In Figure 6.11*(b)* the *Exhibits* page is made context-aware. In the context cloud, the device type used to access the web application is determined in a similar way as in the *Multi-Delivery* customization scenario (cf. Section 6.2.2.1). The *GetUserAgent* GetDataUnit retrieves the UserAgent using as input the information provided by the GetClientParameterUnit, then the device type is computed using the *GetDeviceType* GetDataUnit. In case the device used is a PDA (cf. IfUnit), the user is redirected to the *Exhibits* page of the Alternative *Device-Independence*. The *Exhibits* page contains the IndexUnit *Exhibits*, only, allowing the user to select one exhibit to be modified on an extra page, i.e., the *Modify Exhibits* page. If the user accesses the application using a laptop, per default the *Modify Exhibits* page of the Alternative *Device-Independence* is displayed, which shows a list of exhibits and a form for modifying one of them on the same page. This page actually represents the original Exhibits page in Figure 6.11*(a)*.

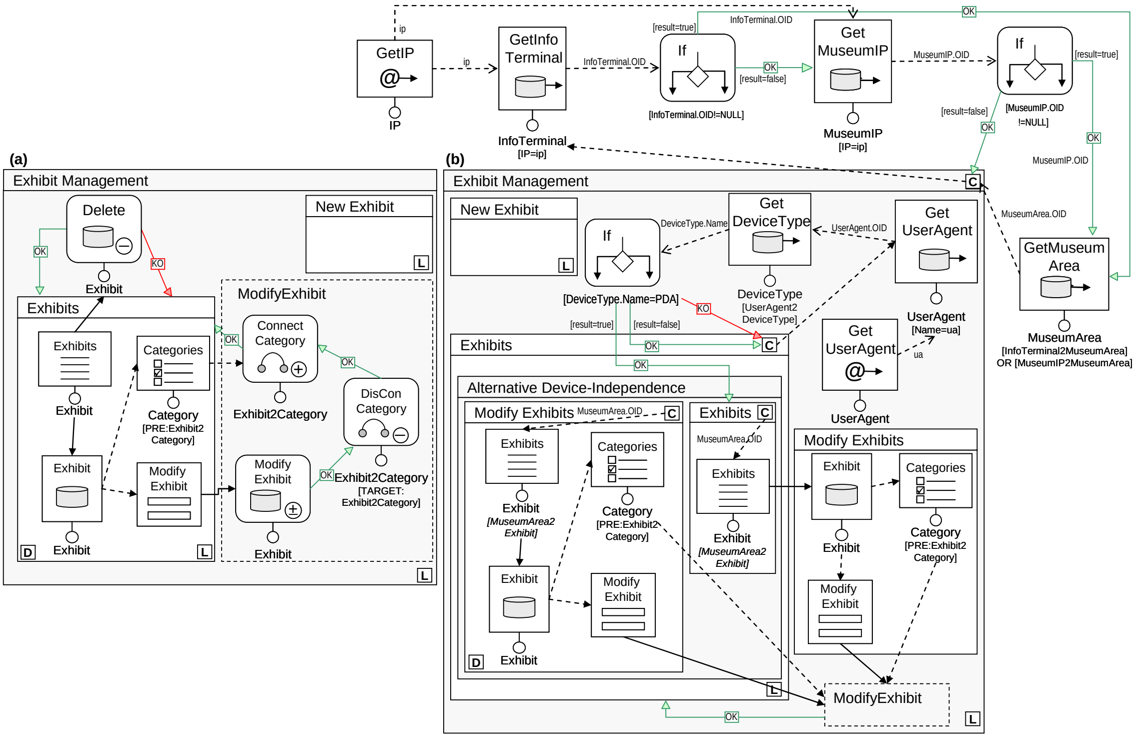


Figure 6.11: Customization Scenario *Context-Aware Exhibit Management*

In order to allow for location-awareness, the *ModifyExhibits* page and the *Exhibits* page in the Alternative are declared context-aware and a SelectorCondition, which filters the exhibits according to the current museum area, is added to both *Exhibits* IndexUnits. The SelectorConditions are filled with the *MuseumArea.OID* from the pages ContextUnits. Furthermore, they are defined to be implied in case the current museum area is not known.

Again, the museum area needs to be made available to the page's ContextUnit through some context-aware container of the pages. Consequently, the *Exhibit Management* area is declared to be context-aware by adding a ContextUnit. In the context cloud the user's current museum area is determined in the same way as for the *Context-Dependent Access to Information* customization scenario (cf Section 6.2.2.5) so that it is available for its contained context-aware pages.

6.3 Designing the Context-Aware Museum with aspectWebML

While the previous sections were dedicated to WebML and how the set of customization scenarios can be realized, this section will explain how the customization functionality, which is scattered across the whole CAM web application, can be encapsulated within Aspects using the aspectWebML approach.

Following the aspectWebML guidelines to be presented in Section 6.3.4, instead of having one huge *Customization* Aspect, each customization scenario will be discussed independent from the others in a first step. This means that each scenario will be treated as a separate Aspect to be composed with the core museum web application without considering the other scenarios' Aspects. For each scenario, the composition of the Aspect with the core museum web application will be specified in terms of a set of AsymmetricCompositionRules. An explanation of what modeling elements of the scenario represent the core functionality of the museum web application and what modeling elements represent the customization functionality will be provided. This explanation will be given when extracting the customization functionality from the models presented for the WebML solution in Section 6.2 into several Advice of the Aspect. Moreover, besides the advantage of having separated the customization functionality with an Aspect, further advantages of the aspectWebML approach shall be pointed out for each scenario.

The configuration of the Aspects in order to be used in combination will be done in a second step, as will be explained in Section 6.3.4. After having fully specified the scenarios, the modeler is able to identify redundancies in multiple pieces of Advice across the Aspects and reflect on reorganizing these Advice within meaningful, independent, and possibly reusable Aspects.

Before, the content model of the museum web application is presented in Section 6.3.1, the parts representing the core functionality and customization functionality, respectively, are pointed out as well as the advantage of using aspectWebML to designing a context model. Subsequently, the discussion of the aspectWebML version of the customization scenarios located in the Public siteview (cf. Section 6.3.2) and the Curator siteview (cf. Section 6.3.3) is given.

6.3.1 The Content Model and the Context Model

As already mentioned before, customization requires the extension of the content model to store context information, meaning that for each customization scenario, the necessary extensions to the content model for storing context information have to be considered. In Figure 6.12, the content model of the CAM web application is presented again. This time, however, the presentation

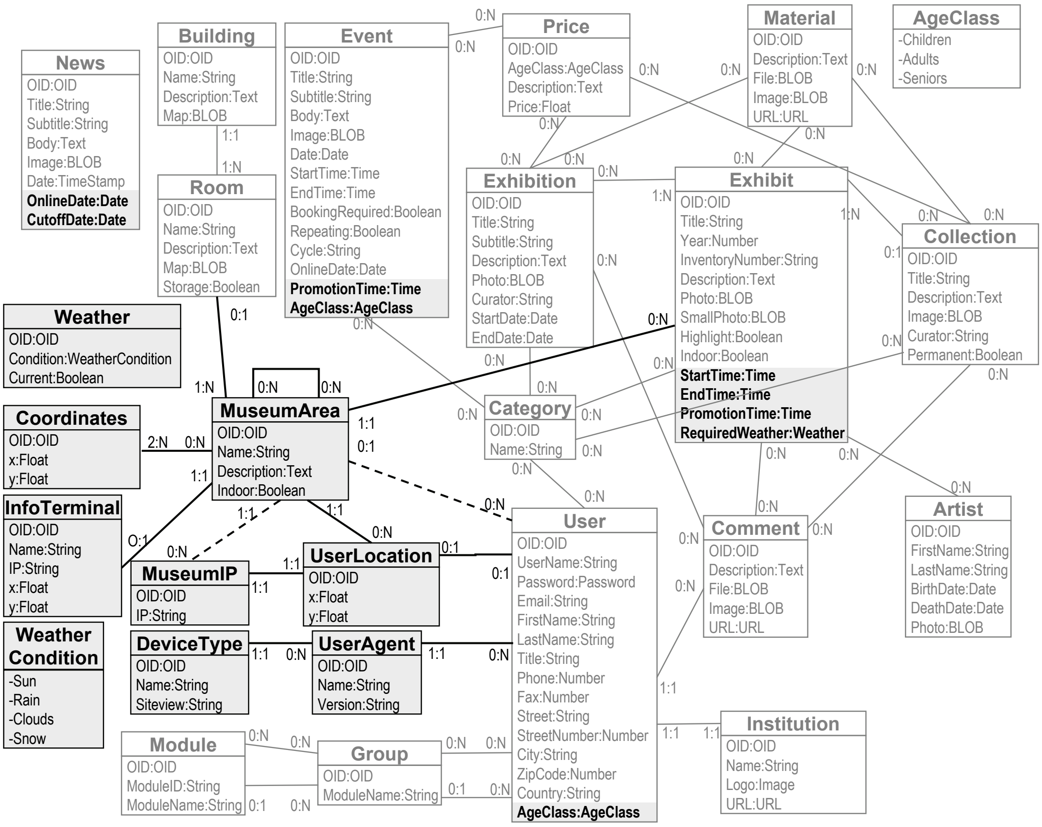


Figure 6.12: The Context-Aware Museum Context Model

of the core content model, i.e., the content model of the original museum web application before its extension with customization functionality, is alleviated in light-grey, while the Entities, Relationships, as well as Attributes contributing to customization are emphasized in bold.

Unlike WebML's way of only visualizing context information in terms of sub-schemata in the content model, with the aspectWebML approach it is possible to really separate the context information from the content model. For example, the Entities, the Relationships as well as the Attributes can be extracted from the content model and encapsulated within several pieces of Advice of a *Context Model* Aspect. In this respect it has to be emphasized that, in contrast to the WebML approach, aspectWebML even allows to keep separate Attributes that need to be introduced to Entities of the content model. For example, the *Exhibit* Entity, needs to be extended with the *RequiredWeather* Attribute so that the Weather information of the context model can be exploited.

Furthermore, in the aspectWebML approach, the User, Group, and Module Entities are considered part of the core web application, since they are the starting entities in every WebML model. The core museum web application, will support non-registered as well as registered users, including academic users, curators, etc. with their respective tasks but without considering context information. Consequently, the Entities Category, Comment, and Institution are also considered part of the core museum web application, unlike in the WebML approach.

In the following descriptions of the customization scenarios, the context information to be introduced to the core museum web application will be considered for each customization scenario separately.

6.3.2 The Public Siteview

Using the aspectWebML language, each customization scenario presented in Section 6.2 can be modeled separately from the core museum web application. As already discussed in Chapter 5, each scenario typically requires several pieces of Advice. For composition they are subsequently associated with an appropriate Pointcut in an AsymmetricCompositionRule. According to the language's metamodel, the different modeling elements to be introduced to the WebML model in a customization scenario have different containers. Consequently, for extracting the customization functionality from the WebML models presented in the previous Section 6.2, each customization scenarios will be analyzed and modeling elements having the same container will be collected within separate pieces of Advice. The Advice of one scenario will be grouped within one Aspect.

For composition purposes, the Advice will need to be applied to appropriate Pointcuts which is specified within a set of AsymmetricCompositionRules. Therefore, each customization scenario will be explained as a sequence of AsymmetricCompositionRules each consisting of an Advice and a Pointcut following the notation introduced in Section 5.2.

6.3.2.1 Customization Scenario Multi-Delivery

In order to realize this customization scenario, the necessary context information needs to be stored in the content model. This means that the *UserAgent* and *DeviceType* entities need to be added to the content model first. Second, when considering the hypertext level for the customization scenario *Multi-Delivery* in Figure 6.13*(a)*, everything but the *Public* siteview (highlighted in grey) represents customization functionality. Thus, besides making the *Public* siteview context-aware by extending it with a ContextUnit and an appropriate context cloud for retrieving the current device type used, the *Choose Device* page is added to the *Public* siteview.

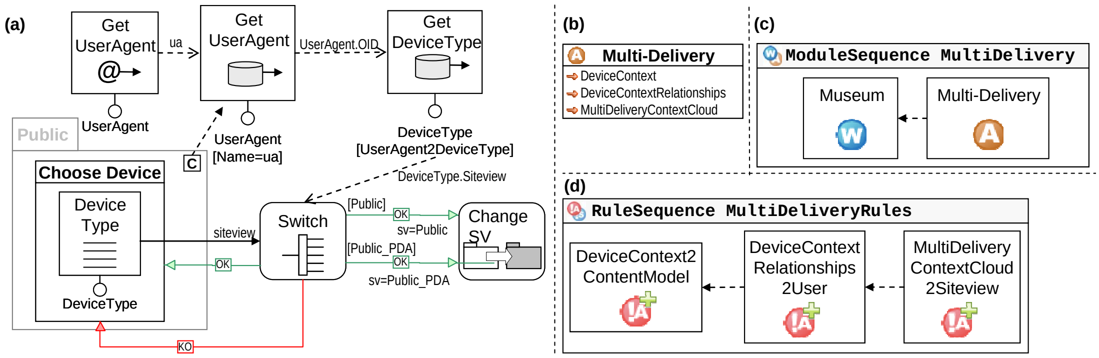


Figure 6.13: Customization Scenario *Multi-Delivery* in aspectWebML (1)

The *Multi-Delivery* Aspect for the scenario is shown in Figure 6.13*(b)* in terms of the Aspect Diagram notation introduced in Chapter 5.2. Figure 6.13*(c)* illustrates the necessary Concern-ModuleSequence for composing the Aspect with the core *Museum* WebML model, while in Figure

6.13*(d)*, the ConcernCompositionRuleSequence is depicted. For the following scenarios, the Aspect Diagram, the Module Sequence Diagram, and the Rule Sequence Diagram will be omitted for readability reasons. Instead, all information will be captured with the detailed notation for AsymmetricCompositionRules, where the necessary AsymmetricCompositionRules combining an Advice of the scenario's Aspect with an appropriate Pointcut are listed in the order of their application during composition (cf Figure 6.14). Again please note that, in the absence of a notational element for a WebML modeling element, the abstract syntax in the style of UML object diagrams is used.

In the Advice *DeviceContext* of the first rule, the *DeviceType* and *UserAgent* entities as well as the Relationship from the *UserAgent* entity to the *User* entity are added to the content model (cf. Pointcut *ContentModel*). The *User* entity is highlighted in grey to indicate that the target of the Relationship is located in the core content model and is not part of the Advice. In the second AsymmetricCompositionRule *DeviceContextRelationships2User* the Advice *DeviceContextRelationships* contains the inverse Relationship to be introduced to the *User* entity (cf. Pointcut *User*). Finally, the third rule *MultiDeliveryContextCloud2Siteview* adds the *Choose Device* page, the ContextUnit and elements from the context cloud, i.e., GetClientParameterUnit, GetDataUnits, SwitchUnit, and ChangeSiteviewUnit, to the *Public* siteview.

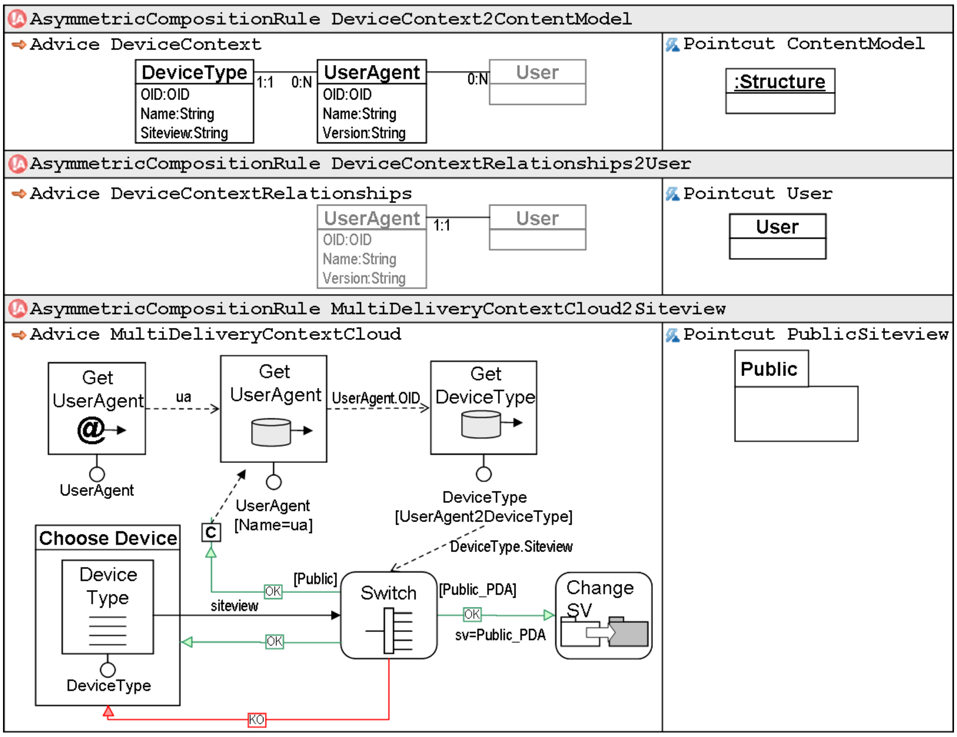


Figure 6.14: Customization Scenario *Multi-Delivery* in aspectWebML (2)

The advantages of the aspectWebML solution compared to the WebML solution of the customization scenario *Multi-Delivery* can be summarized as follows:

- Modeling elements contributing to customization are made explicite, since they are separated from the rest of the web application model and captured within an Aspect.

- The aspectWebML solution requires less modeling effort, in that the same extension of a siteview can be realized by changing a pointcut. For example, imagine the CAM web application shall be extended to also support mobile phones. This means, that a dedicated siteview for mobile phones will have to be extended with the customization functionality defined in Advice *MultiDeliveryContextCloud* as well. This can be done by changing the *ExternalUserSiteviews* Pointcut to also include a *Public_Mobile* siteview.

- In order to support a *Public_Mobile* siteview, the context cloud will also need to be changed, i.e., a further OKLink pointing to the ChangeSiteviewUnit will need to be inserted to the SwitchUnit. In the WebML solution, this change needs to be performed for every context cloud, i.e., for those of the *Public* siteview as well as the *Public_PDA* siteview. In contrast, in aspectWebML the change needs to be performed only once in the Aspect, or more specifically, in the *MultiDeliveryContextCloud* Advice.

- In case user acceptability of a device type is too low, aspectWebML allows for removing the respective customization functionality from the CAM web application in an easy way.

6.3.2.2 Customization Scenario Season's Style

When considering the WebML solution for the customization scenario *Season's Style* in Figure 6.5, everything but the *Public* siteview represents customization functionality.

The *Season's Style* customization scenario can be realized in aspectWebML within a single AsymmetricCompositionRule. Figure 6.15 shows that the Advice *SeasonsStyleContextCloud* contains the ContextUnit together with the context cloud specifying the necessary logic to decide on which stylesheet to use. The Advice can be easily applied to any siteview. As is specified in the Pointcut *ExternalUserSiteviews*, for the CAM web application the *Season's Style* scenario shall be applied to the siteviews designed for external users, i.e., the *Public* siteview as well as the *Public_PDA* siteview designed to support small device types.

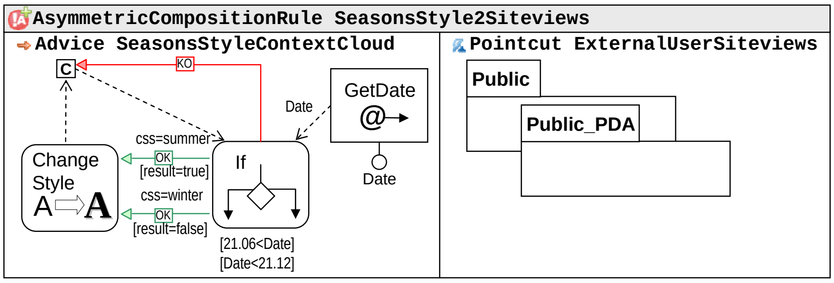


Figure 6.15: Customization Scenario *Season's Style* in aspectWebML

Again, the advantages of the aspectWebML solution lie in the separation of the customization functionality. The Advice can be easily reused for other siteviews by changing the Pointcut of the AsymmetricCompositionRule accordingly. Furthermore, changes to the customization functionality are localized within the Aspect, i.e., the Advice, allowing for better maintainability. For example, the Advice might be changed in order to also support special style sheets for the spring season and the fall seasons as well as for certain holidays.

6.3.2.3 Customization Scenario Current News & Upcoming Events

In the WebML solution, the original *Home* page contained an EntryUnit allowing users to log in to the web application. The remaining modeling elements depicted in Figure 6.6 represent customization functionality including the context cloud as well as the IndexUnits providing information on the current news and events. In the following, an explanation of how these modeling elements can be captured within an Aspect is given.

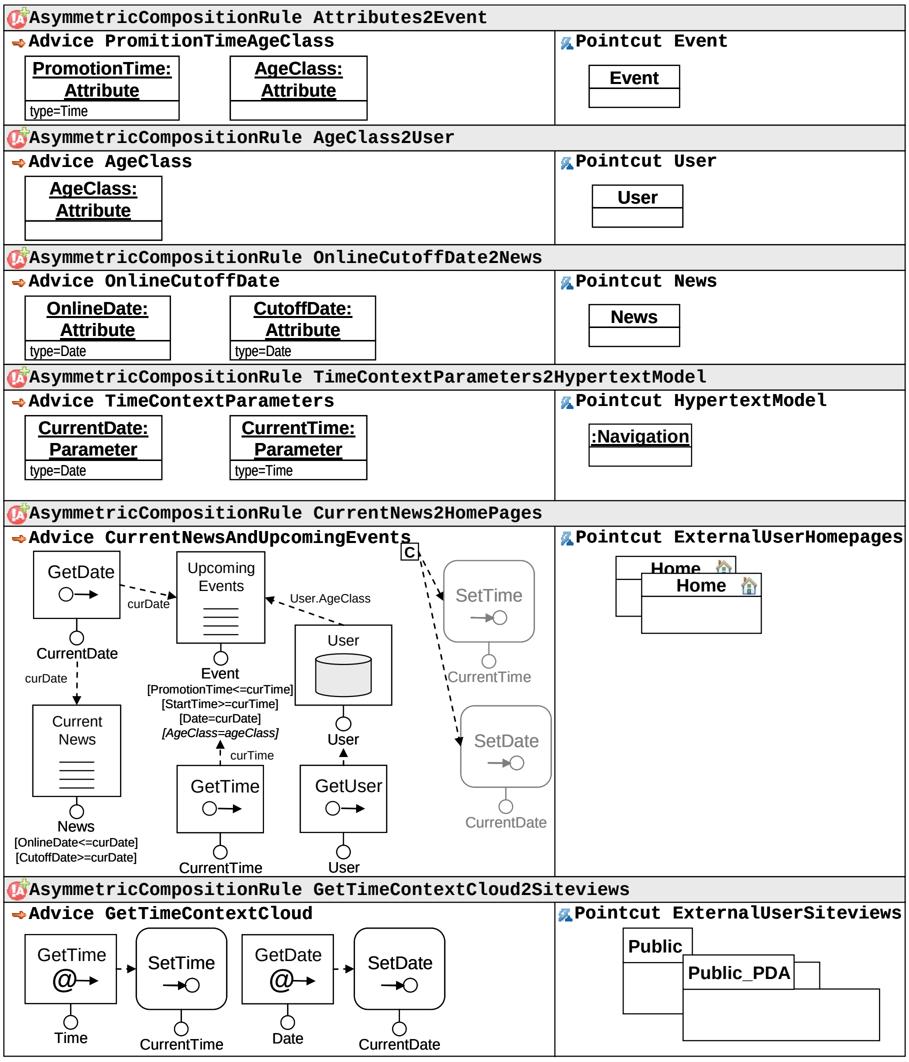


Figure 6.16: Customization Scenario *Current News & Upcoming Events* in aspectWebML

For the *Current News & Upcoming Events* customization scenario, some extensions to the content model are required before being able to model customization functionality at the hypertext level. As illustrated in Figure 6.16, the User, News, and Event entities need to be extended with some Attributes: The *Attributes2Event* AsymmetricCompositionRule adds the Attributes *PromotionTime* and *AgeClass* to the Entity Event. An *AgeClass* Attribute is furthermore added to the User Entity by means of the *AgeClass2User* rule. And the *OnlineDate* and *CutoffDate* Attributes are added to the News entity in rule *OnlineCutoffDate2News*.

Furthermore, two global Parameters have to be created prior to using them in SetUnits and GetUnits. In the *TimeContextParameters2HypertextModel* AsymmetricCompositionRule, Parameters for storing the current time and the current date are added to the WebML hypertext model which serves as the container for all siteviews. In the following rule *CurrentNews2HomePages*, the IndexUnits displaying the news items and events, the GetUnits for retrieving the current date, time, and user, as well as the ContextUnit are part of the rule's Advice and shall be added to the home pages specified in the Pointcut *ExternalUserHomepages*. The transport Links of the ContextUnit point to the "existing" SetUnits of the context cloud to be introduced next, which consequently are highlighted in grey. Finally, from the context cloud, the GetClientParameterUnits and the SetUnits for storing the current date and time are added next, which is specified in the *GetTimeContextCloud2Siteviews* AsymmetricCompositionRule. More specifically, they need to be added to the container of the *Home* page. Since the *Current News & Upcoming Events* scenario shall be applied to the *Home* pages of the *Public* and the *Public_PDA* siteviews, the Pointcut *ExternalUserHomepages* references them as join points. Please note, that for a correct composition the current implementation of the composition algorithm requires a ContextUnit always to be applied before its corresponding context cloud.

6.3.2.4 Customization Scenario Location-Aware Tour

The *Location-Aware Tour* scenario is designed as a new functionality of the museum web application and modeled as a whole new area. More specifically, the *Location-Aware Tour* Area is a new landmark area to be available to users from every point in the web application (cf. Figure 6.7). This new functionality shall be available no matter what kind of device is used to access the web application. Consequently, the *Location-Aware Tour* Area shall be introduced to both the *Public* and the *Public_PDA* siteviews. Thereby, the pages of the *Location-Aware Tour* area are designed such that they are suitable for both, PCs and PDAs.

In this scenario, location context information is required. Consequently, the core content model needs to be extended with the required entities (cf. Figure 6.17). Five new entities, i.e., *MuseumArea*, *Coordinates*, *MuseumIP*, *InfoTerminal*, and *UserLocation*, are introduced in the Advice *LocationContext*, their purpose having already been explained in Section 6.2.1. For some of the entities' relationships pointing to entities from the core content model (highlighted in grey) the inverse relationships need to be introduced to the respective entities in the core content model as well (cf. rules *LocationContextRelationships2User*, *LocationContextRelationships2Room*, *LocationContextRelationships2Exhibit*). As for the *Location-Aware Tour* Area, it can be directly introduced to the siteviews defined in the Pointcut *ExternalUserSiteviews* together with its contained pages and its context cloud. Nevertheless, for reusability purposes, the ContextUnit and the context cloud are specified in two separate pieces of Advice which are then used in the AsymmetricCompositionRules *LocationContextUnit2Area* and *GetLocationContextCloud2Siteviews*.

Again, the customization scenario can be modeled separately and composed with the rest of the museum web application model at the respective places defined within the Pointcuts. If required, the *Location-Aware Tour* Aspect allows the functionality to be "turned on and off". For example, the *Location-Aware Tour* Area can be composed with the *Admin* siteview and the *Curator* siteview instead of the public siteviews. This way, administrators and curators are able to test the web application in case changes to the (wireless) network infrastructure have been made.

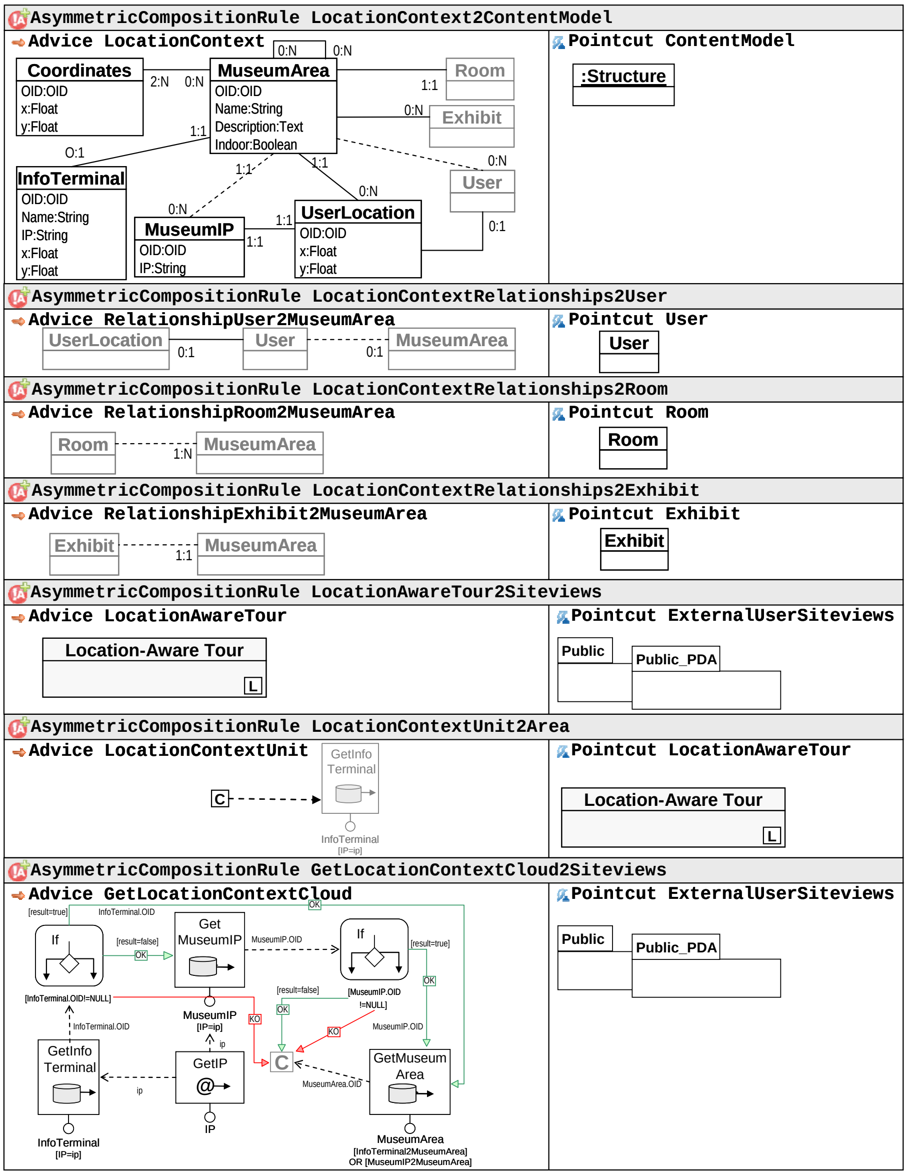
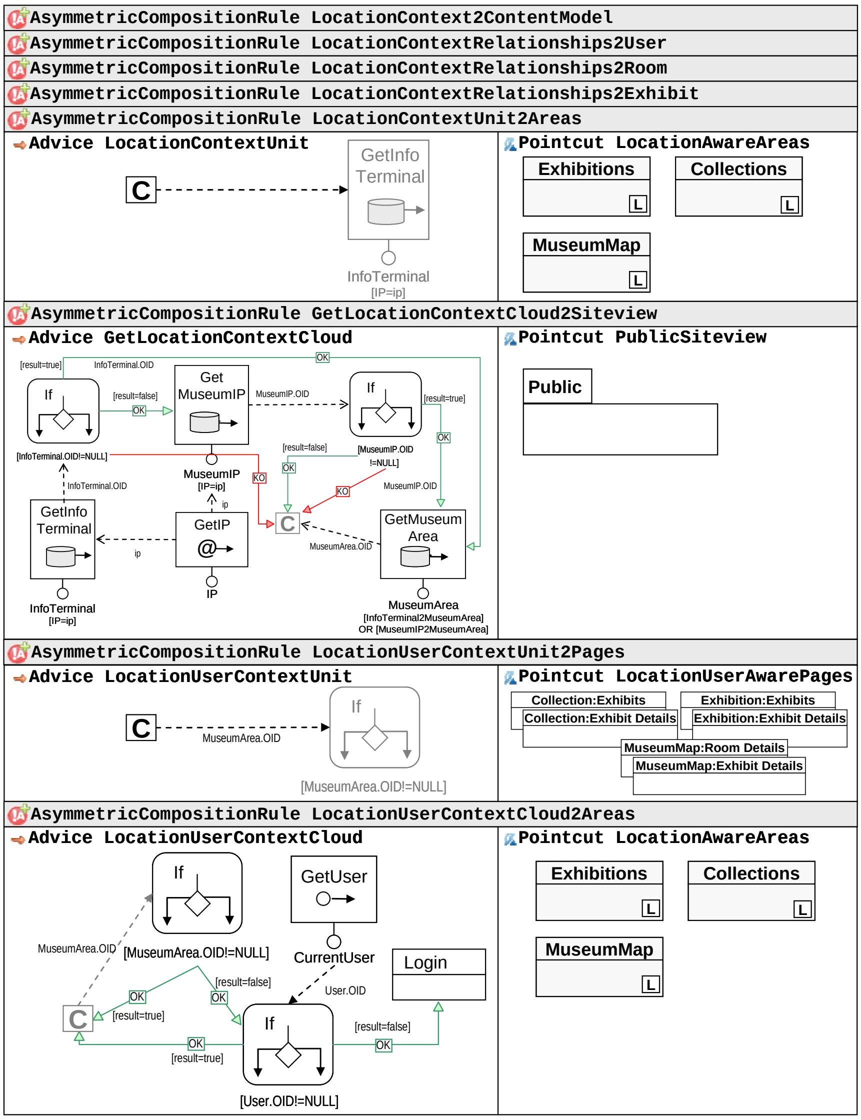


Figure 6.17: Customization Scenario *Location-Aware Tour* in aspectWebML

6.3.2.5 Customization Scenario Context-Dependent Access to Information

In the *Context-Dependent Access to Information* scenario the users are able to access certain pages if either their location is known or they are registered users and have successfully logged in to the web application. Similar to the *Location-Aware Tour* scenario, the necessary context information needs to be stored in the content model. Since each scenario is currently considered independent from the others, in this scenario the previously defined rules *LocationContext2ContentModel*, *LocationContextRelationships2User*, *LocationContextRelationships2Room*, and *LocationContextRelationships-2Exhibit* also need to be modeled. This means that the rules' Advice are modeled as part of the scenario's Aspect (cf. customization scenario *Location-Aware Tour* in Section 6.3.2.4). In Figure 6.18, however, the Advice and Pointcut of these rules will not be depicted for brevity reasons.

The customization functionality in this scenario is represented by the two types of context clouds depicted in Figure 6.8, i.e., one for the *Collections* Area to calculate the current museum area as well as another one for the *Exhibits* Page ensuring the access restrictions. In the aspectWebML solution, these context clouds shall be modeled separately within an Aspect and be applied to several Areas and pages, respectively, as will be specified by means of a set of AsymmetricCompositionRules in Figure 6.18.



AsymmetricCompositionRule LocationContext2ContentModel
AsymmetricCompositionRule LocationContextRelationships2User
AsymmetricCompositionRule LocationContextRelationships2Room
AsymmetricCompositionRule LocationContextRelationships2Exhibit
AsymmetricCompositionRule LocationContextUnit2Areas
Advice LocationContextUnit
C
GetInfo Terminal
InfoTerminal
[IP=ip]
Pointcut LocationAwareAreas
Exhibitions
Collections
MuseumMap
L
AsymmetricCompositionRule GetLocationContextCloud2Siteview
Advice GetLocationContextCloud
[result=true]
InfoTerminal.OID
OK
If
[result=false]
Get MuseumIP
MuseumIP.OID
[result=true]
[InfoTerminal.OID!=NULL]
KO
[result=false]
[MuseumIP.OID !=NULL]
InfoTerminal.OID
MuseumIP
[IP=ip]
ip
MuseumIP.OID
GetInfo Terminal
GetIP
@
C
MuseumArea.OID
GetMuseum Area
ip
InfoTerminal
[IP=ip]
IP
MuseumArea
[InfoTerminal2MuseumArea]
OR [MuseumIP2MuseumArea]
Pointcut PublicSiteview
Public
AsymmetricCompositionRule LocationUserContextUnit2Pages
Advice LocationUserContextUnit
C
MuseumArea.OID
If
[MuseumArea.OID!=NULL]
Pointcut LocationUserAwarePages
Collection:Exhibits
Collection:Exhibit Details
Exhibition:Exhibits
Exhibition:Exhibit Details
MuseumMap:Room Details
MuseumMap:Exhibit Details
AsymmetricCompositionRule LocationUserContextCloud2Areas
Advice LocationUserContextCloud
If
GetUser
MuseumArea.OID
[MuseumArea.OID!=NULL]
CurrentUser
Login
[result=false]
OK
User.OID
C
[result=true]
If
[result=false]
OK
[result=true]
[User.OID!=NULL]
Pointcut LocationAwareAreas
Exhibitions
Collections
MuseumMap
L

Figure 6.18: Customization Scenario *Context-Dependent Access to Information* in aspectWebML

In order to allow the context clouds of the "advised" pages to know the current museum area, an extension of the areas which contain the pages to be context-aware is necessary. These areas will be extended with a context cloud that calculates the required information. Therefore, the AsymmetricCompositionRule *LocationContextUnit2Areas* adds a ContextUnit to the *Collections*, *Exhibitions*, as well as *Museum Map* areas, and the AsymmetricCompositionRule *GetLocationContextCloud2Siteview* adds the necessary context clouds to the *Public* siteview. Then the pages to be "protected" in the three areas need to be made context-aware as well. Therefore, a ContextUnit is also added to the pages defined in the Pointcut *LocationUserAwarePages* of rule *LocationUserContextUnit2Pages*. Furthermore, for each page the context cloud, which checks if either the current museum area or the current user is known, needs to be included in the *Collections*, *Exhibitions*, as well as *Museum Map* areas (cf. AsymmetricCompositionRule *LocationUserContextCloud2Areas*).

For this scenario, a peculiarity of the composition algorithm supporting the aspectWebML language needs to be mentioned. In the *Context-Dependent Access to Information* scenario, presented in the previous section in Figure 6.8, the WebML solution suggests, that pages of the same container, need to have their own context cloud. If we allowed the ContextUnit of page *Exhibit Details* to point to the context cloud of the page *Exhibits* then, every time the user accesses the *Exhibits Details* page s/he would be redirected to the *Exhibits* page. Instead, the *Exhibits Details* page is required to have a context cloud on its own. Having a look at the corresponding AsymmetricCompositionRule *LocationUserContextCloud2Areas* in the aspectWebML solution, it seems that the context cloud is added to the areas defined in the Pointcut *LocationAwareAreas* only once. The composition semantics of the aspectWebML language, or rather the implementation of the composition algorithm, ensures that indeed a separate context cloud is introduced for each context unit. Details on implementation issues can be found in [Tom07].

Concerning maintenance, this example particularly points out the necessity of separating the customization functionality from the rest of the web application model. The context cloud for restricting the access to a page already needs to be introduced to six different pages as is specified in the Pointcut *LocationAwareUserPages*. Imagine the case where the context cloud needs to be extended for further restricting the access to a page according to the current time. For example, access to all exhibits and their details might be given for a short period of time before a new exhibitions starts. Having separated the context cloud within an Aspect, such extensions can be easily made in aspectWebML in one place, while they are automatically propagated to all necessary places in the web application model through composition.

6.3.2.6 Customization Scenario Special Exhibits Recommendation

Considering the WebML solution of the *Special Exhibits Recommendation* customization scenario depicted in Figure 6.9. The customization functionality is represented by the Alternative *Recommendations* as well as the context cloud associated with the *Permanent Collections* Page. The aspectWebML way of modeling the customization scenario *Special Exhibits Recommendation* is depicted in Figure 6.19. The scenario requires a global Parameter to be introduced in order to store the current time (cf. AsymmetricCompositionRule *TimeContextParameter2HypertextModel*) as well as the content model to be extended with the *Weather* entity and the Domain *WeatherCondition* (cf. AsymmetricCompositionRule *WeatherContext2ContentModel*). Furthermore, in the rule *Attributes2Exhibit*, the *Exhibit* entity needs to be extended with four Attributes allowing to filter Exhibits according to the current time and weather context, namely *StartTime*, *PromotionTime*, *EndTime*, and *RequiredWeather*.

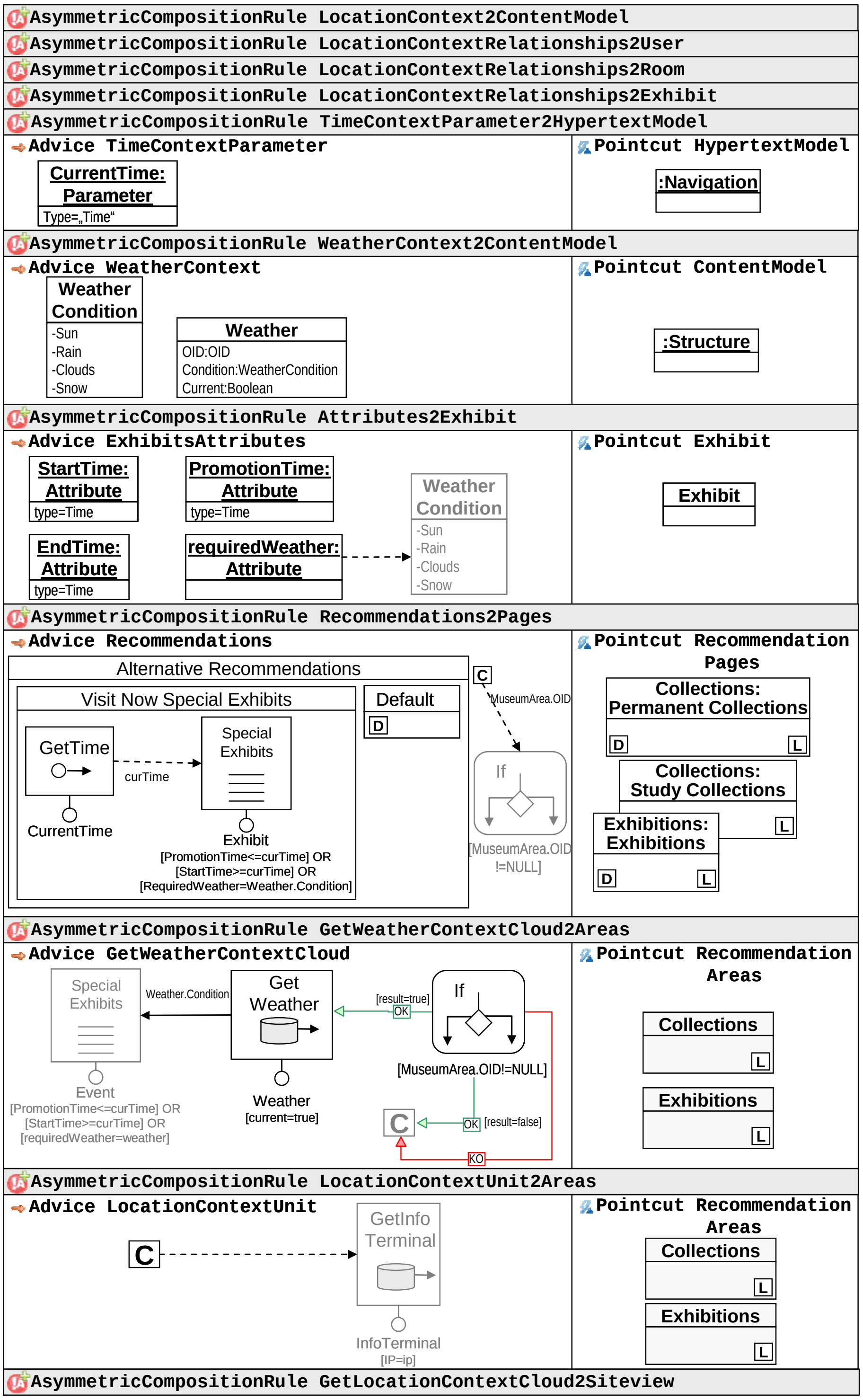

AsymmetricCompositionRule LocationContext2ContentModel
AsymmetricCompositionRule LocationContextRelationships2User
AsymmetricCompositionRule LocationContextRelationships2Room
AsymmetricCompositionRule LocationContextRelationships2Exhibit
AsymmetricCompositionRule TimeContextParameter2HypertextModel
Advice TimeContextParameter
CurrentTime: Parameter
Type=„Time“
Pointcut HypertextModel
:Navigation
AsymmetricCompositionRule WeatherContext2ContentModel
Advice WeatherContext
Weather Condition
-Sun
-Rain
-Clouds
-Snow
Weather
OID:OID
Condition:WeatherCondition
Current:Boolean
Pointcut ContentModel
:Structure
AsymmetricCompositionRule Attributes2Exhibit
Advice ExhibitsAttributes
StartTime: Attribute
type=Time
PromotionTime: Attribute
type=Time
EndTime: Attribute
type=Time
requiredWeather: Attribute
Weather Condition
-Sun
-Rain
-Clouds
-Snow
Pointcut Exhibit
Exhibit
AsymmetricCompositionRule Recommendations2Pages
Advice Recommendations
Alternative Recommendations
Visit Now Special Exhibits
GetTime
curTime
CurrentTime
Special Exhibits
Exhibit
[PromotionTime<=curTime] OR
[StartTime>=curTime] OR
[RequiredWeather=Weather.Condition]
Default
D
C
MuseumArea.OID
If
[MuseumArea.OID
!=NULL]
Pointcut Recommendation Pages
Collections: Permanent Collections
D
L
Collections: Study Collections
L
Exhibitions: Exhibitions
D
L
AsymmetricCompositionRule GetWeatherContextCloud2Areas
Advice GetWeatherContextCloud
Special Exhibits
Event
[PromotionTime<=curTime] OR
[StartTime>=curTime] OR
[requiredWeather=weather]
Weather.Condition
Get Weather
Weather
[current=true]
[result=true]
OK
If
[MuseumArea.OID!=NULL]
C
OK
[result=false]
KO
Pointcut Recommendation Areas
Collections
L
Exhibitions
L
AsymmetricCompositionRule LocationContextUnit2Areas
Advice LocationContextUnit
C
GetInfo Terminal
InfoTerminal
[IP=ip]
Pointcut Recommendation Areas
Collections
L
Exhibitions
L
AsymmetricCompositionRule GetLocationContextCloud2Siteview

Figure 6.19: Customization Scenario *Special Exhibits Recommendation* in aspectWebML

Similar to the *Location-Aware Tour* scenario, the necessary location context information needs to be stored in the content model. The scenario therefore also needs the previously defined rules *LocationContext2ContentModel*, *LocationContextRelationships2User*, *LocationContextRelationships2Room*, and *LocationContextRelationships2Exhibit* (cf. customization scenario *Location-Aware Tour* in Section 6.3.2.4).

In order to present special recommendations on exhibits to be visited according to the current time and/or weather context, the Alternative *Alternative Recommendations* needs to be introduced to the pages *Permanent Collections* and *Study Collections* of the *Collection* area as well as to the *Exhibitions* page of the *Exhibitions* area. Furthermore, the pages need to be declared context-aware as is specified in the AsymmetricCompositionRule *Recommendations2Pages*. In the *GetWeatherContextCloud2Areas* rule, then for each page the necessary context cloud for determining whether the user is accessing the application from within the museum as well as for querying the current weather condition is inserted into the respective area of the page (cf. Advice *GetWeatherContextCloud*). Again the composition algorithm ensures that each page is endowed with its own context cloud. As in the *Context-Dependent Access to Information* scenario, for making the areas of the affected pages location-aware so that the current museum area is available for the contained context-aware pages, ContextUnits need to be added to the *Collections* area and the *Exhibitions* area (cf. rule *LocationContextUnit2RecommendationAreas*). Furthermore, the context clouds need to be introduced to the siteview of the areas (cf. AsymmetricCompositionRule *GetLocationContextCloud2Siteview*). For brevity reasons, the Advice and Pointcut of the *GetLocationContextCloud2Siteview* rule are not detailed (cf. the *Context-Dependent Access to Information* scenario). Please note, that since being applied to the same Pointcut *RecommendationAreas*, the Advice *GetWeatherContextCloud* and *LocationContextUnit* could be merged into one Advice. For reuse purposes they are defined separately, however. For example, the *GetWeatherContextCloud* Advice could be reused in other situations, where the weather context is required.

In contrast to the given example, which considers recommendations on three distinct pages, recommendations are often given in many places of web applications. For example, in the Amazon web application, recommendations are given in a dedicated section of almost every page. Again, having separated the recommendations functionality in a separate Aspect, is can be easily changed in one place for the whole web application or even omitted at all.

6.3.2.7 Customization Scenario Exhibits in Vicinity

Similar to the *Special Exhibits Recommendation* scenario, considering the WebML solution of the *Exhibits in Vicinity* scenario depicted in Figure 6.10, the customization functionality is represented by the *Exhibits in Vicinity* IndexUnit as well as the context cloud associated with the *Exhibit Details* Page.

The *Exhibits in Vicinity* scenario is based on location information. Similar to the *Location-Aware Tour* scenario, the necessary location context information needs to be stored at the content level. The scenario therefore also requires the customization functionality already defined in previous rules, i.e., *LocationContext2ContentModel*, *LocationContextRelationships2User*, *LocationContextRelationships2Room*, and *LocationContextRelationships2Exhibit*, which are not detailed for reasons of brevity (cf. customization scenario *Location-Aware Tour* in Section 6.3.2.4).

As illustrated in Figure 6.20, in this scenario, information on nearby exhibits shall be added to the *Exhibit Details* pages of the *Collections* and *Exhibitions* areas as is described in rule *ExhibitsInVicinity2Page*. More specifically, the required IndexUnit as well as the ContextUnit are added to

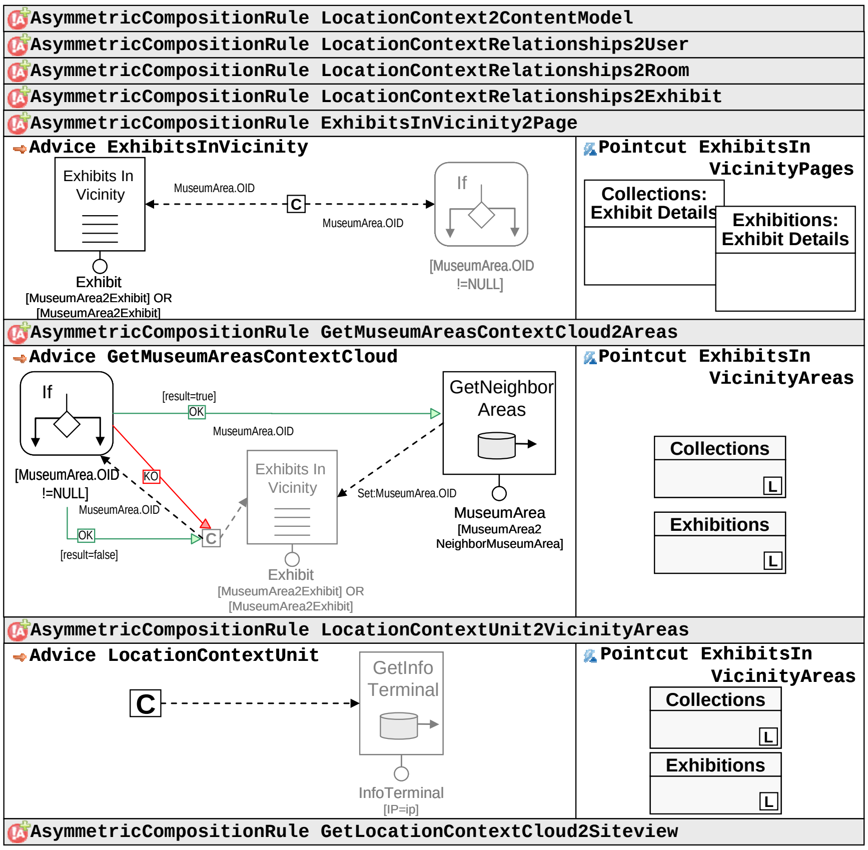

AsymmetricCompositionRule LocationContext2ContentModel
AsymmetricCompositionRule LocationContextRelationships2User
AsymmetricCompositionRule LocationContextRelationships2Room
AsymmetricCompositionRule LocationContextRelationships2Exhibit
AsymmetricCompositionRule ExhibitsInVicinity2Page
Advice ExhibitsInVicinity
Exhibits In Vicinity
MuseumArea.OID
C
MuseumArea.OID
If
[MuseumArea.OID !=NULL]
Exhibit
[MuseumArea2Exhibit] OR [MuseumArea2Exhibit]
Pointcut ExhibitsIn VicinityPages
Collections: Exhibit Details
Exhibitions: Exhibit Details
AsymmetricCompositionRule GetMuseumAreasContextCloud2Areas
Advice GetMuseumAreasContextCloud
If
[result=true]
OK
MuseumArea.OID
GetNeighbor Areas
[MuseumArea.OID !=NULL]
KO
MuseumArea.OID
OK
[result=false]
C
Exhibits In Vicinity
Set:MuseumArea.OID
MuseumArea
[MuseumArea2 NeighborMuseumArea]
Exhibit
[MuseumArea2Exhibit] OR [MuseumArea2Exhibit]
Pointcut ExhibitsIn VicinityAreas
Collections
L
Exhibitions
L
AsymmetricCompositionRule LocationContextUnit2VicinityAreas
Advice LocationContextUnit
C
GetInfo Terminal
InfoTerminal
[IP=ip]
Pointcut ExhibitsIn VicinityAreas
Collections
L
Exhibitions
L
AsymmetricCompositionRule GetLocationContextCloud2Siteview

Figure 6.20: Customization Scenario *Exhibits in Vicinity* in aspectWebML

the specified pages. In a second step, the pages' context clouds need to be introduced as well. In the *GetMuseumAreasContextCloud2Areas* AsymmetricCompositionRule, the context clouds are specified to be added to the *Collections* and *Exhibitions* area.

Again, as in the *Context-Dependent Access to Information* scenario, for making the areas of the affected pages location-aware so that the current museum area is available for the contained context-aware pages, ContextUnits need to be added to the *Collections* area and the *Exhibitions* area (cf. rule *LocationContextUnit2VicinityAreas*). Furthermore, the context clouds need to be inserted to the siteview (cf. rule *GetLocationContextCloud2Siteview* reused from the *Context-Dependent Information* scenario).

Please note, that since being applied to the same Pointcut *ExhibitsInVicinityAreas*, the two the Advice *GetMuseumAreasContextCloud* and *LocationContextUnit*, could be merged into one Advice. For reuse purposes they are defined separately, however. Currently, the scenarios are considered independent from each other and thus, the *GetMuseumAreasContextCloud* Advice has already been defined in several scenarios. When using the scenarios in combination, however, this Advice can be reused in several AsymmetricCompositionRules.

6.3.3 The Curator Siteview

6.3.3.1 Customization Scenario Context-Aware Exhibit Management

In Figure 6.11, the necessary customization functionality extensions have already been explained: the *Exhibits* page needs to be extended with a context cloud for computing the current device used as well as an Alternative presenting a page with the form for editing an exhibit together with the list of all exhibits or a page with a list of exhibits, only. In the latter case, the editing form will be presented on a separate page, which also needs to be introduced to the *Exhibits Management* Area.

In the *Context-Aware Exhibit Management* customization scenario, context information with respect to device and location context needs to be stored in the context model. In Figure 6.21, the previously defined rules from the Multi-Delivery scenario (*DeviceContext2ContentModel*, *DeviceContextRelationships2User*) as well as those from the *Location-Aware Tour* scenario (*LocationContext2ContentModel*, *LocationContextRelationships2User*, *LocationContextRelationships2Room*, and *LocationContextRelationships2Exhibit*) are not detailed.

In the *SelectorCondition2Index* AsymmetricCompositionRule, to the *Exhibits* IndexUnit a SelectorCondition is added, allowing to filter the exhibits according to the current museum area. Then the *ModifyExhibitsSubPage* Advice is applied *around* the *ModifyExhibitsContentUnit* Pointcut, i.e., the units of the *Exhibits* Page in Figure 6.11*(a)*. This means the *Modify Exhibits* Page is added around the specified units as a sub-page of the *Exhibits* Page. The *Modify Exhibits* Page to be used for PDAs is then introduced to the *Exhibit Management* area by means of the rule *ModifyExhibitsPage2Area*.

In order to add the Alternative *Device-Independence*, the rule *DeviceAlternativeAround2Page* applies its *Advice* around the *Modify Exhibits* page previously defined in the Advice *ModifyExhibitsSubPage*. Then the *Exhibits* Page needs to be made context-aware, which is done in the *DeviceContextUnit2Page* AsymmetricCompositionRule, and the necessary context cloud needs to be added. In rule *DeviceContextCloud2Area*, the context cloud responsible for computing the current device and changing the navigation flow accordingly is added to the *Exhibit Management* area.

Last but not least, the *Exhibit Management* area itself is declared context-aware in the rule *LocationContextUnit2ExhibitArea*. In order to gather the current museum area which shall be used by contained context-aware pages, the necessary context cloud (already defined in previous scenarios with the Advice *GetLocationContextCloud* Advice) has to be added to the *Curator* Siteview by means of the rule *GetLocationContextCloud2CuratorSiteview*.

The customization functionality introduced with this scenario is specifically tailored to the exhibit management area of the curator siteview. In this respect, the major advantage of the aspectWebML solution is the possibility of keeping the necessary changes to the core museum web application separate within an Aspect. This allows for traceability of the evolution of the exhibit management core functionality. Furthermore, if user acceptance of the context-aware exhibit management is low, the web application model can be easily ”rolled back” to the original core functionality. Likewise, it might turn out, that curators use their laptops for managing exhibits in the museum rooms, only. For performance reasons, hence, the context-aware exhibit management functionality might be changed to only support laptops and not PDAs. Again, this change is localized within the scenario’s Aspect.

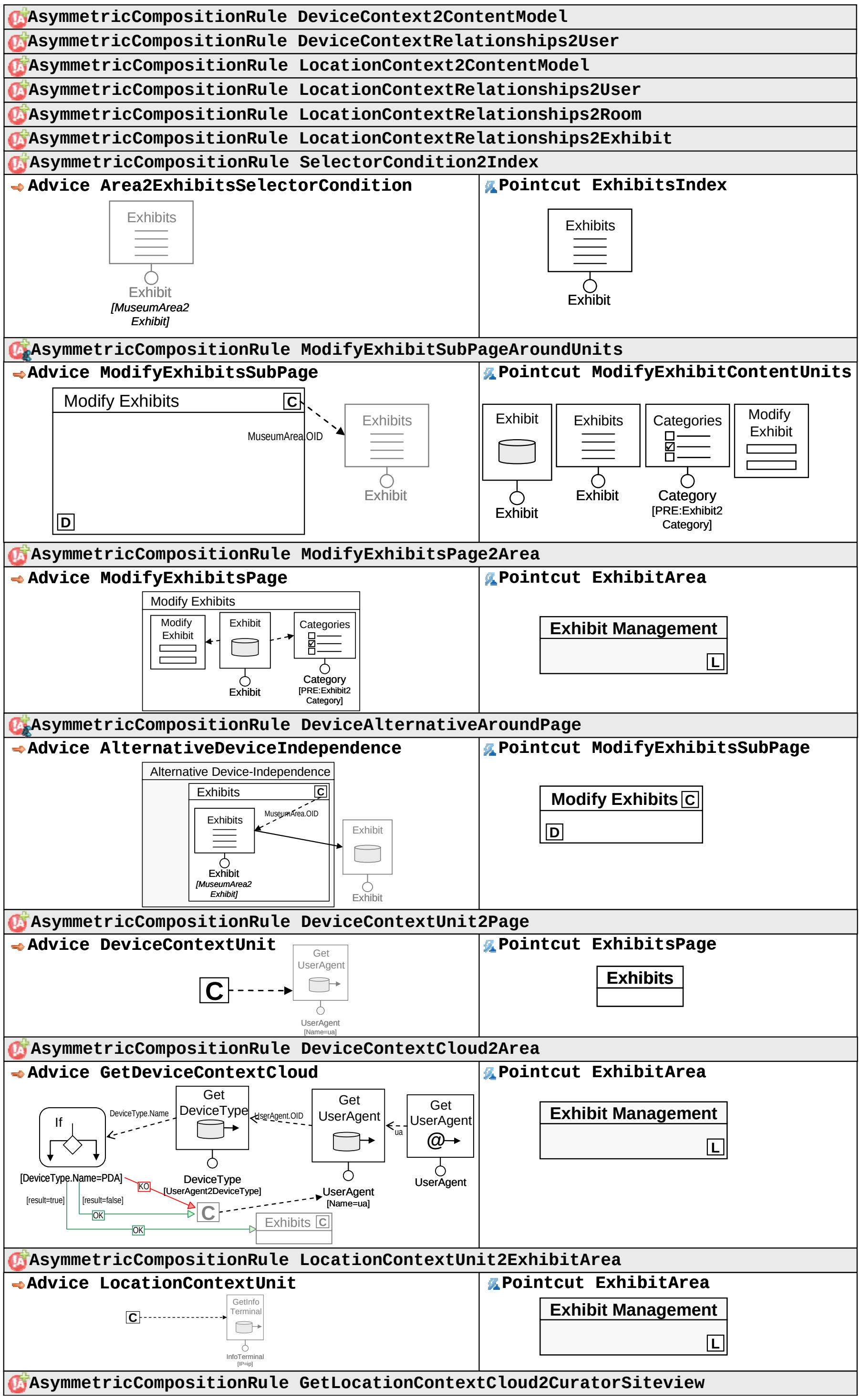

AsymmetricCompositionRule DeviceContext2ContentModel
AsymmetricCompositionRule DeviceContextRelationships2User
AsymmetricCompositionRule LocationContext2ContentModel
AsymmetricCompositionRule LocationContextRelationships2User
AsymmetricCompositionRule LocationContextRelationships2Room
AsymmetricCompositionRule LocationContextRelationships2Exhibit
AsymmetricCompositionRule SelectorCondition2Index
Advice Area2ExhibitsSelectorCondition
Exhibits
Exhibit
[MuseumArea2 Exhibit]
Pointcut ExhibitsIndex
Exhibits
Exhibit
AsymmetricCompositionRule ModifyExhibitSubPageAroundUnits
Advice ModifyExhibitsSubPage
Modify Exhibits
C
MuseumArea.OID
D
Exhibits
Exhibit
Pointcut ModifyExhibitContentUnits
Exhibit
Exhibit
Exhibits
Exhibit
Categories
Category
[PRE:Exhibit2 Category]
Modify Exhibit
AsymmetricCompositionRule ModifyExhibitsPage2Area
Advice ModifyExhibitsPage
Modify Exhibits
Modify Exhibit
Exhibit
Exhibit
Categories
Category
[PRE:Exhibit2 Category]
Pointcut ExhibitArea
Exhibit Management
L
AsymmetricCompositionRule DeviceAlternativeAroundPage
Advice AlternativeDeviceIndependence
Alternative Device-Independence
Exhibits
C
Exhibits
MuseumArea.OID
Exhibit
[MuseumArea2 Exhibit]
Exhibit
Exhibit
Pointcut ModifyExhibitsSubPage
Modify Exhibits
C
D
AsymmetricCompositionRule DeviceContextUnit2Page
Advice DeviceContextUnit
C
Get UserAgent
UserAgent
[Name=ua]
Pointcut ExhibitsPage
Exhibits
AsymmetricCompositionRule DeviceContextCloud2Area
Advice GetDeviceContextCloud
If
DeviceType.Name
Get DeviceType
UserAgent.OID
Get UserAgent
ua
Get UserAgent
@
[DeviceType.Name=PDA]
KO
DeviceType
[UserAgent2DeviceType]
UserAgent
[Name=ua]
UserAgent
[result=true]
[result=false]
OK
C
OK
Exhibits
C
Pointcut ExhibitArea
Exhibit Management
L
AsymmetricCompositionRule LocationContextUnit2ExhibitArea
Advice LocationContextUnit
C
GetInfo Terminal
InfoTerminal
[IP=ip]
Pointcut ExhibitArea
Exhibit Management
L
AsymmetricCompositionRule GetLocationContextCloud2CuratorSiteview

Figure 6.21: Customization Scenario *Context-Aware Exhibit Management* in aspectWebML

6.3.4 The Customization Aspect

The previous sections have shown how all customization scenarios can be modeled in aspectWebML in terms of separate Aspects which then are composed separately with the museum web application according to a set of AsymmetricCompositionRules. In particular, the Aspects and their Advice have been modeled such that after composing them with the museum web application, the composed model is equivalent to the WebML solution. Each customization scenario has been considered to be independent from the others and thus, defined some pieces of Advice redundantly and consequently also some AsymmetricCompositionRules.

The aspectWebML solution to customization modeling requires the modeler to specify more than one Advice in order to re-model one of the customization scenarios. Consequently, when considering all scenarios at once, the resulting *Customization* Aspect will consist of a huge number of Advice. In order to foster understandability, manageability, and reusability of Aspects the modeler should try to split the *Customization* Aspect in several Aspects, for example one for each scenario. Still, since Advice are often coupled, modelers need to consider possible dependencies between Aspects.

For the splitting task, the following peculiarities needs to be considered:

- The hypertext model is based inherently on the content model. Consequently, an Advice specified to provide customization functionality at the hypertext level very likely is based on context information specified in another Advice to be applied to the content model. In the *Exhibits In Vicinity* scenario, the *ExhibitsInVicinity* and *GetMuseumAreasContextCloud* Advice are inherently coupled with the Advice used to extend the content model with location context information (i.e., *LocationContext2ContentModel*, *LocationContextRelationships2User*, *LocationContextRelationships2Room*, and *LocationContextRelationships2Exhibit*). Furthermore, the location context information is required by several other Advice defined for other customization scenarios depending on location context information.

- Advice realizing customization functionality at the hypertext level are often coupled. In the *Exhibits In Vicinity* scenario, the *ExhibitsInVicinity* Advice is coupled with the *GetMuseumAreasContextCloud* Advice, because the ContextUnit's Link in the *ExhibitsInVicinity* Advice points to a model element defined in the *GetMuseumAreasContextCloud* Advice.

Consequently, when designing Aspects, the modeler needs to ensure that coupled Advice are encapsulated within one Aspect and that dependencies between Aspects are considered during composition. Moreover, there are several ways in how a big *Customization* Aspect can be split. For example, it can be split along with the customization scenarios or using different context properties as discriminators.

The following guidelines for designing Aspects are suggested:

1. **Start with Customization Scenarios.** A first step towards splitting the *Customization* Aspect is to design an Aspect for each customization scenario including all Advice necessary to realize the scenario. As can be seen in Section 6.3, this results in some Advice being redundantly defined in several Aspects, such as the Advice for extending the content model with location context information. Nevertheless, this allows for individually testing the customization scenarios or rather the Aspects before starting to use them in combination. The modeler will have to create a CompositionPlan for each scenario including the configuration of the ConcernModuleSequence as well as the ConcernCompositionRuleSequence. After composition, the composedModel can be tested.

Figure 6.22 presents the customization scenarios of the case study in terms of separate Aspects modeled in terms of the Aspect Diagram notation. As can be seen in the figure, some Advice are redundantly defined in several Aspects. When combining the Aspects, the modeler will have to provide appropriate AsymmetricCompositionRules in order to not weave something already specified in another rule. In this respect, the order of the Aspects might as well need to be considered.

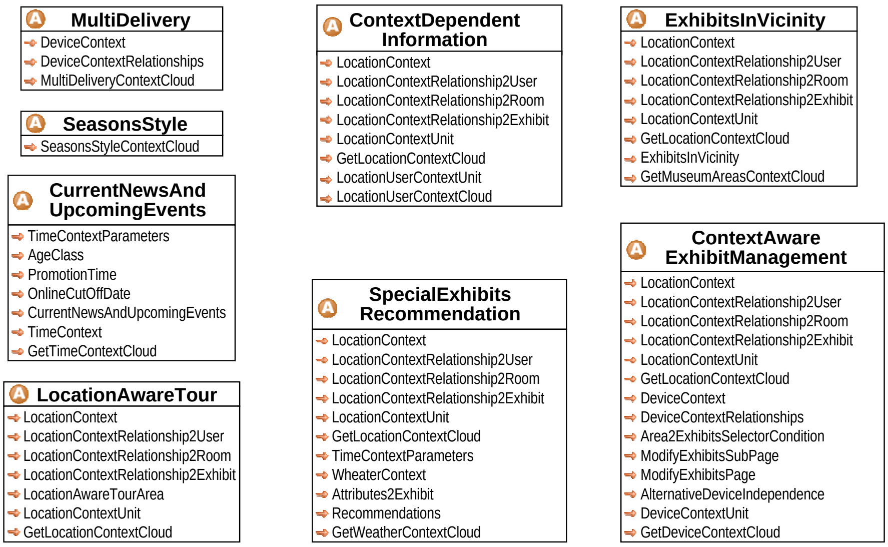

A
MultiDelivery
DeviceContext
DeviceContextRelationships
MultiDeliveryContextCloud
A
SeasonsStyle
SeasonsStyleContextCloud
A
CurrentNewsAnd UpcomingEvents
TimeContextParameters
AgeClass
PromotionTime
OnlineCutOffDate
CurrentNewsAndUpcomingEvents
TimeContext
GetTimeContextCloud
A
LocationAwareTour
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationAwareTourArea
LocationContextUnit
GetLocationContextCloud
A
ContextDependent Information
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationContextUnit
GetLocationContextCloud
LocationUserContextUnit
LocationUserContextCloud
A
SpecialExhibits Recommendation
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationContextUnit
GetLocationContextCloud
TimeContextParameters
WheaterContext
Attributes2Exhibit
Recommendations
GetWeatherContextCloud
A
ExhibitsInVicinity
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationContextUnit
GetLocationContextCloud
ExhibitsInVicinity
GetMuseumAreasContextCloud
A
ContextAware ExhibitManagement
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationContextUnit
GetLocationContextCloud
DeviceContext
DeviceContextRelationships
Area2ExhibitsSelectorCondition
ModifyExhibitsSubPage
ModifyExhibitsPage
AlternativeDeviceIndependence
DeviceContextUnit
GetDeviceContextCloud

Figure 6.22: Scenario Aspects for the Context-Aware Museum Web Application

2. **Group Scenarios According to Context Information.** When starting to use customization scenarios in combination, modelers need to identify redundant Advice in the scenarios. Typically, several customization scenarios will introduce the same context information to the content model. Thus, it is good practice to group scenarios according to their required context information and specify them within one Aspect, e.g., a *Location-Awareness* Aspect.
Ideally, this division creates no dependencies between Aspects. Still, it is very likely that customization scenarios will be based on several different context properties, such as the *Context-Aware Exhibition Management* scenario which relies on location and device context information. In this case, the modeler will have to put up with dependencies between Aspects: The scenario in question will have to be put into one of the Aspects or remain a separate Aspect.
In order to weave the Aspects, the modeler will have to identify which rules in the ConcernCompositionRuleSequences of the scenario Aspects can be reused. Typically, the rules introducing extensions to the content model can be reused from one of the scenario Aspects. Still, some rules extending the hypertext model (e.g., *GetLocationContextCloud2CuratorSiteview* from the *Context-Aware Exhibit Management* scenario and *GetLocationContextCloud2Siteview* from the *Exhibits In Vicinity* scenario) will use the same Advice but possibly different Pointcuts. In this respect, the modeler shall define a new AsymmetricCompositionRule for the Advice and provide an appropriate Pointcut that includes all JoinPoints of the other rules.
In Figure 6.23*(a)* the set of Advice has been reorganized into the *DeviceAwareness* Aspect, the

LocationAwareness Aspect, the *TimeAwareness* Aspect, and the *WeatherAwareness* Aspect and redundant Advice have been eliminated. Due to the dependencies between the Aspects, they have to be applied to the *Museum* WebML model as specified by the ConcernModuleSequence in Figure 6.23*(b)*.

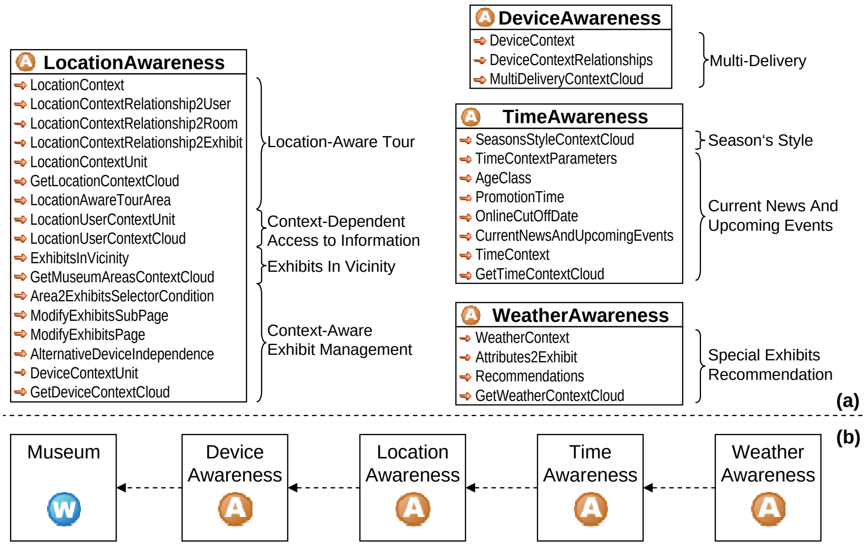


Figure 6.23: Context Aspects for the Context-Aware Museum Web Application

3. **Extract ContextModel Aspect(s).** In the worst case, every scenario is based on all context properties. In this case, the better solution is to extract all Advice to be applied on the content model from the scenario Aspects into a separate Aspect *ContextInformation*. Of course, this creates dependencies between the *ContextInformation* Aspect and the remaining Aspects.

4. **Designing Reusable Aspects.** Some parts of the museum's customization scenarios might be reused in other UWAs. Typically, these parts need to be quite independent from the museum application. For example, the device context information as well as the location context information can be reused in other applications. The *Multi-Delivery* scenario as a whole is reusable, since it is independent from the purpose of the web application. Of course, in order to reuse the location context model, the application will need to provide a similar geopositioning infrastructure as the CAM museum web application. Such an infrastructure very likely is available by default in a university campus environment. In this respect, also the context cloud for calculating the current museum area can be reused.
   In Figure 6.24 some Advice of the previous Aspects have been reorganized within two Aspects, namely the *DeviceContext* and the *LocationContext*, to be reused in other applications. Please note that the Advice *DeviceContextUnit* and *GetDeviceContextCloud* used in the *Context-Aware Exhibit Management* scenario are not fully reusable, since the context cloud contains a redirect of the user to a page of the CAM web application, i.e., one OKLink from the context cloud points to a page of the CAM web application. Nevertheless, the Advice is reusable to a large extent. The modeler merely needs to set the 'to' property of the OKLink to a target in the application in which the Aspect is used.

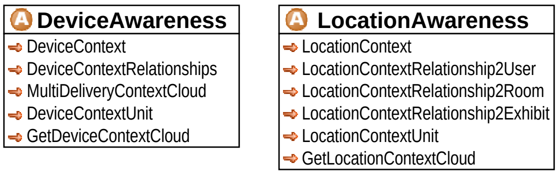

DeviceAwareness
DeviceContext
DeviceContextRelationships
MultiDeliveryContextCloud
DeviceContextUnit
GetDeviceContextCloud
LocationAwareness
LocationContext
LocationContextRelationship2User
LocationContextRelationship2Room
LocationContextRelationship2Exhibit
LocationContextUnit
GetLocationContextCloud

Figure 6.24: Reusable Aspects for the Context-Aware Museum Web Application

It is obvious that the design of several Aspects in order to capture customization functionality of a UWA is a difficult task and designing reusable Aspects is even more difficult. All the more, the modeler needs to be supported in designing Aspects and CompositionPlans within a modeling tool. Amongst others such a tool will be required to visualize coupling between Advice and dependencies between Aspects as well as debugging support when composing ConcernModules. The aspectWebML language is currently supported by the *aspectWebML Modeling Environment* presented in Chapter 7.

6.4 Comparative Discussion

The aspectWebML language is an extension to the WebML language trying to resolve WebML's weaknesses in modeling UWAs. Following, a comparative discussion of the differences of the two approaches is provided. In particular, aspectWebML's strengths with respect to WebML shall be pointed out. Nevertheless, aspectWebML is not for free. The extensions made to the WebML language require modelers not only to learn new modeling concepts but to learn a new modeling paradigm, i.e., the aspect-oriented modeling paradigm. Consequently, the following discussion will also reflect on the price to pay when using aspectWebML and answer the most obvious questions:

> *Why put up with the additional indirection via Aspects if it is possible to model customization functionality with WebML anyway? Why modeling with Aspects if weaving produces the same solution as in WebML?*

Managing Complexity, Understandability, and Traceability. In contrast to WebML, aspectWebML allows keeping separate all customization functionality from the rest of the web application, i.e., from the content model and the hypertext model. Nevertheless, after composing the customization funtionality with the core models, indeed, the original WebML solution for the context-aware museum web application can be obtained which could be used as input to code generation facilities. Still, aspectWebML is not about modeling the customization functionality itself differently than the WebML approach, i.e., in a more elegant or better way. Instead, it is about modelers explicitly dealing with customization as a separate concern. Customization functionality typically is tangled with and scattered across the whole web application model. The WebML approach, however, provides no means for providing the user with a possibility of keeping separate customization or at least of identifying which parts of the models contribute to customization. In this respect, by achieving separation of concerns, aspectWebML allows for better managing complexity, understandability, and traceability:

As already explained in Section 6.2.1, with respect to the content model, the WebML approach suggests pointing out contextual information in the content model with sub-schemata. These sub-schemata, however, only serve as guidelines for designing context information. They are not part of the model itself but rather part of the model's documentation. Furthermore, sub-schemata operate on a more coarse-grained level considering Entities and Relationships, only. They do not allow for associating a single Attribute of an Entity to a sub-schema, as can be done with aspectWebML. For example, for allowing to filter the set of recommended exhibits according to the current weather situation, in the scenario *Special Exhibits Recommendation*, the Entity *Weather* and the Domain *WeatherCondition* have been introduced to the content model. Furthermore, the Entity *Exhibit* has been extended with the Attribute *RequiredWeather*. In the WebML approach, however, this single Attribute cannot be defined as a part of the context sub-schema.

As for the hypertext model, the WebML approach requires the modeler to introduce necessary customization functionality directly into the hypertext model by making siteviews, areas, and pages context-aware. Furthermore, customization functionality might be introduced for a page without making it context-aware. For example, in the customized version of the web application, a separate section of the pages shall present the current weather situation. In case this weather situation has been stored in a GlobalParameter, the information just needs to be queried with a GetUnit instead of making the page context-aware. Again, customization functionality is mixed with the core functionality of the web application. Except for ContextUnits, there are no modeling means to explicitly mark customization functionality.

Generally speaking, in extending the existing web application with customization functionality by enhancing, replacing, and deleting modeling elements of the core web application model, the original model is lost in the WebML approach. In aspectWebML, these extension can be traced, however, by modeling customization separately with Aspects. In this respect, customization functionality can be defined as an add-on feature of the web application.

As a consequence, in aspectWebML, being able to explicitly talk about data design (i.e., the content model), hypertext design (i.e., the hypertext model), and customization design (i.e., the *Customization* Aspect) in the development process of a UWA, will improve the modelers' understanding of what is customization for the specific web application and will have a positive effect on the complexity of the models as well as on the traceability of customization functionality extensions.

Achieving Less Modeling Effort. While at first sight, aspectWebML might seem to impose a greater modeling effort on the modeler when modeling customization functionality with Aspects, e.g., through modeling Advice, Pointcuts, and AsymmetricCompositionRules, this additional effort pays off in case a customization scenario shall be applied in several places of the web application. As already said before, customization functionality is typically scattered across many places in the web application model. For example, consider the *Special Exhibits Recommendation* scenario, where recommendations about special exhibits to be visited because of the current location and weather situation need to be presented to the user on nearly every page. In the WebML approach, this means the modeler is required to visit every place of the web application model where this customization functionality shall be introduced and each time model the same set of extensions. In constrast, in the aspectWebML approach, the customization functionality needs to be modeled only once in the Aspect, while all the places where to apply the particular Advice are specified within one Pointcut. When composing the Aspect with the core web application on the basis of a CompositionPlan, the modeler only needs to associate the Advice and the Pointcut within an AsymmetricCompositionRule.

Moreover, appropriate tool support can help reducing the modeling effort imposed by separating customization with Aspects. For example, when defining an Advice used to add an Entity to the content model having a Relationship to an existing Entity, the inverse Relationship usually needs to be defined in a separate Advice. This Advice typically could be generated automatically by the tool support.

Better Maintenance Through Locality of Change Using aspectWebML has a great impact on locality of change and as a consequence also on maintainability. Keeping customization functionality separate within Aspects allows modelers perform changes to the customization functionality in one place. In contrast, in the WebML approach, modelers need to visit all places in the web application model where a certain customization functionality needs to be changed. Scenarios similar to the *Special Exhibits Recommendation* scenario require the modeler to model the same set of extensions to a number of pages. While the mere extension typically can be supported in a modeling tool via a Copy & Paste functionality, the maintainability of such a customization scenario is difficult. Imagine the *Exhibits* IndexUnit, used to present the recommended exhibits, shall display a further attribute of the exhibit, such as a small photo. In the WebML approach the modeler is required to visit all the pages participating in the scenario in order to perform the same change over and over again. For a real world web application, this manual update of a possibly very large number of pages obviously is prone to errors.

Reusability of Customization. In the WebML approach, all customization functionality is modeled such that it is intermingled with the core functionality of the web application model. As a consequence, there is no way of reusing parts of the customization functionality in other web applications other than performing a Copy & Paste between the different projects. Still, the mere copying of customization functionality has a bad effect on the models' maintainability. As outlined in Section 6.3.4, in aspectWebML, some parts of the customization functionality can be defined such that they can be easily reused within other UWAs. For example, the context information can be stored in a *ContextModel* Aspect and reused in a similar web application. As a second example, the *Multi-Delivery* customization scenario of the case study is actually independent of the museum domain and therefore the corresponding Aspect can be reused in other applications. The Aspects just need to be imported to the project's Module Repository and appropriate AsymmetricCompositionRules specifying where to weave the customization functionality via Pointcuts need to be defined.

Considering the price to pay of introducing a new paradigm such as the aspect-orientation, indeed, such a shift in paradigm requires modelers not only to learn new modeling concepts but to learn new guidelines for properly designing customization functionality within Aspects. Nevertheless, developing ubiquitous web applications requires dealing with customization functionality as a separate concern and aspectWebML provides the necessary means to do so. Being able to explicitly talk about customization design (i.e., the *Customization* Aspect) in the development process of a UWA, improves the modelers' understanding of what parts in the web application model represent customization, inherently guides WebML modelers in developing UWAs, and comes with the above listed advantages concerning complexity, understandability, traceability, maintenance, and reusability.

6.5 Summary

In this chapter, the WebML approach to modeling UWAs has been compared with the aspectWebML approach presented in this thesis using as an example a context-aware museum web application. More specifically, the customization functionality of the museum web application has first been explained and modeled in WebML. In a second step, the same functionality has been modeled with aspectWebML. The major lessons learned of the case study are four-fold.

First, in contrast to WebML, aspectWebML allows to model all customization functionality separately from the rest of the web application model. This separation of concerns allows for managing complexity in the models, for better understandability of what makes up customization, and as a consequence for better traceability of the changes made to the core web application in order to achieve customization. Nevertheless, after composing the customization functionality modeled in terms of Aspects with the core models, the original WebML solution for the context-aware museum web application can be obtained to be used as input to existing code generation facilities.

Second, aspectWebML reduces the modeling effort when designing customization functionality. This means, customization functionality is modeled once within Aspects and then composed with the core web application where it is introduced at all places as is specified by the Pointcuts. Since customization scenarios typically shall be applied to several parts of a web application, the additional modeling effort when modeling customization functionality with Aspects, e.g., through modeling Advice, Pointcuts, and AsymmetricCompositionRules, pays off.

Third, aspectWebML provides for better maintainability of models through locality of change. On the one hand, whenever changes to the customization functionality, i.e., to a specific customization scenario, are required, the modeler will be able to perform these changes once in the Aspect. On the other hand, if a customization scenario shall be applied to further parts in the web application model, the modeler just needs to change the corresponding Pointcut in order to also introduce respective customization functionality to these additional parts.

Fourth, customization functionality defined in terms of Aspects can be reused in (similar) web applications, by importing the Aspect into the ModuleRepository and by specifying the necessary Pointcuts and AsymmetricCompositionRules that allow for composition with the respective web application model.

7 The aspectWebML Modeling Environment

Contents

This chapter is dedicated to the aspectWebML tool support. In Section 7.1, a report is given on modeling and composition support within the *aspectWebML Modeling Environment* (aspectWebML ME), which is currently provided by a tree-based modeling editor built upon the Eclipse Modeling Framework (EMF). In this respect, the focus will be on some features of the editor which have been implemented to provide for better usability when modeling and composing concerns in aspectWebML, including the so-called *Cross References View* as well as the *Console View*. In Section 7.2, an outlook on more advanced modeling support is provided. The existing modeling support is planned to be extended with further ways of graphically visualizing Aspects that go beyond a simple tree-based editor. Therefore, some graphical views and/or editors dedicated to visualizing the aspect-oriented concepts in an aspectWebMLModel are proposed in terms of screenshot mock-ups. Moreover, the long-term goal is the integration of the *aspectWebML ME* with the WebML tool support, in order to profit from existing code generation facilities. Section 7.3 provides a discussion on possible ways to achieve this integration. Finally, a summary of the chapter is given in Section 7.4.

7.1 Tailoring an EMF-based Editor to Support AOM

Having specified a modeling language in Ecore, tool support in terms of a basic tree editor is just a few clicks away. By allowing to generate Java code from the Ecore model, EMF provides the means for easily testing one's language within a simple modeling editor. This tree editor can subsequently also serve as the basis for the modeling language's tool support. In this case, the editor will need to be adapted, however. To name just a few of such adaptations, the automatically generated icons which are used for the modeling elements in the tree should be replaced with dedicated icons for the modeling language in question. Furthermore, the automatically generated labels of modeling elements in the tree might be changed in order to be more informative. Finally, when modeling a reference to another modeling element in the properties view, the standard selection typically offers all modeling elements of the required type but not the set of allowed elements. Again, this needs to be changed for better supporting developers during modeling.

In the context of this thesis, further extensions were necessary. The automatically generated EMF editor should not only serve as a proof-of-concept prototype allowing to model separate concerns in aspectWebML but should also provide for the concerns' composition according to the aspectWebML composition semantics. Consequently, besides the typical adaptations of the standard tree editor the following extensions have been made:

- The implementation of the composition algorithm and its integration within the editor allowing to execute a selected CompositionPlan and thus, to compose separate concerns in an aspectWebMLModel.
- The realization of a so-called *Cross References View* - inspired by the AspectJ Development Tools (AJDT)[1] - supporting modelers in designing Aspects, Advice, and Pointcuts by visualizing how they are related to each other and how they apply to other parts of the aspectWebMLModel.
- The integration of an *Console View*, i.e., an OCL console allowing to test OCL queries on the model prior to defining an OCL-based Pointcut.
- The extension of the editor with a *Problem View* intended to display errors and warnings.

In order to give an overview on the tool support, the workbench of the *aspectWebML ME* is depicted in Figure 7.1[2]. The main parts of the workbench are introduced in the following:

(1) Main Menu Bar. The *Main Menu Bar* provides actions like *create new aspectWebMLModel*, *save*, *copy & paste*, *undo*, *redo*, etc.

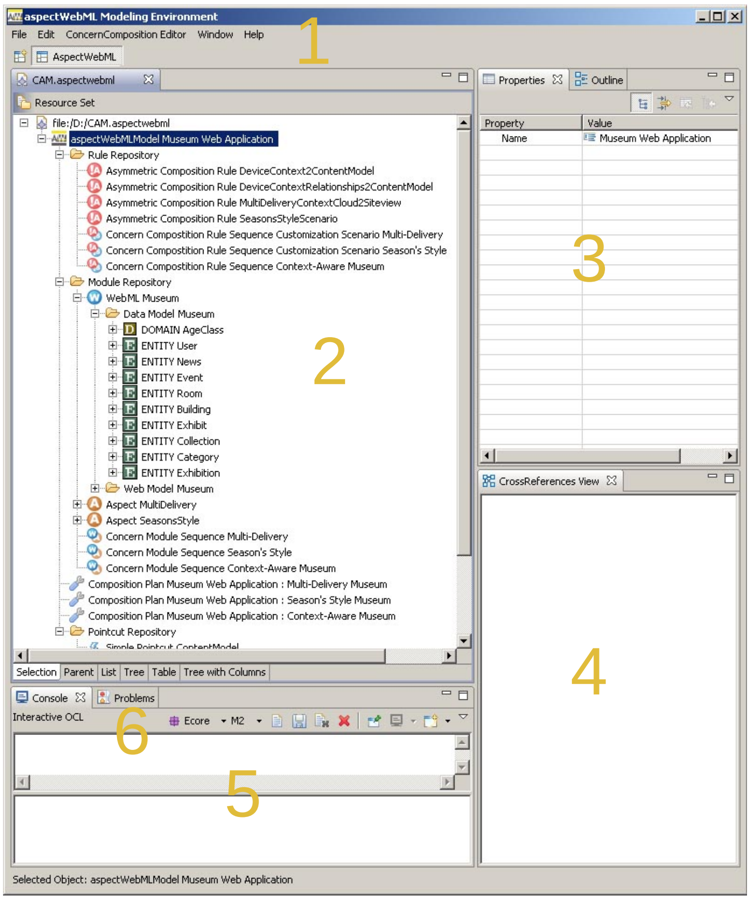
(2) Navigation View. The *Navigation View* represents the heart of the modeling editor in that it allows fast navigation through the aspectWebMLModel, i.e., its CompositionPlans and its repositories for ConcernModules, ConcernCompositionRules and Pointcuts, as well as modeling support by allowing the creation of elements in a context-based manner, i.e., via the context menu of a selected modeling element. Next to that, the Navigation View also provides input to the Properties View and the Cross References View. In the course of this chapter, the Navigation View's role as a kind of "selection provider" will be explained. As a selection provider the Navigation view enables some other views to stay synchronized according to the currently selected modeling element. This means, some views such as the Properties View depend on and will be updated according to the selection of a modeling element in the Navigation View.

(3) Properties View. The *Properties View* simply displays all properties of the currently selected modeling element in the Navigation View, which can be modified immediately. Depending on the type of the specific property, either a simple textfield, a check-box, a dropdown field or some custom window is provided for changing the current settings.

(4) Cross References View. The *Cross References View* has been designed for situations if one does not like to navigate the model in a hierarchical but rather in an "aspect-oriented" way. Due to their crosscutting nature, Aspects and their impact on the rest of the model cannot easily be visualized by standard navigation components, i.e., the Navigation View in combination with the Properties View. The Cross References View has been inspired by AJDT and provides similar functionality in terms of showing the places where an Advice is applied in the aspectWebMLModel as well as if a modeling element of the model is advised. Beyond this, the Cross References View in the *aspectWebML ME* also visualizes dependencies between Advice and between Aspects. As the Cross References View refreshes its content depending on the object selected in the Navigation View, several so-called *View Points* on the model are possible (cf. Section 7.1.1).

[1] www.eclipse.org/ajdt/

[2] I want to thank Piero Fraternali and www.webratio.com for allowing the reuse of the WebML icons for the WebML concepts in the aspectWebML language within the aspectWebML ME.


aspectWebML Modeling Environment
File
Edit
ConcernComposition Editor
Window
Help
AspectWebML
1
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Asymmetric Composition Rule DeviceContext2ContentModel
Asymmetric Composition Rule DeviceContextRelationships2ContentModel
Asymmetric Composition Rule MultiDeliveryContextCloud2Siteview
Asymmetric Composition Rule SeasonsStyleScenario
Concern Compostition Rule Sequence Customization Scenario Multi-Delivery
Concern Compostition Rule Sequence Customization Scenario Season's Style
Concern Compostition Rule Sequence Context-Aware Museum
Module Repository
WebML Museum
Data Model Museum
DOMAIN AgeClass
ENTITY User
ENTITY News
ENTITY Event
ENTITY Room
ENTITY Building
ENTITY Exhibit
ENTITY Collection
ENTITY Category
ENTITY Exhibition
Web Model Museum
Aspect MultiDelivery
Aspect SeasonsStyle
Concern Module Sequence Multi-Delivery
Concern Module Sequence Season's Style
Concern Module Sequence Context-Aware Museum
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Simple Pointcut ContentModel
2
Selection
Parent
List
Tree
Table
Tree with Columns
Properties
Outline
3
CrossReferences View
4
Console
Problems
Interactive OCL
Ecore
M2
6
5

Property	Value
Name	Museum Web Application

Selected Object: aspectWebMLModel Museum Web Application

Figure 7.1: The aspectWebML Modeling Environment: An Overview

(5) Console View. The Console View is used as an interactive OCL console allowing to query the *aspectWebMLModel*. This way, the modeler is able to test an OCL query prior to defining a Pointcut on the basis of the OCL query (cf. 7.1.2).

(6) Problems View. The Problems View's sole duty is to collect and display problems, e.g., errors in terms of model constraint violations, and warnings in terms of unresolved dependencies in a CompositionPlan.

What follows next is an introduction to the *aspectWebML ME* focusing on the extensions made to the standard tree editor, i.e., on how aspect-oriented modeling is supported within the modeling editor by means of the Cross References View and the Console View as well as on how concerns can be composed on the basis of a CompositionPlan.

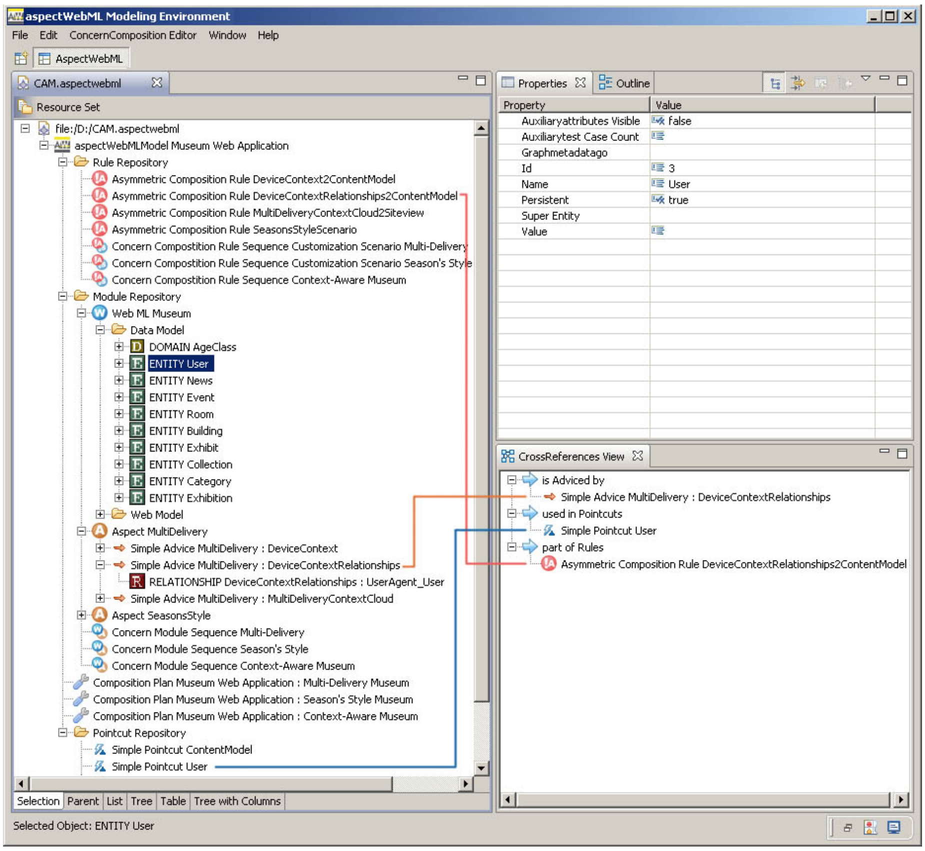
7.1.1 Aspect-Oriented Modeling with the Cross References View

The Cross References View extension has been developed as a plug-in to the editor (cf. [Pre07] for information on implementation issues). As already outlined before, the Cross References View is the first choice for navigating between aspect-oriented modeling concepts provided by the aspectWebML language. It supports modelers in designing separate concerns and Composition-

Plans by visualizing how Aspects, Advice, and Pointcuts are related to each other and how they apply to other parts of the aspectWebMLModel. Depending on the currently selected modeling element in the Navigation View, it provides different so-called *View Points*, tailored to give the modeler the exact amount of information needed to quickly absorb all interdependencies related to the object of interest. For example, if the modeler selects an Advice in the Navigation View, the Cross References View shall provide the modeler with information on where the Advice applies in a WebML model. More specifically, one can distinguish four different *View Points* of the Cross References View, namely the *ModelElement View Point* in case any element from the WebML language is selected, the *Aspect View Point* if an Aspect is selected, the *Advice View Point* if an Advice is selected, and the *Pointcut View Point* in case a Pointcut is selected in the Navigation View. In the following, the Cross References View shall be presented along with its View Points on the basis of examples.

7.1.1.1 The ModelElement View Point

The *ModelElement View Point* allows visualizing the effects of Advice, and Pointcuts on ModelElements of the aspectWebML language, i.e., any sub-class of the ModelElement meta-class whether it is defined as part of a WebML model or part of an Advice.



```
file:/D:/CAM.aspectwebml
  aspectWebMLModel Museum Web Application
    Rule Repository
      Asymmetric Composition Rule DeviceContext2ContentModel
      Asymmetric Composition Rule DeviceContextRelationships2ContentModel
      Asymmetric Composition Rule MultiDeliveryContextCloud2Siteview
      Asymmetric Composition Rule SeasonsStyleScenario
      Concern Compostition Rule Sequence Customization Scenario Multi-Delivery
      Concern Compostition Rule Sequence Customization Scenario Season's Style
      Concern Compostition Rule Sequence Context-Aware Museum
    Module Repository
      Web ML Museum
        Data Model
          DOMAIN AgeClass
          ENTITY User
          ENTITY News
          ENTITY Event
          ENTITY Room
          ENTITY Building
          ENTITY Exhibit
          ENTITY Collection
          ENTITY Category
          ENTITY Exhibition
        Web Model
      Aspect MultiDelivery
        Simple Advice MultiDelivery : DeviceContext
        Simple Advice MultiDelivery : DeviceContextRelationships
          RELATIONSHIP DeviceContextRelationships : UserAgent_User
        Simple Advice MultiDelivery : MultiDeliveryContextCloud
      Aspect SeasonsStyle
      Concern Module Sequence Multi-Delivery
      Concern Module Sequence Season's Style
      Concern Module Sequence Context-Aware Museum
    Composition Plan Museum Web Application : Multi-Delivery Museum
    Composition Plan Museum Web Application : Season's Style Museum
    Composition Plan Museum Web Application : Context-Aware Museum
    Pointcut Repository
      Simple Pointcut ContentModel
      Simple Pointcut User
```

Property	Value
Auxiliaryattributes Visible	false
Auxiliarytest Case Count	
Graphmetadatago	
Id	3
Name	User
Persistent	true
Super Entity	
Value	

```
is Adviced by
  Simple Advice MultiDelivery : DeviceContextRelationships
used in Pointcuts
  Simple Pointcut User
part of Rules
  Asymmetric Composition Rule DeviceContextRelationships2ContentModel
```

Selected Object: ENTITY User

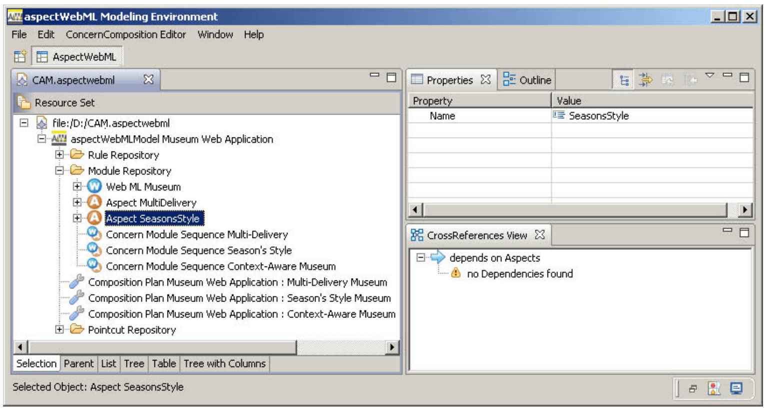
Figure 7.2: The ModelElement View Point

In Figure 7.2, the ModelElement View Point is depicted for the *User* Entity, which is selected in the Navigation View. In the Cross References View, the View Point contains three sections: First, in the *is Advised by* section all Advice of the project are listed that are applied to the selected ModelElement by one of the project's Pointcuts. The *User* Entity is advised by the *DeviceContextRelationships* SimpleAdvice of the Aspect *MultiDelivery*, i.e., it is extended with a Relationship to the *UserAgent* Entity. Second, the Pointcuts that reference the selected ModelElement as one of their JoinPoints are listed in the *used in Pointcuts* section. The SimplePointcut *User*, which selects the *User* Entity as JoinPoint is shown. Finally, in the *part of Rules* section, the AsymmetricCompositionRules are listed in which the ModelElement is advised. In fact, the listed rule *DeviceContextRelationships2ContentModel* combines the SimpleAdvice *DeviceContextRelationships* with the SimplePointcut *User*.

The Cross References View furthermore allows navigating to the listed elements. This means, in case the developer clicks on one of the Advice listed in the Cross References View, the Navigation View is updated and the selected Advice is also highlighted in the Navigation View. This way developers can inspect in detail the selected element along with its properties and child elements. Moreover, in large projects, where it is difficult to maintain an overview, this feature is particularly useful.

7.1.1.2 The Aspect View Point

In the *Aspect View Point*, an Aspect's dependencies with other Aspects shall be visualized and thus, allows the modeler to identify what further Aspects possibly need to be part of a CompositionPlan. More specifically, an Aspect is dependent on another Aspect, if one of its Advice is dependent on one of the other Aspect's Advice (see also the Advice View Point). The Aspects, on which the selected Aspect depends, are listed in the *depends on Aspects* section of the Cross References View. In Figure 7.3, the Aspect *SeasonsStyle* is selected in the Navigation View. Since, the Aspect does not depend on any other Aspect in the project, the Cross References View informs the modeler that no dependencies have been found.


aspectWebML Modeling Environment
File
Edit
ConcernComposition Editor
Window
Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Module Repository
Web ML Museum
Aspect MultiDelivery
Aspect SeasonsStyle
Concern Module Sequence Multi-Delivery
Concern Module Sequence Season's Style
Concern Module Sequence Context-Aware Museum
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Selection
Parent
List
Tree
Table
Tree with Columns
Properties
Outline
CrossReferences View
depends on Aspects

Property	Value
Name	SeasonsStyle

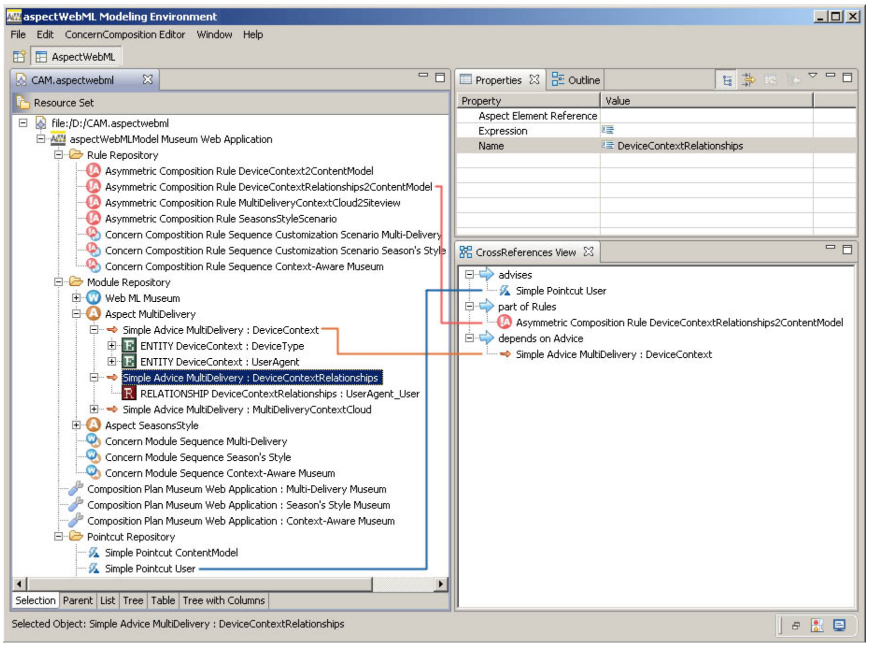
no Dependencies found

Selected Object: Aspect SeasonsStyle

Figure 7.3: The Aspect View Point

7.1.1.3 The Advice View Point

The *Advice View Point* allows visualizing the interrelationships of an Advice with the Pointcuts and AsymmetricCompositionRules it is used with. The *advises* section lists the Pointcuts which are affected by the Advice. In Figure 7.4, the SimpleAdvice *DeviceContextRelationships* is selected in the Navigation View, and consequently, in the Cross References View, the SimplePointcut *User* is listed. In the section *part of Rules*, similarly to the ModelElement View Point, those AsymmetricCompositionRules are listed in which the selected Advice is used. Furthermore, an Advice's dependencies on pieces of other Advice, possibly defined in other Aspects, is visualized. All Advice, on which the selected Advice depends, are listed in the *depends on Advice* section of the Cross References View. In Figure 7.4, the SimpleAdvice *DeviceContextRelationships* depends on the SimpleAdvice *DeviceContext*, since the Relationship *UserAgent_User* points to the Entity *UserAgent* defined in the SimpleAdvice *DeviceContext*.



file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Asymmetric Composition Rule DeviceContext2ContentModel
Asymmetric Composition Rule DeviceContextRelationships2ContentModel
Asymmetric Composition Rule MultiDeliveryContextCloud2Siteview
Asymmetric Composition Rule SeasonsStyleScenario
Concern Compostition Rule Sequence Customization Scenario Multi-Delivery
Concern Compostition Rule Sequence Customization Scenario Season's Style
Concern Compostition Rule Sequence Context-Aware Museum
Module Repository
Web ML Museum
Aspect MultiDelivery
Simple Advice MultiDelivery : DeviceContext
ENTITY DeviceContext : DeviceType
ENTITY DeviceContext : UserAgent
Simple Advice MultiDelivery : DeviceContextRelationships
RELATIONSHIP DeviceContextRelationships : UserAgent_User
Simple Advice MultiDelivery : MultiDeliveryContextCloud
Aspect SeasonsStyle
Concern Module Sequence Multi-Delivery
Concern Module Sequence Season's Style
Concern Module Sequence Context-Aware Museum
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Simple Pointcut ContentModel
Simple Pointcut User

Property	Value
Aspect Element Reference	
Expression	
Name	DeviceContextRelationships

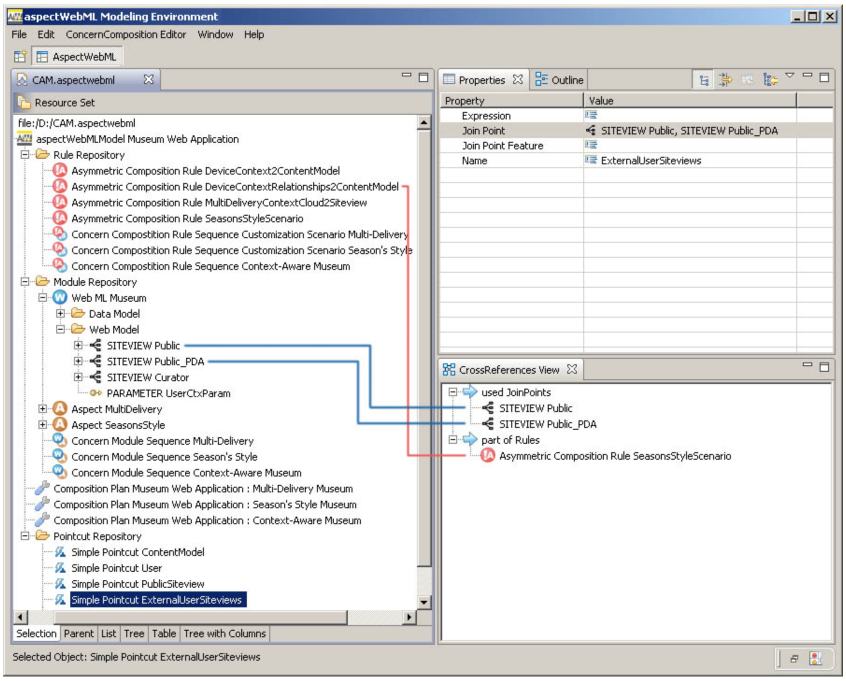
advises
Simple Pointcut User
part of Rules
Asymmetric Composition Rule DeviceContextRelationships2ContentModel
depends on Advice
Simple Advice MultiDelivery : DeviceContext

Selected Object: Simple Advice MultiDelivery : DeviceContextRelationships

Figure 7.4: The Advice View Point

7.1.1.4 The Pointcut View Point

Last but not least, the *Pointcut View Point* visualizes the JoinPoints of the Pointcut selected in the Navigation View no matter how the Pointcut has been defined, i.e., either as an enumeration of JoinPoints or on the basis of an OCL query. In the *used JoinPoints* section, the JoinPoints are listed and can be used to navigate to the respective places in the model using the Navigation View. For example, in Figure 7.5, the *ExternalUserSiteviews* SimplePointcut selects the *Public* Siteview as well as the *Public_PDA* Siteview of the *Museum* WebML model. In the *part of Rules* section the AsymmetricCompositionRules are listed in which the Pointcut has been used, e.g., in the *SeasonsStyleScenario* rule in Figure 7.5.


aspectWebML Modeling Environment
File
Edit
ConcernComposition Editor
Window
Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Asymmetric Composition Rule DeviceContext2ContentModel
Asymmetric Composition Rule DeviceContextRelationships2ContentModel
Asymmetric Composition Rule MultiDeliveryContextCloud2Siteview
Asymmetric Composition Rule SeasonsStyleScenario
Concern Compostition Rule Sequence Customization Scenario Multi-Delivery
Concern Compostition Rule Sequence Customization Scenario Season's Style
Concern Compostition Rule Sequence Context-Aware Museum
Module Repository
Web ML Museum
Data Model
Web Model
SITEVIEW Public
SITEVIEW Public_PDA
SITEVIEW Curator
PARAMETER UserCtxParam
Aspect MultiDelivery
Aspect SeasonsStyle
Concern Module Sequence Multi-Delivery
Concern Module Sequence Season's Style
Concern Module Sequence Context-Aware Museum
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Simple Pointcut ContentModel
Simple Pointcut User
Simple Pointcut PublicSiteview
Simple Pointcut ExternalUserSiteviews
Selection
Parent
List
Tree
Table
Tree with Columns
Properties
Outline
CrossReferences View
used JoinPoints
SITEVIEW Public
SITEVIEW Public_PDA
part of Rules
Asymmetric Composition Rule SeasonsStyleScenario

Property	Value
Expression	
Join Point	SITEVIEW Public, SITEVIEW Public_PDA
Join Point Feature	
Name	ExternalUserSiteviews

Selected Object: Simple Pointcut ExternalUserSiteviews

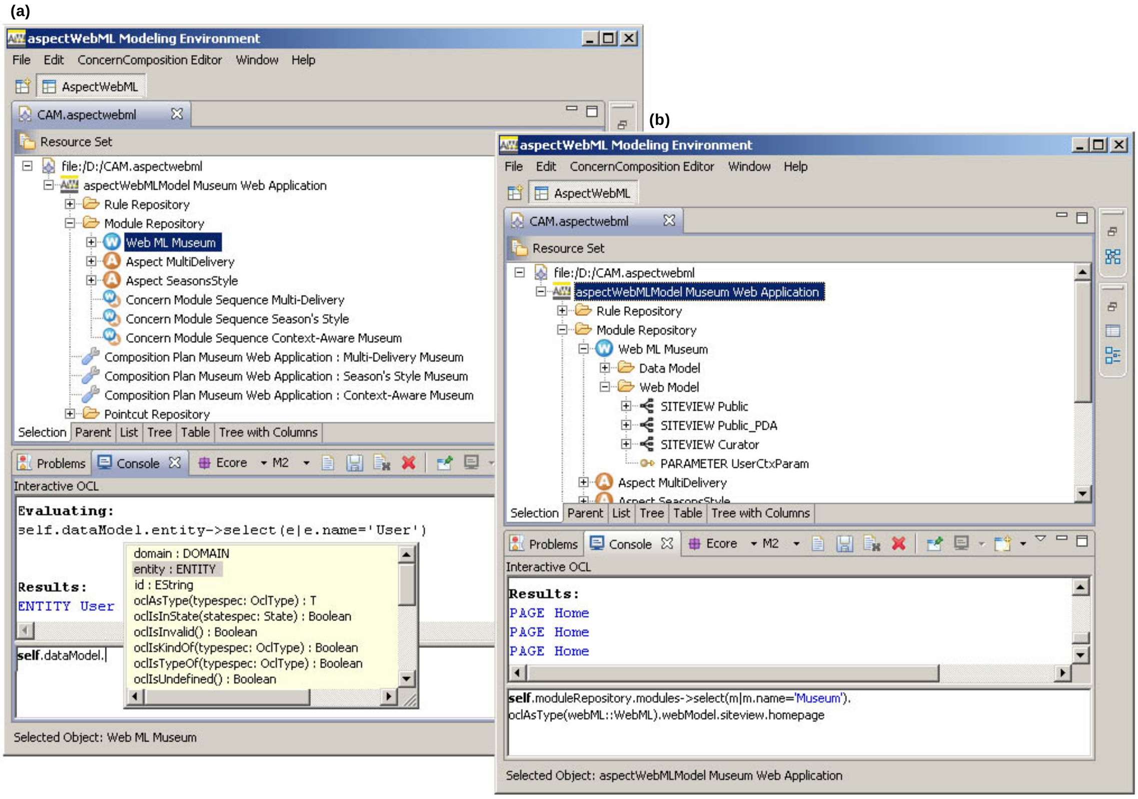
Figure 7.5: The Pointcut View Point

7.1.2 Defining OCL-based Pointcuts with the Console

As already indicated before, the Console View is used to perform OCL queries on the *aspectWebMLModel*. This way, the modeler is able to test an OCL query prior to defining a Pointcut on the basis of the OCL query. To do so, the interactive OCL console released under the Eclipse Model Development Tools[3] (MDT) project has been integrated into the model editor (for implementation details see [Pre07]). The bottom field of the console accepts OCL queries and supports the modeler with a so-called *content-assist* menu, which is similar to typical code-completion features in a programming IDE. As is illustrated in Figure 7.6*(a)*, during the specification of an OCL query, the content-assist menu helps navigating through the aspectWebML metamodel. In this example, the *User* Entity shall be selected by the OCL query. In the Navigation view, the context of the OCL query is specified by the selected model element, i.e., the *Museum* WebML model. Consequently, in the OCL query one needs to navigate from the WebML model to the Entity via the references specified in the aspectWebML metamodel, i.e., to the Data Model and then to its Entities. In Figure 7.6*(a)*, the *content-assist* menu proposes the *entity* reference. From the collection of Entities, then the Entity with the name 'User' needs to be selected with the OCL *select* operation. The full OCL query is already given in the top field which is intended to show the output of a query and errors. As can be seen, the OCL query has been successfully evaluated and the Entity *User* has been returned as a result.

In Figure 7.6*(b)*, a second example describing a first-level Pointcut, i.e., a Pointcut defined on the basis of properties of the aspectWebML language rather than on information from the *Museum Web Application* aspectWebMLModel. The OCL query depicted in the bottom field of the OCL

[3] www.eclipse.org/modeling/mdt/


(a)
aspectWebML Modeling Environment
File Edit ConcernComposition Editor Window Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Module Repository
Web ML Museum
Aspect MultiDelivery
Aspect SeasonsStyle
Concern Module Sequence Multi-Delivery
Concern Module Sequence Season's Style
Concern Module Sequence Context-Aware Museum
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Selection Parent List Tree Table Tree with Columns
Problems Console Ecore M2
Interactive OCL
domain : DOMAIN
entity : ENTITY
id : EString
oclAsType(typespec: OclType) : T
oclIsInState(statespec: State) : Boolean
oclIsInvalid() : Boolean
oclIsKindOf(typespec: OclType) : Boolean
oclIsTypeOf(typespec: OclType) : Boolean
oclIsUndefined() : Boolean
(b)
aspectWebML Modeling Environment
File Edit ConcernComposition Editor Window Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Module Repository
Web ML Museum
Data Model
Web Model
SITEVIEW Public
SITEVIEW Public_PDA
SITEVIEW Curator
PARAMETER UserCtxParam
Aspect MultiDelivery
Aspect SeasonsStyle
Selection Parent List Tree Table Tree with Columns
Problems Console Ecore M2
Interactive OCL

(a)

```
Evaluating:
self.dataModel.entity->select(e|e.name='User')

Results:
ENTITY User
```

```
self.dataModel.
```

Selected Object: Web ML Museum

(b)

```
Results:
PAGE Home
PAGE Home
PAGE Home
```

```
self.moduleRepository.modules->select(m|m.name='Museum').
oclAsType(webML::WebML).webModel.siteview.homepage
```

Selected Object: aspectWebMLModel Museum Web Application

Figure 7.6: Definition of the User Entity Pointcut

console selects all pages that are defined to be the home pages of the web application's siteviews. In the top field of the console the output for the query is depicted. Since the *Museum* WebML model contains three siteviews of which each needs to specify a home page, the query results in a list of three pages all having the same name 'Home'.

Please note that, in Figure 7.6*(b)*, the context of the query is the *Museum Web Application* aspectWebMLModel, i.e., the root element of the project. In contrast to the previous example, in order to specify OCL-based Pointcuts, modelers currently are required to specify the OCL query starting navigation from this root element. This is necessary, in order to allow the Cross References View to correctly calculate the JoinPoints. Furthermore, two peculiarities of the example query of Figure 7.6*(b)* need to be pointed out: First, since the ModuleRepository possibly contains more than one WebML model, in the example query, it is explicitly stated to select home pages from the *Museum* WebML model, only. And second, it is necessary, to cast the selection of ConcernModules obtained with `select(m|m.name='Museum')` to a concrete type, i.e., either to an Aspect or to a WebML model. In this case, the selection needs to be casted to the WebML type, which is done by correctly specifying the namespace according to the aspectWebML metamodel package structure, i.e., `webML::WebML`. Again, the modeler is supported with the *content-assist* feature of the interactive OCL console. Finally, if the OCL query provides the expected results, the modeler can copy it in order to specify an OCL-based Pointcut.

7.1.3 Composing Concerns in the aspectWebML Modeling Environment

Composition of concerns in aspectWebML is supported on the basis of the composition semantics specified in Chapter 5. In the *aspectWebML ME*, composition of concerns into a composed model can be achieved by executing a CompositionPlan. To do so, the CompositionPlan needs to be fully configured. This includes the definition of the ConcernModuleSequence specifying in which order the concerns (e.g., a WebML model and some Aspects) need to be composed as well as as the definition of the ConcernCompositionRuleSequence allowing to fine-tune the composition.

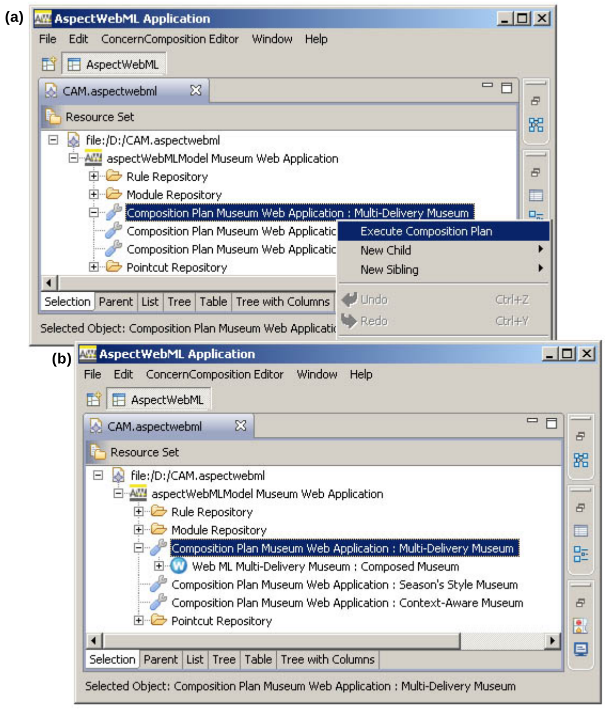

(a)
AspectWebML Application
File Edit ConcernComposition Editor Window Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Module Repository
Composition Plan Museum Web Application : Multi-Delivery Museum
Composition Plan Museum Web Applicatic
Composition Plan Museum Web Applicatic
Pointcut Repository
Execute Composition Plan
New Child
New Sibling
Undo Ctrl+Z
Redo Ctrl+Y
Selection Parent List Tree Table Tree with Columns
Selected Object: Composition Plan Museum Web Applicati
(b)
AspectWebML Application
File Edit ConcernComposition Editor Window Help
AspectWebML
CAM.aspectwebml
Resource Set
file:/D:/CAM.aspectwebml
aspectWebMLModel Museum Web Application
Rule Repository
Module Repository
Composition Plan Museum Web Application : Multi-Delivery Museum
Web ML Multi-Delivery Museum : Composed Museum
Composition Plan Museum Web Application : Season's Style Museum
Composition Plan Museum Web Application : Context-Aware Museum
Pointcut Repository
Selection Parent List Tree Table Tree with Columns
Selected Object: Composition Plan Museum Web Application : Multi-Delivery Museum

Figure 7.7: A Composed Model

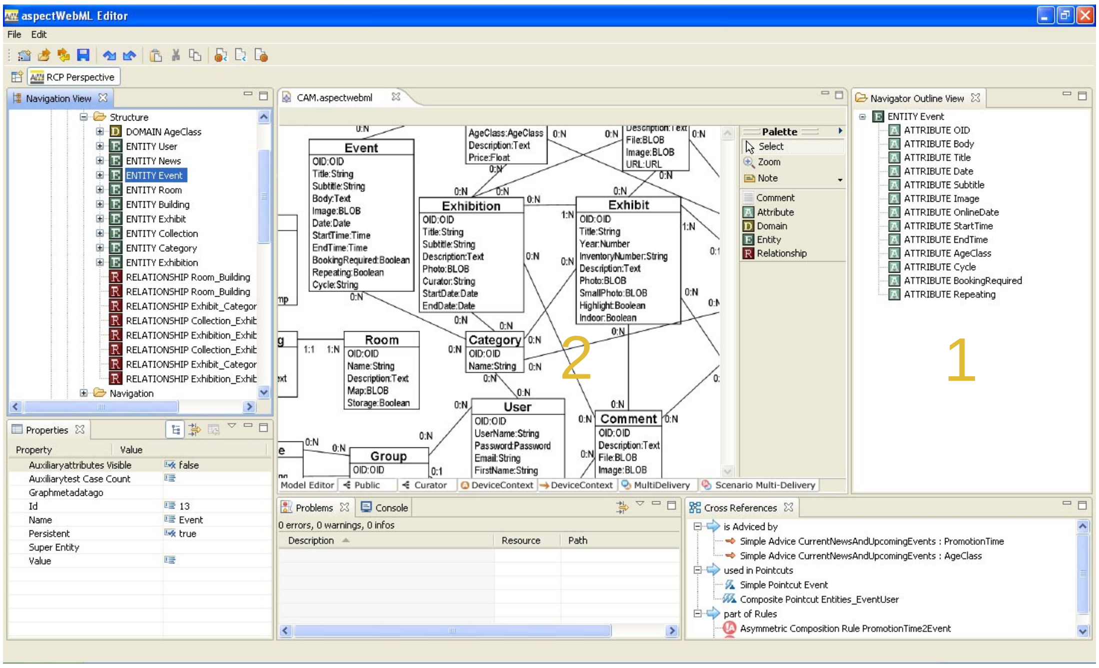
In case the CompositionPlan has been fully configured, the modeler can compose the selected concerns by calling the *Execute Composition Plan* action of the context menu. For example, in Figure 7.7*(a)*, the CompositionPlan *Multi-Delivery Museum* shall be executed, which will weave the *Multi-Delivery* Aspect into the *Museum* WebML model. A successful composition of the CompositionPlan will yield a composed model, i.e., a WebML model where the Aspects have been integrated. In Figure 7.7*(b)*, the execution of the *Multi-Delivery Museum* CompositionPlan has resulted in the *Composed Museum* WebML model. In case of errors during composition or during the consistency check, the execution of the Composition Plan will abort and the user will be provided with information on the occurred errors in the editor's Problem View. The consistency check is automatically performed prior to composition and will detect errors in the CompositionPlan and the Concerns it shall compose. For example, the user will be pointed to unresolved dependencies between Advice or between Aspects.

7.2 Towards Advanced Modeling Support for aspectWebML

While EMF editors have proved their usefulness for providing proof-of-concept prototypes, their cumbersome handling certainly does hamper the design of large scale models. Consequently, the plan is to extend the existing modeling support with further ways of graphically visualizing aspect-oriented concepts in aspectWebML that go beyond a simple tree-based editor. This advanced modeling support shall be realized on the basis of the Eclipse Graphical Modeling Framework[4]. Following, some graphical views and/or editors dedicated to visualizing the aspect-oriented concepts in an *aspectWebML* model are proposed in terms of screenshot mock-ups.

7.2.1 The aspectWebML Workbench

In the future, users shall be able to model aspectWebMLModels in a graph-based editor. In particular, the editing functionality shall resemble that for the WebRatio tool, meaning that the modeling environment shall provide an editor for modeling a WebML *data model* as well as WebML *siteviews*.



- Structure
  - DOMAIN AgeClass
  - ENTITY User
  - ENTITY News
  - ENTITY Event
  - ENTITY Room
  - ENTITY Building
  - ENTITY Exhibit
  - ENTITY Collection
  - ENTITY Category
  - ENTITY Exhibition
  - RELATIONSHIP Room_Building
  - RELATIONSHIP Room_Building
  - RELATIONSHIP Exhibit_Categor
  - RELATIONSHIP Collection_Exhib
  - RELATIONSHIP Exhibition_Exhib
  - RELATIONSHIP Collection_Exhib
  - RELATIONSHIP Exhibit_Categor
  - RELATIONSHIP Exhibition_Exhib
- Navigation

Properties

Property	Value
Auxiliaryattributes Visible	false
Auxiliarytest Case Count	
Graphmetadatago	
Id	13
Name	Event
Persistent	true
Super Entity	
Value	

- ENTITY Event
  - ATTRIBUTE OID
  - ATTRIBUTE Body
  - ATTRIBUTE Title
  - ATTRIBUTE Date
  - ATTRIBUTE Subtitle
  - ATTRIBUTE Image
  - ATTRIBUTE OnlineDate
  - ATTRIBUTE StartTime
  - ATTRIBUTE EndTime
  - ATTRIBUTE AgeClass
  - ATTRIBUTE Cycle
  - ATTRIBUTE BookingRequired
  - ATTRIBUTE Repeating

0 errors, 0 warnings, 0 infos

Description	Resource	Path

- is Adviced by
  - Simple Advice CurrentNewsAndUpcomingEvents : PromotionTime
  - Simple Advice CurrentNewsAndUpcomingEvents : AgeClass
- used in Pointcuts
  - Simple Pointcut Event
  - Composite Pointcut Entities_EventUser
- part of Rules
  - Asymmetric Composition Rule PromotionTime2Event

Figure 7.8: Graphical Modeling in the aspectWebML Modeling Environment

The major parts of the *aspectWebML ME* workbench depicted in Figure 7.8, basically will have the same purpose as in the already existing tree-based editor described in Section 7.1. The advanced tool support, shall extend the aspectWebML workbench with two further views, which are explained in the following.

(1) Outline View. The *Outline View* simply provides the user with the set of child elements for the current selected object of the Navigation View. Hence it takes away some of the complexity

[4] www.eclipse.org/gmf

of the standard Navigation View, which always displays the complete aspectWebMLModel. Like the Navigation View, it also acts as selection provider to other views.

(2) Graphical Editor Area. The *Graphical Editor Area* is placed at the very center of the workbench, which mirrors its significance to the *aspectWebML ME*. The Editor Area will be occupied by various graphical editors, each contributing additional modeling capabilities. At the bottom of the Editor Area in Figure 7.8, several tabs indicate that some of these graphical editors are opened. Currently, the *Data Model Editor* is activated allowing to edit a WebML data model. Similar to WebML's modeling tool WebRatio, a separate editor is available for modeling the hypertext. At the bottom of the Editor Area, two tabs are indicating two siteviews of the hypertext level, namely the *Public* siteview and the *Curator* siteview. In addition to these editors, the intention is to implement four further editors that shall facilitate modeling aspectWebMLModels. In this respect, the notation and icons proposed for the aspectWebML language are reused (cf. Chapter 5).

7.2.2 The Aspect Editor

The *Aspect Editor*'s main task is to visualize a single Aspect and its children in terms of a set of Advice using a notation similar to UML class diagrams. In Figure 7.9*(a)*, the Aspect *MultiDelivery* is visualized in the Aspect Editor. By issuing a double click on a specific Advice, a user can open the *Advice Editor* allowing to visualize the selected Advice (cf. Section 7.2.5). Moreover, an additional Advice can be added to the Aspect and the Advice Editor will be opened allowing for modeling the Advice.

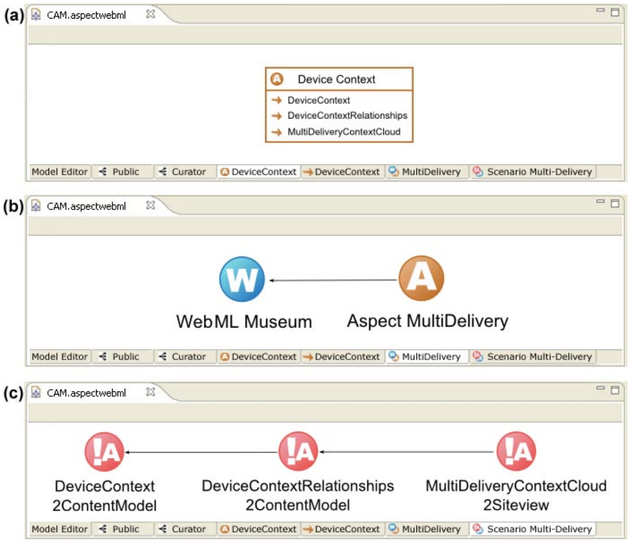


Figure 7.9: Aspect Editor (a), Module Sequence Editor (b), Rule Sequence Editor (c)

7.2.3 The Module Sequence Editor

The *Module Sequence Editor* presents the order of ConcernModules in a CompositionPlan. The diagram depicted in Figure 7.9*(b)* is to be read from left to right, meaning that the *Museum* WebML model is composed with the *MultiDelivery* Aspect before any other ConcernModule is composed.

7.2.4 The Rule Sequence Editor

The *Rule Sequence Editor* displays a sequence of ConcernCompositionRules and possibly its subsequences (cf. Figure 7.9*(c)*). Again, this diagram is read from left to right, meaning that the AsymmetricCompositionRule *DeviceContext2ContentModel* is applied before the *DeviceContextRelationships2ContentModel* rule and the *MultiDeliveryContextCloud2SiteView* rule.

7.2.5 The Advice Editor

Finally, the *Advice Editor* provides the user with modeling support for Advice. Within an Advice, parts of the content or the hypertext are modeled which later can be composed with the core web application. Modeling elements specified in an Advice can have references to modeling elements defined within the core web application or within other Advice. Consequently, the Advice Editor will have to deal with this peculiarity in order to correctly visualize an Advice.

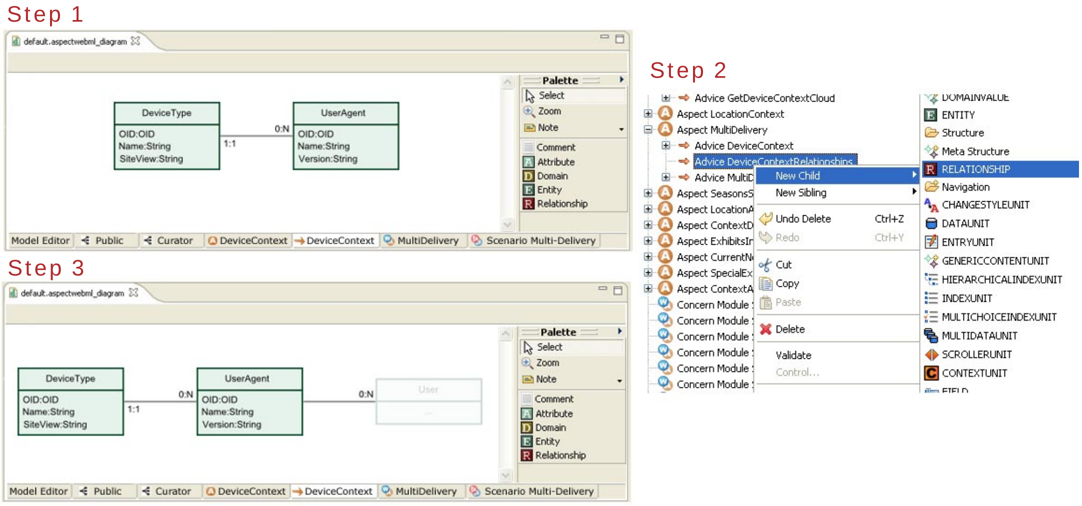


Figure 7.10: The Advice Editor

To explain the Advice Editor in more detail, a trivial example from the case study presented in Chapter 6 is used. Figure 7.10 presents an Advice named *DeviceContext*, which is part of the *Multi-Delivery* customization scenario taken from the case study in Section 6.3.2.1. The Advice shall extend the data model of the *Museum* WebML model with two entities, namely Entity *DeviceType* and Entity *UserAgent*. In addition, the *UserAgent* Entity shall have a Relationship to the *User* Entity of the Museum application.

As is indicated in Figure 7.10, in a first step, the user models these two entities along with their attributes and the relationship in between them. To do so, the user selects the necessary modeling elements from the *Palette* placed on the right-hand side of of the Advice Editor. In a second step, the user is required to model the relationship between the *UserAgent* Entity and the existing *User* Entity of the *Museum* WebML model. The *User* Entity, however, is not part of the diagram visualized by the Advice Editor. Therefore, the user cannot draw the relationship between the entities in the modeling area of the Advice Editor. In such situations, the user has to model the necessary relationship on the basis of the Navigation View as it is outlined in *Step 2* of

Figure 7.10. The Relationship *UserAgent_User* is added to the Advice by using the context menu, while its properties including the reference to the *User* Entity are set in the Properties View. As a result, in the Advice Editor the Relationship to the *User* Entity is visualized. The *User* Entity is highlighted in grey to indicate that it is not part of the Advice and cannot be manipulated or deleted in the Advice Editor.

7.3 Towards Integrating the aspectWebML Modeling Environment with WebRatio

With respect to providing comprehensive tool support for aspectWebML, the integration with the WebRatio tool for exploiting existing code generation facilities is envisaged. Still, a full integration is difficult to achieve for the following reasons:

First, since WebRatio is a commercial tool for which the source code is not available, a direct integration of the aspectWebML language within the WebRatio tool currently is not possible. Furthermore, the WebML models created with WebRatio are still serialized in XML according to the WebML DTD (cf. Chapter 4). As a consequence, a possible solution for integration is to provide import/export facilities in the aspectWebML tool support, which allows for exchanging WebML models between WebRatio and the aspectWebML ME. This way, modelers may (i) develop their web application models in WebRatio, (ii) import the WebML model into the ModuleRepository of the aspectWebML project in the aspectWebML ME, (iii) specify and compose crosscutting concerns in the aspectWebML ME, and (iv) export the composed model to the WebRatio tool for code generation purposes.

Second, the WebML extensions allowing for customization modeling have been recently defined, only. As a consequence, the necessary tool support for modeling customization has not yet been integrated within WebRatio . This means that, WebRatio neither allows for modeling customization nor for generating code for ubiquitous web applications as suggested in [CDMF07]. It is therefore not possibly to export WebML models from the aspectWebML ME that contain instances of WebML's customization modeling concepts. Nevertheless, the aspectWebML approach allows modelers to generally model other concerns than customization, although they are restricted to using WebML concepts originally defined in the WebML DTD (cf. Chapter 4).

7.4 Summary

In this chapter, a report has been given on modeling and composition support within the *aspectWebML Modeling Environment*, the tool support for the aspectWebML language. The aspectWebML Modeling Environment is currently realized on the basis of a tree-based modeling editor built upon the Eclipse Modeling Framework. In order to better support developers in modeling separate concerns with aspectWebML, the aspectWebML Modeling Environment offers a so-called *Cross References View* visualizing how Aspects, Advice, and Pointcuts are related to each other and how they apply to other parts of the aspectWebMLModel. Moreover, a *Console View*, i.e., an OCL console allowing to test OCL queries on the model prior to defining an OCL-based Pointcut, has been integrated. This tool support is made available for download[5] including some sample models as well as further documentation. Moreover, an outlook on advanced modeling

[5] *www.wit.at/people/schauerhuber/aspectUWA*

support to be realized on the basis of the Eclipse Graphical Modeling Framework for developing graphical modeling editors is given. This includes the graphical modeling means for designing WebML models just as in WebML's own tool support as well as four editors, i.e., Aspect Editor, Advice Editor, Module Sequence Editor, and Rule Sequence Editor, dedicated to designing the aspect-oriented parts in an aspectWebML project. Finally, a discussion on how to integrate the aspectWebML Modeling Environment with the WebRatio tool is given, specifically pointing out the limitations of such an integrated modeling environment due to the fact that WebRatio does not yet provide modeling support for customization.

8 Related Work to aspectWebML

Contents

The aspect-orientation paradigm has often been considered as an extension to the object-orientation paradigm. Consequently, it is not surprising that first aspect-oriented languages have been designed as an extension to a general-purpose object-oriented language. At modeling level, it seems almost natural to use and/or extend the standard for object-oriented modeling, i.e., the Unified Modeling Language (UML), for AOM (cf. Chapter 3). In the past years, however, one can observe the propagation of the aspect-orientation paradigm to several other domains such as the web modeling field. This chapter is dedicated to a discussion of approaches related to aspectWebML, i.e., approaches to extend a web modeling language with AOM modeling concepts or more specifically, to employ aspect-orientation for modeling UWAs. In this respect, the benefits of the aspectWebML approach shall be pointed out with respect to two closely related approaches as well as to other more widely related ones. Following, UWE's approach to *modeling adaptivity with aspects* is presented, in Section 8.1, while Hera's *semantics-based aspect-oriented approach to adaptation in web engineering* is discussed in Section 8.2. Thereafter, the focus is on more widely related work, i.e., web modeling approaches that also provide for AOM but which have been designed for supporting (crosscutting) concerns other than customization (cf. Section 8.3).

8.1 UWE - ”Modelling Adaptivity with Aspects”

In the field of web modeling, the work of Baumeister et al. [BKKZ05] was the first to acknowledge the crosscutting nature of customization and to propose the employment of aspect-orientation for modeling customization at the hypertext level. In particular, an extension of UWE's metamodel with aspect-oriented modeling techniques has been proposed and allows making navigation in web applications adaptive (cf. Section 2.2.5). In the UWE metamodel, the Aspect concept has been introduced as a sub-class of the UML meta-class Package. Furthermore an Aspect is defined to have one Pointcut and one Advice each likewise specialized from the UML meta-class Package (cf. Figure 8.1). Moreover, the approach distinguishes between *ModelAspects* that are statically woven and *RuntimeAspects* which are dynamically woven, whereby the latter rely on information which is available at runtime, only.

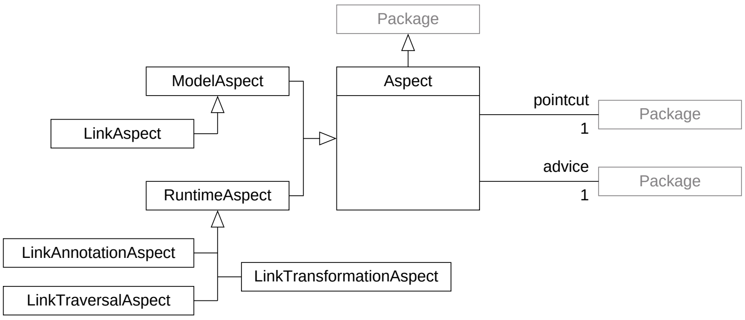


Figure 8.1: Extension of the UWE metamodel with AOM Concepts [BKKZ05]

The approach proposes four types of Aspects:

1. A *LinkAspect* allows adding an UWE *annotation* to a link in the navigation model. The annotation concept is similar to the UML class concept and can have attributes used to store the actual information for a certain link, such as a recommendation level expressed with an integer. In the Pointcut, parts of the *navigation structure diagram* (cf. Section 2.2.5) are modeled in order to identify the links which shall serve as JoinPoints to the Advice. The Advice specifies the *annotation* element to be added.

2. With a *LinkAnnotationAspect*, an attribute of a link's annotation can be changed. In this respect, a LinkAnnotationAspect can be used after a LinkAspect has inserted the annotation to a link. Again in the Pointcut, parts of a navigation structure diagram are modeled to identify the links for which the annotation or rather its attributes shall be updated. In the Advice an optional condition as well as a specification of how to update the annotation can be given in terms of an OCL invariant. For example, the recommendation level could be updated when the link is traversed.

3. A *LinkTraversalAspect* allows modeling updates for the *user model*, i.e., on the user instance when a link is traversed. The links are specified as usual in the Pointcut. In the Advice, the update of the user model is specified using an OCL postcondition.

4. Finally, a *LinkTransformationAspect* provides for adaptive link generation (cf. the *Administrator Links* scenario in Section 2.2.5) and removal. This time, the semantics of the navigation structure model in the Pointcut are different. For link generation, the *navigation classes* are depicted for which a link needs to be inserted. An OCL constraint specifies under which circumstances the link shall be generated, e.g., if the user is member of the administrator user group. In the Advice, the same navigation classes - this time including the link - are depicted, indicating the introduction of the link to the navigation structure diagram. Although not explained in [BKKZ05], it can be assumed that in order to specify the removal of a link, in the Pointcut, the navigation structure model including the link to be deleted is modeled. In the Advice, the navigation classes are modeled without the link.

When comparing the UWE approach to modeling customization with the *aspectWebML* approach, several strengths of the aspectWebML approach can be identified in terms of more expressivity and generality as well as the specification of the composition semantics and tool support.

EXPRESSIVITY.

Comprehensive Join Point Model. Considering separation of concerns, in *aspectWebML*, every element of the WebML language can serve as *JoinPoint* allowing for separation of customization at all levels of a web application. Instead, in the UWE approach, customization can only be separated for the hypertext level, while the user model is inherently tangled with the content level. The join point model of UWE in this respect is quite restrictive by allowing to use links as JoinPoints for *LinkAspects*, *LinkAnnotationAspects*, and *LinkTraversalAspects*, only. In case of *LinkTransformationAspects*, the navigation classes specified in the *Pointcut* serve as JoinPoints.

Several Ways of Quantifying JoinPoints. Considering quantification, UWE supports an enumeration-based quantification method of identifying JoinPoints, where the Pointcut comprises (references to) all model element on whose occurrence the Advice is to be applied [BKKZ05]. In the aspectWebML language in contrast, a Pointcut can either enumerate all model elements through references or declaratively specify an OCL query that returns the specific model elements. In this respect, the use of OCL in aspectWebML has two advantages: First, it allows for reusing Pointcuts within other aspectWebML projects. And second, OCL-based Pointcuts are more robust to changes of the web application model. This means that new model elements that match the query are automatically selected by the Pointcut. In an enumeration-based Pointcut, each JoinPoint would have to be specified explicitly.

Comprehensive Set of AOM concepts. The aspectWebML language has been designed according to the aspectUWA approach, i.e., on the basis of the Conceptual Reference Model for AOM. The CRM has been designed to capture the important concepts of AOM as well as to support different composition mechanisms. Since based on the CRM, the *aspectWebML* language has been extended with aspect-oriented concepts supporting asymmetric as well as symmetric composition mechanisms. In the UWE approach, however, the asymmetric *open class* composition mechanism is considered, only.

Furthermore, several AOM concepts available in the CRM are not supported in the UWE approach. For example, in order to foster reusability of aspect-oriented concepts, in *aspectWebML*, the concepts of *CompositePointcut* as well as *CompositeAdvice* are supported. In the UWE approach, however, it is not clear if a pointcut or an advice can be reused. Moreover, the RelativePosition as well as the EffectKind of Advice are not explicitly considered in the language extension.

An Aspect in *aspectWebML* can consist of more than one Advice as opposed to the UWE approach. This allows for managing complexity through grouping Advice that together realize (part of) a crosscutting concern. In the UWE approach, currently, there are no means of dealing with a possible large number of Advice. Pointcuts in *aspectWebML* are not part of an Aspect but are stored in a PointcutRepository to be available for reuse in the same web application model as well as in future *aspectWebML* projects. Moreover, in UWE it is currently not possible to specify the order of how Aspects shall be applied to the core web application.

GENERALITY.

Instead of introducing dedicated aspects such as a *LinkAspect*, the CRM provides for a general extension of the WebML approach to AOM. This way, the AOM concepts of *aspectWebML* can also be used to separately model other concerns than the customization concern. *aspectWebML*’s expressiveness in terms of thirteen different kinds of AsymmetricCompositionRules which combine a Pointcut, an Advice, a RelativePositionKind and an EffectKind (cf. Chapter 5), allows for modeling, e.g., the evolution of a web application model with Aspects. As an example, consider a web shop selling CD’s and DVD’s which is to be expanded in order to offer books as well. This will require several changes to all levels of a web application. In case the user acceptance for the new book section of the web shop turns out to be unprofitable, however, the books concern can be easily removed by not composing the books concern with the web application. This is similar to the notion of volatile concerns supported by the OOHDM approach [GRUD07] discussed in the following Section 8.3.2.

Specification of Composition Semantics and Tool Support.
Finally, it seems that the composition semantics of the UWE language have not yet been specified, although the modeling notation as well as some results of composing the different kinds of Aspects in UWE have been outlined in [BKKZ05]. Furthermore, the UWE language currently does not provide any tool support for modeling Aspects. In contrast, in the *aspectWebML Modeling Environment*, developers are able to model UWAs and separate the customization concern within aspect(s). Since the composition semantics of the aspectWebML language have been implemented as well as integrated within the aspectWebML tool support, developers are also able to compose the previously defined aspects with the rest of the web application model.

8.2 Hera - ”A Semantics-based Aspect-Oriented Approach to Adaptation in Web Engineering”

The approach of Casteleyn et al. [CWH07] has been introduced only recently and presents the extension of the Hera-S approach, an evolution of the Hera approach, with AOM concepts for modeling customization (cf. Section 2.2.2). In order to separate adaptations from the hypertext level, i.e., Hera’s *application model*, the textual language SEAL - *Semantics-based Aspect-Oriented Adaptation Language* has been designed.

Similar to the UWE approach, a Hera Aspect consists of one Pointcut and one Advice (cf. the *Administrator Links* scenario in Section 2.2.2). For specifying Pointcuts the SEAL language provides a grammar allowing to select a restricted set of 10 *types* from the hypertext level. This set includes ”unit”, ”subunit”, ”attribute”, ”relationship”, ”query”, ”form”, ”label”, ”tour”, ”target”, and ”source”. Besides, the JoinPoints can be selected according to certain *conditions*. For example, types can be selected according to their *name* or according to their aggregation relationships with other types. In addition, string pattern-matching is supported as well as logical operators including conjunction and disjunction. In Hera, models are serialized in RDF(S) and Sesame RDF Query Language[1] (SeRQL) queries, e.g., to define the *application model* on the basis of the *domain model*. Consequently, for weaving purposes, the Pointcuts defined on the basis of SEAL need to be translated into SeRQL queries. In case the offered Pointcut language does not allow to define the required JoinPoints, developers can always fall back on SeRQL.

Furthermore, SEAL provides a grammar for specifying the Advice of an Aspect. Currently four kinds of Advice are distinguished [CWH07].

- A modeler can add an *appearance condition* to the elements specified in the Pointcut using the ADD CONDITION *condition* ”command” (cf. the *Administrator Links* scenario in Section 2.2.2).
- New model elements can be added to the application model depending on a certain condition (e.g., IF *condition* ADD *elements*).

[1] http://www.openrdf.org/doc/sesame/users/

- Likewise, existing elements of the application model selected in the Pointcut can be deleted depending on a certain condition (e.g., IF *condition* DELETE).
- Finally, existing elements of the application model can be replaced, if a certain condition is fulfilled (e.g., IF *condition* REPLACE *element* BY *element*).

Again, when comparing the aspectWebML approach to the Hera approach, several benefits of the aspectWebML approach can be identified, which are explained in the following:

EXPRESSIVITY.

Comprehensive Join Point Model. Similar to the UWE approach, separation of customization with SEAL currently is limited to the hypertext level of web applications and does neither support the content level nor the presentation level. The join point model of SEAL is restricted to the above listed types of the hypertext level, while in aspectWebML every WebML model element can serve as JoinPoint. This provides for separation of customization at all levels of a web application.

Several Ways of Quantifying JoinPoints. With respect to quantification, the Hera approach, allows for selecting the JoinPoints in a declarative manner. Still, the modeler is required to learn a new language, i.e., SEAL, for specifying a Pointcut. In contrast, for the declarative specification of Pointcuts in the aspectWebML approach, a standard query language for models in terms of OCL is used. As already mentioned before, aspectWebML additionally allows for an enumeration-based way of Pointcut definition which can be used in case the developer is not acquainted with OCL.

Comprehensive Set of AOM concepts. As already mentioned before, the main AOM concepts used in the Hera approach are Aspect, Pointcut, and Advice, whereby an Aspect consists of one Pointcut and one Advice. Consequently as for the UWE approach, there is no way of managing complexity in terms of a large number of adaptations through grouping Advice that semantically belong together within an Aspect. Currently, it is neither possible to specify the order of SEAL Aspects. Hera does not incorporate further concepts of the CRM including *CompositePointcuts* or *CompositeAdvice* which would provide for further reusability in the language. In contrast, the aspectWebML language supports the basic ingredients of AOM in terms of modeling concepts covering asymmetric as well as symmetric composition mechanisms, since it has been designed on the basis of the CRM for AOM.

The four kinds of Advice proposed by SEAL provide support for different effects in terms of enhancement, replacement, and deletion of model elements. Nevertheless, reusability is limited when compared to the aspectWebML approach, which specifies the EffectKind within the AsymmetricCompositionRule connecting a Pointcut and an Advice. Furthermore, the aspectWebML language is more expressive in providing thirteen kinds of AsymmetricCompositionRules, which allow the adaptation of modeling elements but also the adaptation of a modeling element's properties such as its name.

SPECIFICATION OF COMPOSITION SEMANTICS AT MODELING LEVEL AND TOOL SUPPORT.

The composition semantics within the Hera approach have already been specified on the basis of *Sesame* - an open source Java framework for storing, querying and reasoning with RDF and RDF Schema. To do so, the domain model and application models are stored in Sesame. In addition, parsers have been generated using the *JavaCC parser generator*[2] in order to translate the Pointcut and Advice parts of Aspects into SeRQL queries. It is not clear if the composed model can be visualized for the modeler, since the corresponding tool support is not yet publicly available. In

[2] https://javacc.dev.java.net/

aspectWebML, the specification of the composition semantics at modeling level is intended to allow for verifying the composition results also at modeling level.

With respect to modeling support, the aspectWebML Modeling environment provides modelers with modeling and composition support as well as means for specifically supporting aspect-oriented modeling such as the OCL console for testing Pointcuts and the Cross References View for viewing relationships between aspect-oriented concepts. Concerning the Hera approach, for specifying SEAL Aspects, a normal text editor basically is enough. Nevertheless, further tool support helping modelers to produce syntactically correct SEAL Aspects or to verify parts of them such as the Pointcut definition currently does not seem to be available.

8.3 General Approaches to AOM in the Web Modeling Domain

This section's focus is a discussion of the current research that is rather widely related to the aspectWebML approach in the sense that AOM has been incorporated into web modeling approaches for other concerns than customization. Following, two further approaches to AOM in the web modeling field are presented.

8.3.1 UWE - Modeling Access Control with Aspects

Besides having extended the UWE metamodel with AOM concepts in order to achieve separation of the customization concern in models, in Zhang et al. [ZBKK05] the UWE approach has been extended to separately model the *access control* aspect in web applications. More specifically, state machines are used for specifying access control behavior. Thus, in the UWE metamodel it is specified that each *navigation class* must have a state machine which specifies the detailed behavior of the navigation class when it is accessed by a user.

In order to allow some access control rule in terms of a state machine to be applied to several navigation classes of the hypertext level, the UWE approach is extended with aspect-oriented modeling concepts. Similar to the previously presented extensions of UWE's metamodel, the Aspect concept is introduced as a sub-class of the UML meta-class Package. Still, no explicit concepts for Pointcut and Advice are incorporated into the metamodel. Instead it is specified, that an Aspect contains (references to) an ordered set of navigation classes, representing the JoinPoints of the Aspect. Since specialized from the meta-class Package, an Aspect is a model element and therefore can have a state machine associated. In this context, the state machine basically represents the Advice. Consequently, the Aspect's state machine, which models access control behavior, applies to all navigation classes specified in the Aspect.

Again, the proposed approach considers the hypertext level of a web application, only. In this respect, navigation classes are the only possible JoinPoints of an Aspect. It is not clear if the approach offers modeling support within the ArgoUwe tool but there is no implementation of the composition semantics. Having presented both of UWE's proposals for aspect-oriented modeling, it has to be noted that, the AOM extensions applied to UWE are tailored to a specific aspect, only, being the access control aspect in [ZBKK05] and the navigation adaptivity aspect in [BKKZ05], respectively. In contrast, the aspectUWA approach is to use the CRM as a blueprint to extend the metamodel of a web modeling language with AOM concepts in order to support separation of the customization concern. Still, the CRM provides the basic ingredients of AOM independent from a specific concern, thus, allowing to model different concerns with one coherent set of concepts. For example, in the aspectWebML language it is also possible to model access control rules within

WebML context clouds of context-aware pages, which then can be separated from the rest of the web application model with Aspects.

8.3.2 OOHDM - Modeling Volatile Functionality

Recently, the OOHDM approach has been extended in order to compose volatile concerns with a core web application [GRUD07]. In the context of this work, a volatile concern might offer some additional services in the web application for a short and determined period of time. As an example, the Amazon shop offered for some products a special link to the Valentine's substore. This link was removed after St. Valentine's day. Another example is the evolution of the web application by the inclusion of an additional service in the web application in order to check users' acceptability.

For supporting such volatile concerns, the OOHDM approach proposes to model each concern separately, i.e., to specify the content, hypertext, and presentation levels of the concern, and provides a textual language for defining the integration of the separate models. This textual notation is similar to the OOHDM node definition syntax already discussed in Section 2.2.4. At hypertext level, a *navigation node* can be extended with links or new information or components from the volatile concern. The integration specifications are called *Affinities*. At presentation level, for each node an Abstract Data View (ADV) can be specified (cf. Section 2.2.4). Again the volatile concern is captured in a separate ADV which needs to be integrated with the core ADV using an integration specification. At content level, no textual integration specification is required. Instead, inversion of control is used in order to ensure that the core functionality is oblivious of the volatile functionality. Thus, the aforementioned integration specifications are available for the hypertext level and the presentation level, only. Summing up, the OOHDM approach provides for the extension of nodes with information and links as well as the extension of ADVs with sub-ADVs. In contrast, the aspectWebML approach, offers developers more expressiveness by allowing all modeling elements of a WebML model to be extended, deleted, or replaced.

Finally, the OOHDM approach is supported by the so-called *Cazon* framework built on top of Apache Struts[3] in order to semi-automatically translate core and volatile OOHDM models into an XML serialization and to compose the concerns at code level. In this respect, the concerns as well as the specifications for their integration which are transformed into XSLT code are kept separately until the code level. In the aspectWebML approach, modeling level composition of concerns is advocated instead. This allows the inspection of the composed model and the subsequent generation of the web application through model transformations and/or code generation reusing existing code generation facilities. Moreover, in the sense of MDE, models shall be used as programs while the actual code shall be hidden from developers. Currently, the OOHDM tool support, i.e., the Cazon framework, is not publicly available, and there seems to be no modeling tool support for graphically specifying crosscutting concerns on the basis of OOHDM's notation either.

8.4 Summary

This chapter has provided a discussion of related work to the aspectWebML approach including closely related approaches to modeling customization of web applications on the basis of AOM

[3] http://struts.apache.org/

concepts as well as widely related approaches extending a web modeling language with AOM concepts for other concerns than customization. The major strengths of the aspectWebML approach when compared to these approaches can be summarized as follows:

First, the aspectWebML approach is *more expressive* than related approaches, due to a more comprehensive join point model allowing any model element of the WebML language to serve as a JoinPoint and thus, for separation of concerns at all levels of the web application model. Since based on the CRM, the aspectWebML language covers a large set of AOM concepts putting a strong emphasis on reusability, e.g., through the inclusion of the CompositePointcut and CompositeAdvice concepts as well as through the specific design of AsymmetricCompositionRules as a combination of a Pointcut, an Advice, a RelativePositionKind, and an EffectKind. aspectWebML even provides the necessary modeling means for changing a modeling elements properties, e.g., its *name* meta-attribute. Furthermore, both, enumeration-based Pointcuts as well as a declarative Pointcuts on the basis of the OCL standard, are supported.

Second, the aspectWebML approach is *more general*, since the aspect-oriented extensions made to the WebML language are independent of the customization concern, possibly allowing to separate arbitrary concerns with Aspects.

And third, with respect to the *specification of the composition semantics and tool support*, aspectWebML advocates and offers composition support at modeling level with the aspectWebML Modeling Environment (cf. Chapter 7), enabling the visualization of the composed model, e.g., for verification purposes, and as a consequence, the reuse of possibly existing code generation facilities.

9 Conclusion

Contents

This chapter gives a brief overview of the work that has been presented in this thesis. The aim of this thesis was to address the problem of *insufficient consideration of the crosscutting nature of customization* in current web modeling languages for designing ubiquitous web applications (UWA). The *aspectUWA* approach presented in this thesis, proposes the application of the ideas of the aspect-orientation paradigm in order to allow for separately modeling customization functionality for UWAs and to profit from typical advantages of separation of concerns, e.g., reduction of complexity, higher maintainability due to better locality of change, as well as reusability. In the following, the major contributions of this thesis are summarized in Section 9.1, while a discussion of current limitations together with an outlook on future research is given in Section 9.2.

9.1 Summary of the Major Contributions of This Thesis

The investigation of the state-of-the-art in modeling UWAs has revealed several weaknesses of current web modeling approaches, including the *insufficient consideration of the crosscutting nature of customization* in current web modeling languages. As a solution to this problem, this thesis has presented *aspectUWA - Applying* ***Aspect****-Orientation to the Model-Driven Development of* ***U****biquitous* ***W****eb* ***A****pplications*. The *aspectUWA* approach proposes the exhaustive use of aspect-orientation as driving paradigm for comprehensively capturing customization separately from all levels of the web application model, i.e., the content level, the hypertext level, and the presentation level. More specifically, *aspectUWA* proposes the general idea of extending any existing web modeling language with concepts from the aspect-orientation paradigm in order to separately model customization functionality and to profit from typical advantages of separation of concerns, e.g., reduction of complexity, higher maintainability due to better locality of change, as well as reusability. In this respect, the so-called *Conceptual Reference Model* (CRM) for Aspect-Oriented Modeling (AOM) allowing the extension of any web modeling language with AOM concepts through a set of extension points has been developed to serve as a generic framework. Furthermore, in order to benefit from the advantages of model-driven development, e.g., higher quality of software products through automation of software development, the *aspectUWA* approach advocates its realization within the realms of model-driven engineering (MDE).

The *aspectUWA* idea has been applied to one web modeling language that supports customization modeling but does not allow modeling customization separately. Being one of the web modeling languages that already provides for a more powerful mechanism for modeling customization but intermingled with the rest of the web application model, the WebML approach has been chosen to be extended with AOM concepts according to the aspectUWA approach which has resulted in the *aspectWebML* language. In this respect, the contributions of this thesis can be summarized as follows:

The Conceptual Reference Model for Aspect-Oriented Modeling. The CRM represents a contribution to the AOM community by providing a taxonomy as well as a conceptual model for AOM. More specifically, the CRM, which has been designed in terms of a UML class diagram, identifies the basic ingredients of AOM and abstracts from different composition mechanisms known in literature while providing as well their refinement in specialized packages of the CRM. The CRM's applicability has been shown in two ways. On the one hand, the CRM has been used as the basis of the aspectUWA approach by capturing the important AOM concepts, their interrelationships, and even more importantly, their relationships to an arbitrary modeling language representing the extension points of the framework. In this respect, can serve as a blueprint when designing new AOM languages or for extending existing (domain-specific) modeling languages with concepts of the aspect-oriented paradigm like it has been shown in this thesis. On the other hand, the CRM has served as a basis for deriving a catalogue of criteria used for the structured evaluation of AOM approaches.

The WebML Metamodel. Since the aspectUWA approach advocates its realization within the realms of MDE, a language specification of the web modeling language to be bridged to AOM needs to be available in terms of a metamodel based on the Meta Object Facility (MOF). In this thesis, a MOF-based metamodel has been designed for the WebML language, since is originally has been partly specified in terms of XML Document Type Definitions (DTD) and partly hard-coded within the tool accompanying the language, i.e., *WebRatio*. Besides its particular necessity in this thesis, a MOF-based WebML metamodel represents an important prerequisite and thus, an initial step towards employing MDE techniques within the WebML approach in general. It also enables interoperability with other MDE tools and is another step towards a common reference metamodel for web modeling languages.

The DTD2MOF Framework. In order to bridge WebML to MDE, the existing DTD-based language specification as well as constraints hard-coded within the language's modeling tool have been reused within a semi-automatic process for metamodel generation from DTDs. The DTD2MOF framework represents a generic framework for semi-automatically generating MOF-based metamodels from arbitrary DTD-based language specifications. Although its design has been motivated by the necessity of developing a metamodel for the WebML language, it has been designed for generality. The work on the DTD2MOF framework includes the elaboration on the *deficiencies of DTDs* when used as means for specifying a modeling language instead of using metamodels. Moreover, a set of *rules and heuristics* for transforming arbitrary DTDs into MOF-based metamodels is provided and appropriate *tool support* for a semi-automatic transformation process from DTD to MOF has been developed. In this respect, the transformation approach enables the "visual" representation of any DTD-based language in terms of MOF-based metamodels and thus, enhances their understandability.

The aspectWebML Web Modeling Language. The *aspectWebML* web modeling language is an extension of the existing WebML language with concepts from the aspect-orientation paradigm in order to better support modeling of crosscutting concerns, i.e., in particular customization, and includes a proposal for a concrete modeling notation for the aspect-oriented concepts introduced. In this respect, the CRM has served as a blueprint for designing the aspectWebML metamodel on top of the WebML metamodel. Thirteen kinds of manipulating WebML models with aspects have been provided going beyond their enhancement of a web application model with modeling elements of the aspect and beyond the replacement as well as the deletion of existing modeling elements of the web application model. More specifically, modelers are also provided with the possibility of changing existing modeling elements' attributes and references. This way, aspectWebML provides ample ways of manipulating web application models with aspects and is not limited to supporting the customization concern. Allowing for OCL-based pointcuts, modelers are enabled to define a repository of reusable pointcuts which can be imported into other aspectWebML projects. In case, modelers are not acquainted with the OCL standard, pointcuts can also be defined be simply enumerating the intended join points in the web application model. The composition semantics of the aspectWebML language have been specified in detail. On the basis of examples, all thirteen ways of defining AsymmetricCompositionRules have been discussed as well as the issues to be considered during composition. Furthermore, an explanation of the mode of operation of the composition algorithm, which has been implemented in Java and is integrated within the modeling environment for aspectWebML, has been given. In this respect, the aspectWebML language is amongst those few AOM approaches already supported with means for composing (crosscutting) concerns.

An Initial Set of Guidelines for Modeling Customization in aspectWebML. On the basis of the case study used to compare the WebML and aspectWebML approaches, an initial set of guidelines to be used for modeling customization in an "aspect-oriented" way within aspectWebML has been presented. Moreover, first extensions to the original WebML development process in order to better support the development of UWAs have been proposed.

The aspectWebML Modeling Environment. Initial modeling support on the basis of a tree-based editor has been made publicly available, together with model examples from the case study as well as further documentation[1]. The preliminary version of the editor has been automatically generated using the Eclipse Modeling Frameworks's (EMF) code generation facilities and has been extended with special features for a better support of developing UWAs with aspectWebML: First, the composition semantics have been implemented in Java and have been integrated within the editor. In this respect, aspectWebML is one of those few AOM approaches that provides modelers not only with tool support for decomposing concerns but also with means for their composition. Second, inspired by the *AspectJ Development Tools*[2], i.e., AspectJ's development environment, a so-called *Cross Reference View* has been implemented and is intended to ease AOM by visualizing interrelationships, e.g., between aspects and the web application model. And third, though based on a tree-editor, the aspectWebML Modeling Environment provides first modeling support for WebML's customization modeling concepts, which are not yet considered within the WebRatio tool.

[1] *www.wit.at/people/schauerhuber/aspectUWA*

[2] www.eclipse.org/ajdt/

9.2 Current Limitations and Outlook

In this thesis, the ground has been paved for modeling customization with WebML in an aspect-oriented way, by providing the technical support in terms of the aspectWebML metamodel, a proposal for the modeling elements' notation, as well as the necessary semantics allowing to compose previously separated concerns. Nevertheless, several limitations with respect to the aspectWebML language need to be tackled in future work. This includes achieving full support of aspect-oriented concepts and composition mechanisms defined in the CRM (cf. Section 9.2.1 and Section 9.2.2) as well as the provision of a development process (cf. Section 9.2.3) and a notation (cf. Section 9.2.4) that allow to easily cope with the shift in paradigm imposed on modelers when introducing aspect-oriented modeling concepts. In this respect, appropriate tools are required as well in order to support modelers when designing ubiquitous web applications in an aspect-oriented way (cf. Section 9.2.5). Furthermore, in the sense of MDE, modelers shall be provided with an integrated modeling environment allowing also for generating the final web applications through model transformation and/or code generation (cf. Section 9.2.6 and Section 9.2.7).

With respect to the aspectUWA approach, an interesting direction for future work is its application to other web modeling approaches for modeling the customization concern (cf. Section 9.2.9) in particular as well as arbitrary concerns in general. In this respect, the aspectWebML language can serve as a first candidate for testing the generality of the aspect-oriented concepts introduced on the basis of the CRM (cf. Section 9.2.8).

9.2.1 Supporting Aspect Dependencies and Interactions

In the current version of the aspectWebML language, the issue of *aspect dependencies and interactions*, which is supported by the CRM through the *module interaction* and *rule interaction* concepts, is currently not (fully) considered. This means, that modelers are not able to explicitly indicate interactions between concerns in their models. Still, in a paradigm such as aspect-orientation where different "components" are configured in order to form a working application (model), knowing about possible dependencies and interactions between these components is crucial during development. As a consequence, it will be subject to future work, to find out what kinds of interactions (e.g., dependency, conflict, mutual exclusion) are relevant for the aspectWebML language and thus need to be supported.

Still, some form of conflict resolution is already provided in aspectWebML by allowing modelers to explicitly specify the order of composing concerns. Moreover, in the implementation of the composition semantics in the aspectWebML tool support, dependencies between Advice as well as between Aspects can already be computed and visualized in the Cross References View of the aspectWebML Modeling Environment.

9.2.2 Providing for Asymmetric as well as Symmetric Composition

The CRM has been designed such that it abstracts from several composition mechanisms and at the same time specializes the aspect-oriented concepts in order to support the *pointcut-advice* and *open class* asymmetric composition mechanisms as well as the *compositor* symmetric composition mechanism. This particular design allows language designers to decide on implementing both kinds of composition mechanisms or only one of them while being able to easily complete the language with the other one later on.

Due to the special goal of separately modeling customization in UWAs, in this thesis the focus has been on asymmetric composition. Customization functionality is a crosscutting concern that needs to be defined on top of existing models. In this respect, it represents a concern that cannot exist on its own, meaning that it has to be applied to the functionality of a web application. As a consequence, asymmetric composition mechanisms which distinguish between core concerns and crosscutting concerns are more suitable to support customization modeling in the development of UWAs than symmetric composition mechanisms. Nevertheless, supporting both asymmetric as well as symmetric composition mechanisms allows for a powerful language. Therefore, the technical support for both kinds of composition mechanisms is already given in the aspectWebML language through the extension of the WebML metamodel with aspect-oriented concepts for both mechanisms. Still, the concepts for modeling concerns according to the symmetric composition mechanism need to be refined and the composition semantics for symmetric composition need to be specified. Furthermore, it is still unclear how exactly symmetric composition can be used in aspectWebML for modeling (crosscutting) concerns. This includes an appropriate notation in terms of dedicated diagram types as well as a new set of guidelines for designing separate concerns and composing them. And finally, further research is required in order to find out if and how both kinds of composition mechanisms can be successfully used in parallel.

With respect to asymmetric composition, the aspectWebML language currently provides full support of the asymmetric open class composition mechanism, only. While the open class composition mechanism is used for enhancing, replacing, or deleting aspectual structure in a model (e.g., add an attribute to class), the pointcut-advice composition mechanism, is used for enhancing, replacing, or deleting aspectual behavior in a model (e.g., intercept a method call and do something before the call). With respect to the WebML approach, the language mainly supports modeling the structural features of a web application, for which the open class composition mechanism is enough. Still, some kind of behavior can be modeled by introducing content management functionality to a WebML model, e.g., in terms of model elements realizing typical CRUD operations. In this case, some flow of actions is available through connecting these *operation units* with *links*. Consequently, applying aspectual behavior to such a flow of actions, e.g., before or after a certain point in the flow, corresponds to using the pointcut-advice asymmetric composition mechanism. Since different types of *operation units* can have a different number of incoming and outgoing *links*, using the pointcut-advice mechanism might have unexpected side-effects on the flow of actions and might change the semantics of the content management functionality. Consequently, behavioral flows in WebML are currently ignored and the WebML language is considered from a structural view point using the open class composition mechanism, only. Nevertheless, future research will need to investigate the suitability of AspectJ-like pointcut designators for allowing to better specify the join points of a pointcut used in the pointcut-advice composition mechanism.

9.2.3 Designing an aspectWebML Development Process

The WebML language does not yet provide a development process dedicated to the development of UWAs. In this thesis, first extensions to the original development process have been proposed including the extension of the schema of WebML's *user group description sheets* which are used in the *requirements specification* phase. More specifically, the schema has been extended to consider the *customization scenarios* in which the given user group is involved and the usual *context* in which the user group is accessing the web application. Furthermore, a set of guidelines for designing Aspects in the aspectWebML language has been provided, e.g., to start with one Aspect per cus-

tomization scenario and then strive for the design of reusable Aspects.
Nevertheless, developing a customization scenario in aspectWebML from scratch is a difficult task even for experienced aspectWebML modelers. For example, in order to design a customization scenario often many Advice need to be defined. As a consequence, future work will include the extension of the current set of guidelines to help modelers to quickly absorb the new modeling paradigm as well as the integration of this set of guidelines with the original WebML development process. Furthermore, appropriate tool support shall help modelers in using the aspectWebML concepts (cf. Section 9.2.5). The proposed development process for the aspectWebML approach is required to be tested within case studies and real-world applications. Still, this also raises currently unaddressed questions associated with empirical evaluations in web engineering, e.g., how to get an unbiased set-up for an evaluation including control groups.

9.2.4 Evaluating the Notation of aspectWebML

The notation of aspectWebML resembles UML class diagrams in many ways, e.g., by using compartments. Furthermore, for representing WebML modeling concepts the corresponding notation and icons of the WebML language has been reused. For the aspect-oriented concept in the aspectWebML language, new icons have been designed, whereby a sub-set has been inspired by the notation for aspect-oriented concepts used in the *AspectJ Development Tools*. The current proposal for a notation, however, needs to be evaluated with respect to usability and intuitiveness requirements. Since, the introduction of the aspect-orientation concepts to the WebML language actually represent a shift in paradigm when modeling UWAs, usability and intuitiveness are important factors for the success of the aspectWebML approach. For example, it is currently, not clear if the ″reuse″ of the AspectJ notation, is appropriate for modelers that possibly never have come into contact with aspect-oriented programming before. Accordingly, the notation needs to be exhaustively tested and refined within case studies and subsequently within real-world applications.

9.2.5 A Graphical Integrated Development Environment for aspectWebML

Currently, the aspectWebML Modeling Environment is based on a tree-editor built on top of the EMF. While EMF editors have proved their usefulness for providing proof-of-concept prototypes, their cumbersome handling certainly does hamper the design of large scale models. Consequently, it is necessary to extend the existing modeling support with further ways of graphically visualizing aspect-oriented concepts in aspectWebML that go beyond a simple tree-based editor. This advanced modeling support shall again be realized on the basis of the Eclipse Graphical Modeling Framework[3] [Pre07]. In this thesis, some graphical views and/or editors dedicated to visualizing the aspect-oriented concepts in an aspectWebML model already have been proposed.

Furthermore, besides providing graphical visualization, modelers will have to be supported by reducing the effort when modeling aspect-oriented concepts. As already pointed out before, modeling a customization scenario from scratch within an Aspect is a difficult task. As a consequence, appropriate tool support is essential for the adoption of the aspectWebML approach. For example, the designer should be able to start modeling a scenario without using aspectWebML concepts, i.e., intermingled with the rest of the web application model. In a second step, the modeler could then select some model elements that realize customization functionality and encapsulate them within an Aspect using a so-called ″Add to Aspect″ feature of the tool. Likewise, modelers could

[3] http://www.eclipse.org/gmf/

be supported with a feature that automatically generates OCL-based Pointcuts from a set of selected model elements.

9.2.6 Model-Driven Engineering with aspectWebML

In order to allow for a model-driven development of web applications, future work will have to strive for providing the semi-automatic generation of ubiquitous web applications in the sense of MDE, i.e., to provide for model transformation and/or code generation. In this respect, two options can be distinguished, namely the integration with the existing WebRatio tool (cf. Section 9.2.7) or the extension of the aspectWebML Modeling Environment to support MDE on the basis of model transformations and/or code generation. Concerning the latter option, again MDE techniques developed under the hood of the Eclipse project could serve as a basis. For example, the Atlas Transformation Language (ATL), developed by the team of researchers of Jean Bézivin at the University of Nantes, is a model transformation language for Ecore-based metamodels and recently has been declared as part of the Eclipse model-to-model (M2M) transformation project[4]. With respect to code generation, several approaches including MOFScript as well as openArchitectureWare (oAW) are available within the Eclipse Generative Modeling Technologies (GMT) project[5].

Besides, an interesting direction for future work is to not compose Aspects at modeling level but to maintain Aspects until the programming level, i.e., to use as target platform for code generation an aspect-oriented programming language like AspectJ. In this respect, an open question is if a mapping of modeling level Aspects of the aspectWebML language to programming level Aspects is possible and reasonable.

9.2.7 Integration of the aspectWebML Modeling Environment with WebRatio

With respect to exploiting existing code generation facilities of WebRatio through the integration of the aspectWebML Modeling Environment with WebRatio, it has to be noted that a full integration is difficult to achieve for the following reasons:

First, since WebRatio is a commercial tool for which the source code is not available, a direct integration of the aspectWebML language within the WebRatio tool currently is not possible. Furthermore, the WebML models created with WebRatio are still serialized in XML according to the WebML DTD. As a consequence, a possible solution for integration is to provide import/export facilities in the aspectWebML tool support, which allows for exchanging WebML models between WebRatio and the aspectWebML Modeling Environment. This way, modelers may (i) develop their web application models in WebRatio, (ii) import the WebML model into the ModuleRepository of the aspectWebML project in the aspectWebML Modeling Environment, (iii) specify and compose crosscutting concerns in the aspectWebML Modeling Environment, and (iv) export the composed model to the WebRatio tool for code generation purposes.

Second, the WebML extensions allowing for customization modeling have been recently defined, only. As a consequence, the necessary tool support for modeling customization has not yet been integrated within WebRatio. This means that WebRatio neither allows for modeling customization nor for generating code for UWAs. It is therefore not possibly to export WebML models from the aspectWebML Modeling Environment that contain instances of WebML's customization

[4] http://www.eclipse.org/m2m/

[5] http://www.eclipse.org/gmt/

modeling concepts. Nevertheless, as will pointed out in the next section, the aspectWebML approach should allow modelers to generally model other concerns than customization, although they are restricted to using WebML concepts originally defined in the WebML DTD.

9.2.8 Modeling Arbitrary Aspects in aspectWebML

In contrast, to related aspect-oriented web modeling languages, the aspectWebML language has not been exclusively designed for the customization concern but to support generic aspect-oriented concepts as described in the CRM, which then are used to separately model customization. As a consequence, an interesting direction for future work is to investigate in what ways the aspect-oriented extensions made to WebML can be used to model other concerns than customization, the more, since customization modeling is not yet supported within the WebRatio tool. As already pointed out in previous sections, aspectWebML could be used e.g. for the evolution of a web application model with Aspects. For example, a web shop selling CD's and DVD's is to be expanded in order to offer books as well. This will require several changes to all levels of a web application. In case the user acceptance for the new book section of the web shop turns out to be unprofitable, however, the books concern can be easily removed by not composing the books concern with the web application. In this respect, aspectWebML could be used to model different features of a web application within several Aspects.

9.2.9 Applying aspectUWA to Other Web Modeling Approaches

The aspectUWA approach proposes a generic framework for extending existing web modeling languages with AOM concepts and has been applied to the WebML approach in this thesis. A further direction for future research is to further prove the aspectUWA approach's, or rather, the CRM's generality through its application to other web modeling languages. In this respect, the application to the UWE approach as well as to the Hera approach would be of particular interest, since they have already been extended with aspect-oriented concepts. In this vein, their ways of introducing aspect-oriented modeling concepts can be better compared with the aspectUWA way. Since based on the UML, the application of the CRM to the UWE approach is also interesting due to UML's ample ways of behavioral modeling. In this respect, existing ways of composing aspectual behavior already proposed by some general-purpose, UML-based AOM approaches investigated in this thesis could be reused or serve as valuable input. As a long term goal, the CRM's generality should also be investigated in other domains than the web modeling domain.

Bibliography

[ABV92] Mehmet Aksit, Lodewijk Bergmans, and Sinan Vural. An Object-Oriented Language-Database Integration Model: The Composition-Filters Approach. In *Proc. of the 6th European Conference on Object-Oriented Programming (ECOOP'92), Utrecht, The Netherlands*, volume 615 of *LNCS 615*, pages 372–395, 1992.

[AEB03] Omar Aldawud, Tzilla Elrad, and Atef Bader. UML Profile for Aspect-Oriented Software Development. In *3rd International Workshop on Aspect Oriented Modeling (AOM'03), in conjunction with AOSD'03, Boston, Massachusetts*, March 2003.

[AFGP02] Silvia Mara Abrahão, Joan Fons, Magalí González, and Oscar Pastor. Conceptual Modeling of Personalized Web Applications. In *Proc. of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Malaga, Spain*, LNCS 2347, pages 358–362, 2002.

[AK03] Colin Atkinson and Thomas Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41, 2003.

[BBR$^+$05] Gordon S. Blair, Lynne Blair, Awais Rashid, Ana Moreira, João Araújo, and Ruzanna Chitchyan. Engineering Aspect-Oriented Systems. In R.E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 379–406. Addison-Wesley, Boston, 2005.

[BCC$^+$03] Marco Brambilla, Stefano Ceri, Sara Comai, Piero Fraternali, and Ioana Manolescu. Specification and Design of Workflow-Driven Hypertexts. *J. Web Eng.*, 1(2):163–182, 2003.

[BCC$^+$06] Marco Brambilla, Irene Celino, Stefano Ceri, Dario Cerizza, Emanuele Della Valle, and Federico Michele Facca. A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *Proc. of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA*, LNCS 4273, pages 172–186, November 2006.

[BCF02] Marco Brambilla, Sara Comai, and Piero Fraternali. Hypertext Semantics for Web Applications. In *Proc. of the 10th Italian National Symposium on Advanced DataBase Systems (SEBD), Portoferraio, Italy*, pages 73–86, 2002.

[BCFC06] Alessandro Bozzon, Sara Comai, Piero Fraternali, and Giovanni Toffetti Carughi. Conceptual Modeling and Code Generation for Rich Internet Applications. In *Proc. of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, CA, USA*, pages 353–360. ACM, July 2006.

[BCFK99] Grady Booch, Magnus Christerson, Matthew Fuchs, and Jari Koistinen. UML for XML Schema Mapping Specification. Technical report, Rational Software and CommerceOne, August 1999.

[BCFM06] Marco Brambilla, Stefano Ceri, Piero Fraternali, and Ioana Manolescu. Process Modeling in Web Applications. *ACM Trans. Softw. Eng. Methodol.*, 15(4):360–409, 2006.

[BCMM06] Luciano Baresi, Sebastiano Colazzo, Luca Mainetti, and Sandro Morasca. W2000: A Modeling Notation for Complex Web Applications. In Emilia Mendes and Nile Mosley, editors, *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*, pages 335–364. Springer, 2006.

[BFH02] Peter Barna, Flavius Frasincar, and Geert-Jan Houben. Specification Framework for Engineering Adaptive Web Applications. In *Proc. of the 11th International World Wide Web Conference, Web Engineering Track (WWW 2002), Honolulu, Hawaii, USA*, May 2002.

[BFH06] Peter Barna, Flavius Frasincar, and Geert-Jan Houben. A Workflow-driven Design of Web Information Systems. In *Proc. of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, CA, USA*, pages 321–328. ACM, July 2006.

[BFHV03] Peter Barna, Flavius Frasincar, Geert-Jan Houben, and Richard Vdovjak. Methodologies for Web Information System Design. In *Proc. of the International Conference on Information Technology: Computers and Communications (ITCC 2003), Las Vegas, NV, USA*, pages 420–424. IEEE Computer Society, April 2003.

[BFK$^+$00] B. R. Badrinath, Armando Fox, Leonard Kleinrock, Gerald J. Popek, Peter L. Reiher, and Mahadev Satyanarayanan. A conceptual framework for network and client adaptation. *Mobile Networks and Applications*, 5(4):221–231, 2000.

[BKKZ05] Hubert Baumeister, Alexander Knapp, Nora Koch, and Gefei Zhang. Modelling Adaptivity with Aspects. In *Proc. of the 5th International Conference on Web Engineering (ICWE 2005), Sydney, Australia*, LNCS 3579, pages 406–416, July 2005.

[BKM99] Hubert Baumeister, Nora Koch, and Luis Mandel. Towards a UML Extension for Hypermedia Design. In *Proc. of the 2nd International Conference on the Unified Modeling Language (UML 1999), Fort Collins, CO, USA*, LNCS 1723, pages 614–629, October 1999.

[BL03] Chris Barry and Michael Lang. A comparison of 'traditional' and multimedia information systems development practices. *Information & Software Technology*, 45(4):217–227, 2003.

[BLW05] Paul Baker, Shiou Loh, and Frank Weil. Model-Driven Engineering in a Large Industrial Context - Motorola Case Study. In *Proc. of the 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005), Montego Bay, Jamaica*, LNCS 3713. Springer, October 2005.

[BM02] Luciano Baresi and Franca Garzotto Monica Maritati. W2000 as a MOF metamodel. In *Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2002)*, July 2002.

[BSM$^+$04] Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, and Timothy J. Grose. *Eclipse Modeling Framework.* Addison-Wesely, 1st edition, 2004.

[CAE07] Thomas Cottenier, Aswin van den Berg, and Tzilla Elrad. The Motorola WEAVR: Model Weaving in a Large Industrial Context. In *Proc. of the 6th International Conference on Aspect-Oriented Software Development (AOSD 2007), Vancouver, Canada*, March 2007.

[CB05] Siobhán Clarke and Elisa Banaissad. *Aspect-Oriented Analysis and Design The Theme Approach*. Addison-Wesley, Upper Saddle River, March 2005.

[CDF06] Stefano Ceri, Florian Daniel, and Federico Michele Facca. Modeling Web Applications reacting to User Behaviors. *Computer Networks*, 50(10):1533–1546, 2006.

[CDFM07] Stefano Ceri, Florian Daniel, Federico Michele Facca, and Maristella Matera. Model-Driven Engineering of Active Context-Awareness. *World Wide Web*, in print, 2007.

[CdL95] Donald D. Cowan and Carlos José Pereira de Lucena. Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse. *IEEE Trans. Software Eng.*, 21(3):229–243, 1995.

[CDM03] Stefano Ceri, Florian Daniel, and Maristella Matera. Extending WebML for Modeling Multi-Channel Context-Aware Web Applications. In *Proc. of the Workshop on Mobile Multi-channel Information Systems, in conjuntion with WISE 2003, Rome, Italy*, pages 615–626, December 2003.

[CDMF07] Stefano Ceri, Florian Daniel, Maristella Matera, and Federico Michele Facca. Model-driven Development of Context-Aware Web Applications. *ACM Transactions on Internet Technology*, 7(1), 2007.

[CF01] Sara Comai and Piero Fraternali. A semantic model for specifying data-intensive Web applications using WebML. In *Proc. of the 1st Semantic Web Working Symposium (SWWS'01), Stanford University, CA, USA*, pages 566–585, July/August 2001.

[CFB$^+$03] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, and Maristella Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 1st edition, 2003.

[CGP00] Cristina Cachero, Jaime Gómez, and Oscar Pastor. Object-Oriented Conceptual Modeling of Web Application Interfaces: the OO-HMethod Abstract Presentation Model. In *Proc. of the 1st International Conference on Electronic Commerce and Web Technologies (EC-Web 2000), London, UK*, LNCS 1875, pages 206–215, September 2000.

[CGP01] Cristina Cachero, Jaime Gómez, and Antonio Párraga. Migration of Legacy Systems to the Web. In *VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2001), Almagro (Ciudad Real)*, pages 601–614, November 2001.

[CGPP01] Cristina Cachero, Jaime Gómez, Antonio Párraga, and Oscar Pastor. Conference Review System: A Case of Study. In *1st International Workshop on Web-Oriented Software Technology (IWWOST 2001), Valencia, Spain*, June 2001.

[CGT05] Sven Casteleyn, Irene Garrigós, and Olga De Troyer. Automatic Runtime Validation and Correction of the Navigational Design of Web Sites. In *7th Asia-Pacific Web Conference on Web Technologies Research and Development (APWeb 2005), Shanghai, China*, LNCS 3399, pages 453–463, March/April 2005.

[Che76] Peter P. Chen. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[CHOT99] Siobhán Clarke, William Harrison, Harold Ossher, and Peri Tarr. Subject-Oriented Design: Towards Improved Alignment of Requirements, Design and Code. In *Proc. of the 14th Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 1999), Denver, Colorado, USA*, pages 325–339, November 1999.

[CJ06] Siobhán Clarke and Andrew Jackson. Refined AOD Process. Technical Report D57 AOSD-Europe-TCD-D57, AOSD-Europe, August 2006.

[Cla01] Siobhán Clarke. *Composition of Object-Oriented Software Design Models*. PhD thesis, Dublin City University, January 2001.

[Cla02] Siobhán Clarke. Extending Standard UML with Model Composition Semantics. *Science of Computer Programming*, 44(1):71–100, July 2002.

[CM06] Wesley Coelho and Gail C. Murphy. Presenting Crosscutting Structure with Active Models. In *Proc. of the 5th International Conference on Aspect-Oriented Software Development (AOSD 2006), Bonn, Germany*, pages 158–168, March 2006.

[Con02] Jim Conallen. *Building Web Applications with UML*. Addison-Wesely, 2nd edition, 2002.

[CPT06] Sven Casteleyn, Peter Plessers, and Olga De Troyer. On Generating Content and Structural Annotated Websites Using Conceptual Modeling. In *Proc. of the 25th International Conference on Conceptual Modeling (ER 2006), Tucson, AZ, USA*, LNCS 4215, pages 267–280, November 2006.

[CRS+05] Ruzanna Chitchyan, Awais Rashid, Pete Sawyer, Alessandro Garcia, Mónica Pinto Alarcon, Jethro Bakker, Bedir Tekinerdoğan, Siobhán Clarke, and Andrew Jackson. Survey of Aspect-Oriented Analysis and Design Approaches. Technical Report D11 AOSD-Europe-ULANC-9, AOSD-Europe, May 2005.

[CTB03] Sven Casteleyn, Olga De Troyer, and Saar Brockmans. Design Time Support for Adaptive Behavior in Web Sites. In *Proc. of the 18 th Symposium on Applied Computing (SAC 2003), Melbourne, FL, USA*, pages 1222–1228, March 2003.

[CvdBE07] Thomas Cottenier, Aswin van den Berg, and Tzilla Elrad. Joinpoint Inference from Behavioral Specification to Implementation. In *Proc. of the 21st European Conference on Object-Oriented Programming (ECOOP 2007), Berlin, Germany*, LNCS 4609, pages 476–500, July/August 2007.

[CW05] Siobhán Clarke and Robert J. Walker. Generic Aspect-Oriented Design with Theme/UML. In R.E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 425–458. Addison-Wesley, Boston, 2005.

[CWH07] Sven Casteleyn, William Van Woensel, and Geert-Jan Houben. A Semantics-based Aspect-Oriented Approach to Adaptation in Web Engineering. In *Proceedings of the 18th Conference on Hypertext and Hypermedia (HT 2007), Manchester, UK*, pages 189–198, September 2007.

[dbTB$^+$06] Steven Op de beeck, Eddy Truyen, Nelis Boucké, Frans Sanen, Maarten Bynens, and Wouter Joosen. A Study of Aspect-Oriented Design Approaches. Technical Report CW435, Department of Computer Science, Katholieke Universiteit Leuven, February 2006.

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[EAB05] Tzilla Elrad, Omar Aldawud, and Atef Bader. Expressing Aspects Using UML Behavioral and Structural Diagrams. In R.E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 459–478. Addison-Wesley, Boston, 2005.

[EK04] María José Escalona and Nora Koch. Requirements Engineering for Web Applications - A Comparative Study. *Journal of Web Engineering*, 2(3):193–212, 2004.

[EVP01] Jacob Eisenstein, Jean Vanderdonckt, and Angel R. Puerta. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In *Proc. of the 6th International Conference on Intelligent User Interfaces (IUI 2001), Santa Fe, New Mexico, United States*, pages 69–76, January 2001.

[FBHF04] Flavius Frasincar, Peter Barna, Geert-Jan Houben, and Zoltán Fiala. Adaptation and Reuse in Designing Web Information Systems. In *Proc. of the International Conference on Information Technology: Coding and Computing (ITCC 2004), Las Vegas, Nevada, USA*, pages 387–291, April 2004.

[FECA05] Robert E. Filman, Tzilla Elrad, Siobhán Clarke, and Mehmet Akşit, editors. *Aspect-Oriented Software Development*. Addison-Wesley, Boston, 2005.

[FFH$^+$04] Zoltán Fiala, Flavius Frasincar, Michael Hinz, Geert-Jan Houben, Peter Barna, and Klaus Meißner. Engineering the Presentation Layer of Adaptable Web Information Systems. In *Proc. of the 4th International Conference on Web Engineering (ICWE 2004), Munich, Germany*, volume 3140 of *LNCS 3140*, pages 459–472, July 2004.

[FH02] Flavius Frasincar and Geert-Jan Houben. Hypermedia Presentation Adaptation on the Semantic Web. In *Proc. of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Malaga, Spain*, LNCS 2347, pages 133–142, May 2002.

[FHB06] Flavius Frasincar, Geert-Jan Houben, and Peter Barna. HPG: the Hera Presentation Generator. *Journal of Web Engineering*, 5(2):175–200, 2006.

[FHHF04] Zoltán Fiala, Michael Hinz, Geert-Jan Houben, and Flavius Frasincar. Design and Implementation of Component-based Adaptive Web Presentations. In *Proc. of the Symposium on Applied Computing (SAC 2004), Nicosia, Cyprus, 2004*, pages 1698–1704, March 2004.

[FHV01] Flavius Frasincar, Geert-Jan Houben, and Richard Vdovjak. An RMM-Based Methodology for Hypermedia Presentation Design. In *Proc. of the 5th East European Conference on Advances in Databases and Information Systems (ADBIS 2001), Vilnius, Lithuania*, LNCS 2151, pages 323–337, September 2001.

[FKGS04] Robert B. France, Dae-Kyoo Kim, Sudipto Ghosh, and Eunjee Song. A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering*, 30(3):193–206, 2004.

[FPAP03] Joan Fons, Vicente Pelechano, Manoli Albert, and Oscar Pastor. Development of Web Applications from Web Enhanced Conceptual Schemas. In *Proc. of the 22nd International Conference on Conceptual Modeling (ER 2003), Chicago, IL, USA*, LNCS 2813, pages 232–245, October 2003.

[FPT07] Lidia Fuentes, Mónica Pinto, and José M. Troya. Supporting the Development of CAM/DAOP Applications: An Integrated Development Process. *Software - Practice and Experience*, 37(1):21–64, 2007.

[FRGG04] Robert France, Indrakshi Ray, Geri Georg, and Sudipto Ghosh. Aspect-oriented Approach to Early Design Modelling. *IEE Proceedings Software*, 151(4):173– 185, August 2004.

[GBP05] Jaime Gómez, Alejandro Bia, and Antonio Párraga. Tool Support for Model-Driven Development of Web Applications. In *6th International Conference on Web Information Systems Engineering (WISE 2005), New York, NY, USA*, LNCS, 3806, pages 721–730, November 2005.

[GCG05] Irene Garrigós, Sven Casteleyn, and Jaime Gómez. A Structured Approach to Personalize Websites Using the OO-H Personalization Framework. In *Proc. of the 7th Asia-Pacific Web Conference on Web Technologies Research and Development (APWeb 2005), Shanghai, China*, LNCS 3399, pages 695–706, March/April 2005.

[GCG07] Irene Garrigós, Cristian Cruz, and Jaime Gómez. A Prototype Tool for the Automatic Generation of Adaptive Websites. In *Proc. of the 2nd International Workshop on Adaptation and Evolution in Web Systems Engineering (AEWSE 2007), in conjunction with ICWE 2007, Como, Italy*. CEUR Workshop Proceedings, July 2007.

[GCP00] Jaime Gómez, Cristina Cachero, and Oscar Pastor. Extending a Conceptual Modelling Approach to Web Application Design. In *Proc. of the 12th International Conference on Advanced Information Systems Engineering (CAISE 2000), Stockholm, Sweden*, LNCS 1789, pages 79–93, June 2000.

[GCP01] Jaime Gómez, Cristina Cachero, and Oscar Pastor. Conceptual Modeling of Device-Independent Web Applications. *IEEE MultiMedia*, 8(2):26–39, 2001.

[GG06] Irene Garrigós and Jaime Gómez. Modeling User Behaviour Aware WebSites with PRML. In *3rd Workshop on Web Information Systems Modelling (WISM 2006), in conjunction with ICWE 2006, Palo Alto, California, USA*, July 2006.

[GGBH05] Irene Garrigós, Jaime Gómez, Peter Barna, and Geert-Jan Houben. A Reusable Personalization Model in Web Application Design. In *2nd Workshop on Web Information Systems Modelling (WISM 2005), in conjunction with ICWE 2005, Sydney, Australia*, July 2005.

[GGC03a] Irene Garrigós, Jaime Gómez, and Cristina Cachero. Modelling Adaptive Web Applications. In *Proc. of the IADIS International Conference WWW/Internet, Algarve, Portugal*, pages 813–816, November 2003.

[GGC03b] Irene Garrigós, Jaime Gómez, and Cristina Cachero. Modelling Dynamic Personalization in Web Applications. In *Proc. of the 3rd International Conference on Web Engineering (ICWE 2003), Oviedo, Spain*, LNCS 2722, pages 472–475, July 2003.

[GHHV04] Erich Gamma, Richard Helm, Ralph Hohnson, and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesely, 2004.

[GPS93] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):1–26, 1993.

[Gru00] John Grundy. Multi-Perspective Specification, Design and Implementation of Software Components Using Aspects. *International Journal of Software Engineering and Knowledge Engineering*, 20(6), 2000.

[GRUD07] Jeronimo Ginzburg, Gustavo Rossi, Matias Urbieta, and Damiano Distante. Transparent Interface Composition in Web Applications. In *Proc. of the 7th International Conference on Web Engineering (ICWE 2007), Como, Italy*, LNCS 4607, pages 152–166, July 2007.

[Hal01] Terry Halpin. *Information Modeling and Relational Databases*. Morgan Kaufmann, 2001.

[Han05] Stefan Hanenberg. *Design Dimensions of Aspect-Oriented Systems*. PhD thesis, University Duisburg-Essen, October 2005.

[HBI98] Anna Hester, Renato Borges, and Roberto Ierusalimschy. Building Flexible and Extensible Web Applications with Lua. *Journal of Universal Computer Science*, 4(9):748–762, 1998.

[HFBV04] Geert-Jan Houben, Flavius Frasincar, Peter Barna, and Richard Vdovjak. Modeling User Input and Hypermedia Dynamics in Hera. In *Proc. of the 4th International Conference on Web Engineering, Munich, Germany*, LNCS 3140, pages 60–73. Springer, 2004.

[HJPP02] Wai-Ming Ho, Jean-Marc Jézéquel, François Pennaneac'h, and Noël Plouzeau. A Toolkit for Weaving Aspect Oriented UML Designs. In *Proc. of the 1st International Conference on Aspect-oriented Software Development (AOSD 2002), Enschede, The Netherlands*, pages 99–105, April 2002.

[HO93] William H. Harrison and Harold L. Ossher. Subject-Oriented Programming - A Critique of Pure Objects. In *Proc. of the 8th Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 1993), Washington, DC, USA*, pages 411–428, September 1993.

[HOT02] William H. Harrison, Harold L. Ossher, and Peri L. Tarr. Asymmetrically vs. Symmetrically Organized Paradigms for Software Composition. Technical report, IBM Research Division, Thomas J. Watson Research Center, December 2002.

[ISB95] Tomás Isakowitz, Edward A. Stohr, and P. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8):34–44, 1995.

[Jac90] Michael Jackson. Some Complexities in Computer-Based Systems and Their Implications for System Development. In *Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering (CompEuro 1990)*, pages 344–351. IEEE computer Society Press, May 1990.

[JC06] Andrew Jackson and Siobhán Clarke. Towards the Integration of Theme/UML and JPDDs. In *Proc. of the 8th International Workshop on Aspect-Oriented Modeling (AOM), in conjunction with AOSD 2006, Bonn, Germany*, March 2006.

[JK06] Frédéric Jouault and Ivan Kurtev. Transforming Models with ATL. In *Proc. of the Workshop on Model Tranformations in Practice (MTiP) Montego Bay, Jamaica*, LNCS 3844, pages 128–138, October 2006.

[JKBC06] Andrew Jackson, Jacques Klein, Benoit Baudry, and Siobhán Clarke. KerTheme: Testing Aspect Oriented Models. In *Proc. of the Workshop on Integration of Model Driven Development and Model Driven Testing, in conjunction with ECMDA 2006, Bilbao, Spain*, July 2006.

[JN05] Ivar Jacobson and Pan-Wei Ng. *Aspect-Oriented Software Development with Use Cases*. Addison-Wesley, 2005.

[JSR02] Mark D. Jacyntho, Daniel Schwabe, and Gustavo Rossi. A Software Architecture for Structuring Complex Web Applications. *J. Web Eng.*, 1(1):37–60, 2002.

[Ker05] Mik Kersten. AOP Tools Comparison (Part 1 & 2). http://www-128.ibm.com/developerworks/java/library/j-aopwork1/, March 2005.

[KFG04] Dae-Kyoo Kim, Robert B. France, and Sudipto Ghosh. A UML-based Language for Specifying Domain-Specific Patterns. *Journal of Visual Languages and Computing*, 15(3-4):265–289, 2004.

[KG06] Jörg Kienzle and Samuel Gélineau. AO Challenge - Implementing the ACID Properties for Transactional Objects. In *Proc. of the 5th International Conference on Aspect-Oriented Software Development AOSD 2006, Bonn, Germany*, pages 202–213, March 2006.

[KHJ06] Jacques Klein, Loïc Hélouët, and Jean-Marc Jézéquel. Semantic-Based Weaving of Scenarios. In *Proc. of the 5th International Conference on Aspect-Oriented Software Development AOSD 2006, Bonn, Germany*, pages 27–38, March 2006.

[KK02a] Nora Koch and Andreas Kraus. The expressive Power of UML-based Web Engineering. In *Proc of the 2nd International Workshop on Web-oriented Software Technology (IWWOST 2002), in conjunction with ECOOP 2002, Málaga, Spain*, pages 21–32, June 2002.

[KK02b] Andreas Kraus and Nora Koch. Generation of Web Applications from UML. Models using an XML Publishing Framework. In *Proc. of the 5th World Conference on Integrated Design and Process Technology (IDPT 2002), Pasadena, CA, USA*, June 2002.

[KK03] Nora Koch and Andreas Kraus. Towards a Common Metamodell for the Development of Web Appliactions. In *Proc. of the 3rd International Conference on Web Engineering (ICWE 2003), Oviedo, Spain*, LNCS 2722, pages 497–506, July 2003.

[KK06] Mika Katara and Shmuel Katz. A Concern Architecture View for Aspect-Oriented Software Design. *Software and System Modeling*, 6(3):247–265, 2006.

[KKCM04] Nora Koch, Andreas Kraus, Cristina Cachero, and Santiago Meliá. Integration of Business Processes in Web Application Models. *Journal of Web Engineering*, 3(1):22–49, 2004.

[KKR04] Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. Integrating XML and Relational Database Systems. *World Wide Web*, 7(4):343–384, 2004.

[KL06] Sergei Kojarski and David H. Lorenz. Modeling Aspect Mechanisms: A Top-Down Approach. In *Proc. of the 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China*, pages 212–221, May 2006.

[KLM$^+$97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *Proc. of the 11th Europeen Conference on Object-Oriented Programming, Jyväskylä, Finland*, pages 220–242, June 1997.

[Kob01] Alfred Kobsa. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction*, 11(1-2):49–63, 2001.

[Koc99] Nora Koch. A Comparative Study of Methods for Hypermedia Development. Technical Report 9905, Ludwig-Maximilians-University Munich, Germany, 1999.

[Koc01] Nora Koch. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. PhD thesis, Ludwig-Maximilians-University Munich, Germany, October 2001.

[Koc07] Nora Koch. Classification of Model Transformation Techniques used in UML-based Web Engineering. *IET Software*, 1(3):98–111, 2007.

[KPR$^+$01] Gerti Kappel, Birgit Pröll, Werner Retschitzegger, Wieland Schwinger, and Thomas Hofer. Modeling Ubiquitous Web Applications - A Comparison of Approaches. In *Proc. of the 3rd International Conference on Information Integration and Web-based Applications & Services (IIWAS 2001), Linz, Austria*, September 2001.

[KPRR06] Gerti Kappel, Birgit Pröll, Siegfried Reich, and Werner Retschitzegger. An Introduction to Web Engineering. In Gerti Kappel, Birgit Pröll, Siegfried Reich, and Werner Retschitzegger, editors, *Web Engineering - Systematic Development of Web Applications*, pages 1–21. Wiley, June 2006.

[KPRS03] Gerti Kappel, Birgit Pröll, Werner Retschitzegger, and Wieland Schwinger. Customisation for Ubiquitous Web Applications a Comparison of Approaches. *International Journal of Web Engineering and Technology*, 1(1):79–111, 2003.

[Kru00] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, 2000.

[KRV05] Nora Koch, Gustavo Rossi, and Antonio Vallecillo, editors. *Proc. of the 1st Workshop on Model-Driven Web Engineering (MDWE 2005), in conjunction with ICWE 2005, Sydney, Australia*, July 2005.

[KW02] Nora Koch and Martin Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. In *Proc. of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Malaga*, LNCS 2347, pages 213–222, May 2002.

[KYX03] Jörg Kienzle, Yang Yu, and Jie Xiong. On Composition and Reuse of Aspects. In *Proc. of the Workshop on Foundations of Aspect-Oriented Languages (FOAL 2003), in conjunction with AOSD 2003, Boston, Massachusetts*, pages 17–24, March 2003.

[KZC06] Nora Koch, Gefei Zhang, and María José Escalona Cuaresma. Model transformations from requirements to web system design. In *Proc. of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, CA, USA*, pages 281–288. ACM, July 2006.

[LC00] Dongwon Lee and Wesley W. Chu. Comparative Analysis of Six XML Schema Languages. *ACM SIGMOD Record*, 29(3):76–87, 2000.

[Lie96] Karl J. Lieberherr. *Adaptive Object-Oriented Software: the Demeter Method with Propagation Patterns*. PWS Publishing Company, Boston, 1996.

[LM05] Ralf Lämmel and Erik Meijer. Mappings make data processing go 'round. In *Pre-Proc. the International Summer School on Generative and Transformation Techniques in Software Engineering (GTTSE 2005), Braga, Portugal*, July 2005.

[MBAE04] Mark Mahoney, Atef Bader, Omar Aldawud, and Tzilla Elrad. Using Aspects to Abstract and Modularize Statecharts. In *Proc. of the 5th Aspect-Oriented Modeling Workshop (AOM), in conjunction with (UML 2004), Lisbon, Portugal*, October 2004.

[MBC$^+$05] Ioana Manolescu, Marco Brambilla, Stefano Ceri, Sara Comai, and Piero Fraternali. Model-Driven Design and Deployment of Service-Enabled Web Applications. *ACM Transactions on Internet Technology*, 5(3):439–479, 2005.

[MFJ05] Pierre-Alain Muller, Franck Fleurey, and Jean-Marc Jézéquel. Weaving Executability into Object-Oriented Meta-languages. In *Proc. of the 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005), Montego Bay, Jamaica*, LNCS 3713, pages 264–278, October 2005.

[MFV06] Nathalie Moreno, Piero Fraternalli, and Antonio Vallecillo. A uml 2.0 profile for webml modeling. In *Proc. of the 2nd International Workshop on Model-Driven Web Engineering (MDWE 2006), in conjunction with ICWE 2006, Palo Alto, California, USA*, page 4. ACM Press, July 2006.

[MFV07] Nathalie Moreno, Piero Fraternalli, and Antonio Vallecillo. WebML modelling in UML. *IET Software*, 1(3):67–80, 2007.

[MG06] Santiago Meliá and Jaime Gómez. The WebSA Approach: Applying Model Driven Engineering to Web Applications. *Journal of Web Engineering*, 5(2):121–149, 2006.

[MK03] Hidehiko Masuhara and Gregor Kiczales. Modeling Crosscutting in Aspect-Oriented Mechanisms. In *Proc. of the 17th European Conference on Object-Oriented Programming (ECOOP'03), Darmstadt, Germany*, July 2003.

[MSFB05] Pierre-Alain Muller, Philippe Studer, Frédéric Fondement, and Jean Bézivin. Platform independent Web application modeling and development with Netsilon. *Software and System Modeling*, 4(4):424–442, 2005.

[MSZJ04] Haohai Ma, Weizhong Shao, Lu Zhang, and Yanbing Jiang. Applying OO Metrics to Assess UML Meta-models. In *Proc. of the 7th International Conference on the Unified Modelling Language: Modelling Languages and Applications (UML 2004), Lisbon, Portugal*, LNCS 3273, pages 12–26, October 2004.

[OMG01] Object Management Group OMG. UML Specification Version 1.4. http://www.omg.org/docs/formal/01-09-67.pdf, September 2001.

[OMG02] Object Management Group OMG. Meta Object Facility (MOF) Specification 1.4. http://www.omg.org/docs/formal/02-04-03.pdf, April 2002.

[OMG03] Object Management Group OMG. MDA Guide Version 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf, June 2003.

[OMG04] 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, October 2004.

[OMG05a] Object Management Group OMG. Meta Object Facility (MOF) 2.0 Query/View/-Transformation Specification. Final Adopted Specification. ptc/05-11-01, November 2005.

[OMG05b] Object Management Group OMG. MOF 2.0/XMI Mapping Specification, v2.1. http://www.omg.org/docs/formal/05-09-01.pdf, September 2005.

[OMG05c] Object Management Group OMG. OCL Specification Version 2.0. http://www.omg.org/docs/ptc/05-06-06.pdf, June 2005.

[OMG05d] Object Management Group OMG. UML Specification: Superstructure Version 2.0. http://www.omg.org/docs/formal/05-07-04.pdf, August 2005.

[PAF00] Oscar Pastor, Silvia Abrahão, and Joan Fons. OOWS: An Object-Oriented Approach for Web-Solutions Modeling. In *Proc. of the International Conference on Information Society (ICIS 2000), Ljubljana, Slovenia*, October 2000.

[Par72] David L. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12):1053–1058, December 1972.

[Pat00] Fabio Paterno. Model-Based Design of Interactive Applications. *ACM Intelligence*, 11(4):26–38, 2000.

[PCT05] Peter Plessers, Sven Casteleyn, and Olga De Troyer. Semantic Web Development with WSDM. In *Proc. of the 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot2005), in conjunction with ISWC 2005, November 6-10, Galway, Irland*. CEUR Workshop Proceedings, 2005.

[PD99] Norman W. Paton and Oscar Díaz. Active database systems. *ACM Computing Surveys*, 31(1):63–103, 1999.

[PDF$^+$02] Renaud Pawlak, Laurence Duchien, Gerard Florin, Fabrice Legond-Aubry, Lionel Seinturier, and Laurent Martelli. A UML Notation for Aspect-Oriented Software Design. In *Proc. of the 1st Workshop on Aspect-Oriented Modeling with UML (AOSD'02), Enschede, The Netherlands*, March 2002.

[PFPA06] Oscar Pastor, Joan Fons, Vicente Pelechano, and Silvia Abrahão. Conceptual Modelling of Web Applications: The OOWS Approach. In Emilia Mendes and Nile Mosley, editors, *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*, pages 277–302. Springer, 2006.

[PIP$^+$97] Oscar Pastor, Emilio Insfrán, Vicente Pelechano, José Romero, and José Merseguer. OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods. In *Proc. of the 9th International Conference on Advanced Information Systems Engineering (CAISE 1997), Barcelona, Catalonia, Spain*, LNCS 1250, pages 145–158, June 1997.

[Pre07] Gerhard Preisinger. Towards model-driven web application development with aspectwebml - an integrated graphical development environment. Master's thesis, Vienna University of Technology, To be finished, 2007.

[PSD$^+$05] Renaud Pawlak, Lionel Seinturier, Laurence Duchien, Laurent Martelli, Fabrice Legond-Aubry, and Gérard Florin. Aspect-Oriented Software Development with Java Aspect Components. In R.E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 343–369. Addison-Wesley, Boston, 2005.

[PTSC05] Juan Carlos Preciado, Marino Linaje Trigueros, F. Sanchez, and Sara Comai. Necessity of methodologies to model Rich Internet Applications. In *Proc. of the 7th IEEE International Workshop on Web Site Evolution, Budapest, Hungary*, pages 7–13, September 2005.

[PZ03] Eduardo Kessler Piveta and Luiz Carlos Zancanella. Observer Pattern using Aspect-Oriented Programming. In *Proc. of the 3rd Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP 2003), Porto de Galinhas, PE, Brazil*, August 2003.

[RFG05] Raghu Reddy, Robert France, and Geri Georg. An Aspect Oriented Approach to Analyzing Dependability Features. In *6th International Workshop on Aspect-Oriented Modeling (AOM), in conjunction with AOSD 2005, Chicago, Illinois*, March 2005.

[RGR$^+$06] Raghu Reddy, Sudipto Ghosh, Robert B. Rance, Greg Straw, James M. Bieman, Eunjee Song, and Geri Georg. Directives for Composing Aspect-Oriented Design Class Models. In *Transactions on Aspect-Oriented Software Development I*, LNCS 3880, pages 75 – 105. Springer-Verlag, 2006.

[RJB05] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Guide*. Addison-Wesley, Boston, 2nd edition, 2005.

[RS06] Gustavo Rossi and Daniel Schwabe. Model-Based Web Application Development. In Emilia Mendes and Nile Mosley, editors, *Web Engineering: Theory and Practice of Metrics and Measurement for Web Development*, pages 203–333. Springer, 2006.

[RSdLC95] Gustavo Rossi, Daniel Schwabe, Carlos José Pereira de Lucena, and Donald D. Cowan. An Object-Oriented Model for Designing the Human-Computer Interface Of Hypermedia Applications. In *Proc. of the International Workshop on Hypermedia Design (IWHD 1995), Montpellier, France*, pages 123–143, June 1995.

[RSFG06] Raghu Reddy, Arnor Solberg, Robert France, and Sudipto Ghosh. Composing Sequence Models using Tags. In *9th International Workshop on Aspect-Oriented Modeling (AOM) in conjunctino with MoDELS 2006, Genova, Italy*, October 2006.

[RSG01] Gustavo Rossi, Daniel Schwabe, and Robson Guimarães. Designing Personalized Web Applications. In *Proc. of the 10th International World Wide Web Conference (WWW 2001), Hong Kong, China*, pages 275–284, May 2001.

[RTT04] Antonia M. Reina, Jesus Torres, and Miguel Toro. Separating Concerns by Means of UML-profiles and Metamodels in PIMs. In *5th Aspect-Oriented Modeling Workshop (AOM), in conjunction with UML 2004, Lisbon, Portugal*, October 2004.

[RVP06] Gonzalo Rojas, Pedro Valderas, and Vicente Pelechano. Describing Adaptive Navigation Requirements of Web Applications. In *Proc. of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2006), Dublin, Ireland*, LNCS 4018, pages 318–322, June 2006.

[Sch01] Wieland Schwinger. *Modelling Ubiquitous Web Applications - Requirements and Concepts*. PhD thesis, Johannes Kepler University Linz, November 2001.

[Sch06a] Andrea Schauerhuber. aspectUWA: Applying Aspect-Orientation to the Model-Driven Development of Ubiquitous Web Applications. Student Extravaganza: Poster Event, 5th International Conference on Aspect-Oriented Software Development (AOSD 2006), Bonn, Germany, March 2006.

[Sch06b] Douglas C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.

[SdAPM99] Daniel Schwabe, Rita de Almeida Pontes, and Isbela Moura. OOHDM-Web: An Environment for Implementation of. Hypermedia Applications in the WWW. *ACM SIGWEB Newsletter*, 8(2):18–34, 1999.

[Sec02] ITU Telecommunication Standardization Sector. ITU-T Recommendation Z.100: Specification and Description Language (SDL), Geneva, Switzerland. http://www.itu.int/rec/T-REC-Z/en, August 2002.

[Sec04] ITU Telecommunication Standardization Sector. ITU-T Recommendation Z.120: Message Sequence Chart (MSC), Geneva, Switzerland. http://www.itu.int/rec/T-REC-Z/en, April 2004.

[SGR02] Daniel Schwabe, Robson Guimarães, and Gustavo Rossi. Cohesive Design of Personalized Web Applications. *IEEE Internet Computing*, 6(2):34–43, 2002.

[SHU02a] Dominik Stein, Stefan Hanenberg, and Rainer Unland. An UML-based Aspect-Oriented Design Notation. In *Proc. of the 1st International Conference on Aspect-Oriented Software Development (AOSD 2002), Enschede, The Netherlands*, pages 106–112, April 2002.

[SHU02b] Dominik Stein, Stefan Hanenberg, and Rainer Unland. Designing Aspect-Oriented Crosscutting in UML. In *1st Workshop on Aspect-Oriented Modeling with UML, in conjunction with AOSD 2002, Enschede, The Netherlands*, March 2002.

[SHU02c] Dominik Stein, Stefan Hanenberg, and Rainer Unland. On Representing Join Points in the UML. In *2nd International Workshop on Aspect-Oriented Modeling with UML, in conjunction with UML 2002*, September 2002.

[SHU06] Dominik Stein, Stefan Hanenberg, and Rainer Unland. Expressing Different Conceptual Models of Join Point Selections in Aspect-Oriented Design. In *Proc. of the 5th International Conference on Aspect-Oriented Software Development (AOSD 2006), Bonn, Germany*, pages 15–26, March 2006.

[SK06] Wieland Schwinger and Nora Koch. Modeling Web Applications. In Gerti Kappel, Birgit Pröll, Siegfried Reich, and Werner Retschitzegger, editors, *Web Engineering - Systematic Development of Web Applications*, pages 39–64. Wiley, June 2006.

[SKK04] Dominik Stein, Jörg Kienzle, and Mohamed Kandé. 5th International Workshop on Aspect-Oriented Modeling. In *UML Modeling Languages and Applications: 2004 Satellite Activities, Lisbon, Portugal*, pages 13–22. Springer-Verlag, October 2004.

[Sof00] Rational Software. Migrating from XML DTD to XMLSchema using UML. Rational Software White Paper, August 2000.

[SR94] Daniel Schwabe and Gustavo Rossi. From Domain Models to Hypermedia Applications: An Object-Oriented Approach. In *International Workshop on Methodologies for Designing and Developing Hypermedia Applications, Edinburgh*, September 1994.

[SR98] Daniel Schwabe and Gustavo Rossi. An Object Oriented Approach to Web-Based Applications Design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.

[SR05] Stanley M. Sutton, Jr. and Isabelle Rouvellou. Concern Modeling for Aspect-Oriented Software Development. In R.E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 479–505. Addison-Wesley, Boston, 2005.

[SRB96] Daniel Schwabe, Gustavo Rossi, and Simone Diniz Junqueira Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proc. of the 7th Conference on Hypertext (HT 1996), Washington DC*, pages 116–128, March 1996.

[SS02] Daniel Schwabe and César Simões Salim. Integrating Knowledge Management Applications in the Enterprise The Xerox Knowledge Portal Project. *Knowledge and Process Management*, 9(3):190–201, 2002.

[SSK$^+$06] Andrea Schauerhuber, Wieland Schwinger, Elisabeth Kapsammer, Werner Retschitzegger, and Manuel Wimmer. Towards a Common Reference Architecture for Aspect-Oriented Modeling. In *Proc. of the 8th International Workshop on Aspect-Oriented Modeling (AOM) at AOSD'06, Bonn, Germany*, March 2006.

[SSK$^+$07] Andrea Schauerhuber, Wieland Schwinger, Elisabeth Kapsammer, Werner Retschitzegger, Manuel Wimmer, and Gerti Kappel. A Survey on Aspect-Oriented Modeling Approaches. Technical report, Vienna University of Technology, October 2007.

[SSR$^+$05] Arnor Solberg, Devon Simmonds, Raghu Reddy, Sudipto Ghosh, and Robert B. France. Using Aspect Oriented Techniques to Support Separation of Concerns in Model Driven Development. In *Proc. of the 29th International Computer Software and Applications Conference (COMPSAC 2005), Edinburgh, Scotland, UK*, pages 121–126, July 2005.

[SSW$^+$07] Andrea Schauerhuber, Wieland Schwinger, Manuel Wimmer, Werner Retschitzegger, and Gerti Kappel. A Survey on Web Modeling Approaches for Ubiquitous Web Applications. Technical report, Vienna University of Technology, October 2007.

[STW$^+$06] Frans Sanen, Eddy Truyen, Bart De Win, Wouter Joosen, Neil Loughran, Geoff Coulson, Awais Rashid, Andronikos Nedos, Andrew Jackson, and Siobhán Clarke. Study on interaction issues. Technical Report D44 AOSD-Europe-KUL-7, AOSD-Europe, February 2006.

[SVWJ05] Davy Suvée, Wim Vanderperren, Dennis Wagelaar, and Viviane Jonckers. There are no Aspects. *Electr. Notes Theor. Comput. Sci.*, 114:153–174, 2005.

[SWK06] Andrea Schauerhuber, Manuel Wimmer, and Elisabeth Kapsammer. Bridging Existing Web Modeling Languages to Model-Driven Engineering: A Metamodel for WebML. In *Proc. of the 2nd International Workshop on Model-Driven Web Engineering (MDWE 2006), in conjunction with ICWE 2006, Palo Alto, California, USA*, page 5. ACM Press, July 2006.

[SWK$^+$07] Andrea Schauerhuber, Manuel Wimmer, Elisabeth Kapsammer, Wieland Schwinger, and Werner Retschitzegger. Briding WebML to model-driven engineering: from document type definitions to meta object facility. *IET Software*, 1(3):81–97, 2007.

[SWS$^+$07] Andrea Schauerhuber, Manuel Wimmer, Wieland Schwinger, Elisabeth Kapsammer, and Werner Retschitzegger. Aspect-Oriented Modeling of Ubiquitous Web Applications: The aspectWebML Approach. In *Proc. of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), Tucson, Arizona, USA*, pages 569–576, March 2007.

[TC04] Olga De Troyer and Sven Casteleyn. Designing Localized Web Sites. In *Proc. of the 5th International Conference on Web Information Systems Engineering (WISE 2004), Brisbane, Australia*, LNCS 3306, pages 547–558, November 2004.

[Tea05] The AspectJ Team. The AspectJ (TM) Programming Guide. http://www.eclipse.org/aspectj/, October 2005.

[TFPP04] Victoria Torres, Joan Fons, Vicente Pelechano, and Oscar Pastor. Navigational modeling and the semantic web. an ontology based approach. In *Proceedings of the Joint Conference 10th Brazilian Symposium on Multimedia and the Web & 2nd Latin American Web Congress, (WebMedia & LA-Web 2004), Ribeirao Preto-SP, Brazil*, pages 94–96. IEEE Computer Society, October 2004.

[TL98] Olga De Troyer and C. J. Leune. WSDM: A User Centered Design Method for Web Sites. *Computer Networks*, 30(1-7):85–94, 1998.

[TOHS99] Peri L. Tarr, Harold L. Ossher, William H. Harrison, and Stanley M. Sutton, Jr. *N* Degrees of Separation: Multi-Dimensional Separation of Concerns. In *Proc. of the 21st International Conference on Software Engineering (ICSE 1999), Los Angeles, California*, pages 107–119, May 1999.

[Tom07] Cornelia Tomasek. Integration von crosscutting concerns in aspectwebml. Master's thesis, Vienna University of Technology, To be finished, 2007.

[TPRV05] Victoria Torres, Vicente Pelechano, Marta Ruiz, and Pedro Valderas. A Model Driven Approach for the Integration of External Functionality in Web Applications. The Travel Agency System. In *Proc. of 1st Workshop on Model-Driven Web Engineering, in conjunction with ICWE 2005), Sydney, Australia*, pages 1–11, July 2005.

[vdBCC05] Klaas van den Berg, José M. Conejero, and Ruzanna Chitchyan. AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation. Technical Report D9 AOSD-Europe-UT-01, AOSD-Europe, May 2005.

[vdSHBC06] Kees van der Sluijs, Geert-Jan Houben, Jeen Broekstra, and Sven Casteleyn. Hera-S - Web Design Using Sesame. In *Proc. of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, CA, USA*, pages 337–344, July 2006.

[vFGC04] Christina von Flach Garcia Chavez. *A Model-Driven Approach for Aspect-Oriented Design*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, April 2004.

[vFGCdL03] Christina von Flach Garcia Chavez and Carlos J. P. de Lucena. A Theory of Aspects for Aspect-Oriented Software Development. In *7th Brazilian Symposium on Software Engineering (SBES 2003)*, October 2003.

[VFP05] Pedro Valderas, Joan Fons, and Vicente Pelechano. Using Task Descriptions for the Specification of Web Application Requirements. In *Anais do WER05 - Workshop em Engenharia de Requisitos, Porto, Portuga*, pages 257–268, June 2005.

[VH02] Richard Vdovjak and Geert-Jan Houben. Providing the Semantic Layer for WIS Design. In *Proc. of the 14th International Conference on Advanced Information Systems Engineering (CAISE 2002), Toronto, Canada*, LNCS 2348, pages 584–599, May 2002.

[VIG05] Fabio Vitali, Angelo Di Iorio, and Daniele Gubellini. Design patterns for descriptive document substructures. In *Proceedings of the Extreme Markup Languages Conference, Montréal, Quebec, Canada*, August 2005.

[VSdS00] Patricia Vilain, Daniel Schwabe, and Clarisse Sieckenius de Souza. A Diagrammatic Tool for Representing User Interaction in UML. In *Proc. of the 3rd International Conference on the Unified Modeling Language (UML 2000), York, UK*, LNCS 1939, pages 133–147, October 2000.

[VVFP07] Francisco Valverde, Pedro Valderas, Joan Fons, and Oscar Pastor. A MDA-based Environment for Web Applications Development: From Conceptual Models to Code. In *International Workshop on Web-oriented Software Technology (IWWOST 2007), in conjunction with ICWE 2007, Como, Italy*, July 2007.

[W3C04a] World Wide Web Consortium W3C. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. http://www.w3.org/TR/CCPP-struct-vocab/, January 2004.

[W3C04b] World Wide Web Consortium W3C. OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/, February 2004.

[W3C04c] World Wide Web Consortium W3C. RDF Primer. (XML) 1.1 (Second Edition). http://www.w3.org/TR/rdf-primer/, February 2004.

[W3C04d] World Wide Web Consortium W3C. XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/XML Schema-0/, October 2004.

[W3C06] World Wide Web Consortium W3C. Extensible Markup Language (XML) 1.1 (Second Edition). http://www.w3c/TR/xml11/, September 2006.

[WCWH03] Arouna Woukeu, Leslie Carr, Gary Wills, and Wendy Hall. Rethinking Web Design Models: Requirements for Addressing the Content. Technical Report ECSTR-IAM03-002, University of Southampton, 2003.

[WS00] Retschitzegger Werner and Wieland Schwinger. Towards Modeling of Data Web Applications A Requirements' Perspective. In *Proc. of Americas Conference on Information Systems (AMCIS 2000), Long Beach, USA*, August 2000.

[WS01] Roy Want and Bill N. Schilit. Guest Editors' Introduction: Expanding the Horizons of Location-Aware Computing. *IEEE Computer*, 34(8):31–34, 2001.

[WSKK06] Manuel Wimmer, Andrea Schauerhuber, Elisabeth Kapsammer, and Gerhard Kramler. From Document Type Definitions to Metamodels: The WebML Case Study. Technical report, Vienna University of Technology, March 2006.

[ZBKK05] Gefei Zhang, Hubert Baumeister, Nora Koch, and Alexander Knapp. Aspect-Oriented Modeling of Access Control in Web Applications. In *6th International Workshop on Aspect-Oriented Modeling (AOM), in conjunction with AOSD 2005, Chicago, Illinois*, March 2005.

[ZCVG06] Jing Zhang, Thomas Cottenier, Aswin Van Den Berg, and Jeff Gray. Aspect Interference and Composition in the Motorola Aspect-Oriented Modeling Weavr. In *9th International Workshop on Aspect-Oriented Modeling (AOM), in conjunction with MODELS 2006, Genova, Italy*, October 2006.