3   Modeling Web Applications

Wieland Schwinger, Nora Koch

It is not (yet) common to model Web applications in practice. This is unfortunate as a model-based approach provides a better alternative to the *ad-hoc* development of Web applications and its inherent problems. As mentioned in previous chapters, these are for example insufficient fulfillment of requirements, faulty specification, or missing system documentation. Models represent a solid starting point for the implementation of a Web application taking into account static and dynamic aspects of the content, hypertext, and presentation levels of a Web application. While the content model of a Web application which aims at capturing underlying information and application logic is similar to the corresponding model of a non-Web application, the need to consider the hypertext is particular to Web applications. The hypertext model represents all kinds of navigation possibilities based on the content. The presentation model maps hypertext structures to pages and their links thus represent the graphical user interface. The inclusion of context information, such as user, time, location, and device used, and the adaptation of the Web application which is ''derived'' from this information, has gained increasing attention in modeling efforts. This is undoubtedly a consequence of ubiquitous Web applications that have become increasingly popular. This chapter discusses the spectrum of existing methods and some tools available to model Web applications and their highlights to help the reader select a suitable modeling method. Such methods are the basis for model-based development and code-generation tools, which allow us to consider the use of different Web clients and run-time platforms.

3.1   Introduction

To build a dog house you simply need two skillful hands, the required materials, and a few tools to quickly start hammering and sawing and achieve an attractive result, depending on your personal creativity. Nobody, however (Booch et al. 1999), would set about building a skyscraper with the same naïve light-heartedness – the result would surely be fatal! What's clear to everybody when it comes to building a skyscraper is often ignored when it comes to building complex Web applications. A systematic approach and a specification of the Web application to be built in the form of visual models are recommended if we need to develop complex Web applications.

This chapter deals with the model-based development of Web applications. Section 3.2 provides an insight into general modeling basics, followed by section 3.3 which discusses the

specifics in modeling Web applications. The subsequent sections describe different models for Web applications, starting from a requirements description. We will use an example of an online conference paper reviewing system throughout these sections. Section 3.9 gives an overview of existing methods and some tools to model Web applications. Finally, the last section gives an overview of future development trends in the field of Web application modeling.

3.2   Fundamentals

Engineering disciplines have successfully used models to reduce complexity, document design decisions, and facilitate communication within project teams. Modeling is aimed at providing a specification of a system to be built in a degree of detail sufficient for that system's implementation. The result of a modeling process are models representing the relevant aspects of the system in a simplified and – ideally – comprehensible manner.
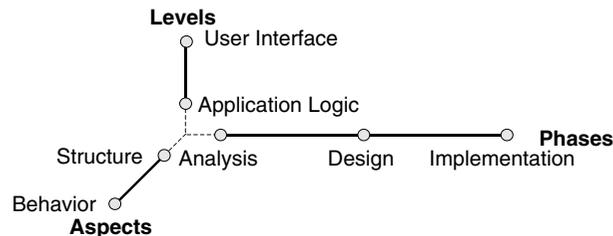
Figure 3-1   Requirements of software application modeling.

Computer science has also been using the modeling approach to develop software for some time. In this field, the object of modeling is the application to be created. Figure 3-1 shows that the scope of modeling spans along three orthogonal dimensions. The first dimension traditionally comprises the application logic level and the user interface level in the sense of an encapsulation of the ''what'' and ''how'' of an application. Aspects known as structure (i.e., objects, their attributes, and their relationships to other objects) and behavior (i.e., functions and processes), both of the application logic and the user interface, form another dimension. Since an application cannot be developed ''in one shot'', but has to be gradually refined and expanded during the development process, the development phases form the third application modeling dimension. Through successive refinements the requirements identified in the requirements analysis are transformed to analysis models first and design models later, on which the implementation will be based.

The roots of modeling are found on the one hand in Data Engineering and, on the other hand, in Software Engineering. Historically, Data Engineering modeling focuses on the structural aspects, i.e., the data aspects of an application. Identification of entities, their grouping and their relationships is the major focus. The best-known model in this respect is the *Entity-Relationship* (*ER*) model (Chen 1976). In contrast, modeling in Software Engineering focuses on behavioral aspects, to fulfill the needs of programming languages. Today, it is mainly based on an object-oriented approach. The most important characteristics of object-oriented modeling are a holistic approach to system modeling and the central concept of the object, comprising structure and behavior.

The Unified Modeling Language (UML) OMG 2004, Hitz et al. 2005 is an object-oriented modeling language and seen as a kind of *lingua franca* in object-oriented software development; it forms the basis of most modeling methods for Web applications. UML allows to specify the aspects of a software system in the form of models, and uses various diagrams to represent them graphically. UML has two types of diagrams: structural diagrams such as class diagrams, component diagrams, composite structure diagrams, and deployment diagrams, as well as behavioral diagrams, such as use case diagrams, state machine diagrams, and activity diagrams.

3.3   Modeling Specifics in Web Engineering

The tools of the trade in Web application modeling are basically not new, however, methods to model traditional applications are not expressive enough for specific characteristics of Web applications (see also section 1.3). For example, traditional modeling languages (such as UML) do not provide appropriate concepts for the specification of hyperlinks. This was the reason why special modeling approaches for Web applications have been developed during the past few years, which allow to address a Web application in the three dimensions introduced above, i.e., levels, aspects, and phases.

3.3.1   Levels

To model Web applications, the document-like character of its content as well as its non-linear hypertext navigation has to be taken into account. This is the reason why we distinguish three levels when modeling Web applications, as shown in Figure 3-2, in contrast to the two levels used in the modeling methods for traditional applications. The three levels are *content*, i.e., the information and application logics underneath the Web application, *hypertext*, i.e., the structuring of the content into nodes and links between these nodes, and the *presentation*, i.e., the user interface or page layout. Most methods which are used to model Web applications follow this separation into three levels (Fraternali 1999).
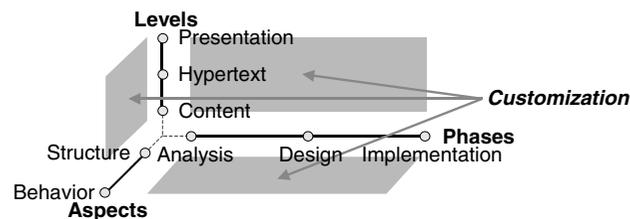
Figure 3-2   Requirements of Web application modeling.

A clear separation of these three levels allows reuse and helps to reduce complexity. For example, we could specify a number of different hypertext structures that will do justice to the specific requirements of different user groups and used devices for a given content. The aim of a content model is the explicit definition of the information structure. Comparable to a database schema in data modeling this eliminates redundancies. This means that the structure of the information will remain unchanged, even if the information itself changes frequently.

To design efficient navigation, content may be offered redundantly on several nodes on the hypertext level. Due to the separation of concerns, content is just modeled once in the content model and the hypertext structure model just references the corresponding content. In this way, users can find this information over several access paths. To prevent users from getting lost while navigating and to keep the cognitive stress on users as low as possible, hypertext modeling should rely on recurring navigation patterns (Bernstein 1998).

In turn, when modeling the presentation level, the focus is on a uniform presentation structure for the pages to achieve a brand recognition effect for the Web application among its users. Although the visual appearance of a Web application is of importance, aesthetic aspects are not within the major focus of modeling.

Despite a separation of concerns and the different objectives at the three levels, we would like to map the levels to one another. To achieve this mapping between levels, level inter-dependencies have to be captured explicitly. For example, different personalized hypertext access paths could be mapped onto one single content model. A comprehensive model of a Web application includes all three levels discussed here, however, the emphasis can vary depending on the type of Web application. Web applications that provide a purely hypertext-oriented user interface to a large data set will probably require the modeling focus to be on content and hypertext structure. In contrast, presentation-oriented Web applications, e.g., corporate portals or online shopping malls, will most likely have larger demands on presentation modeling.

3.3.2 Aspects

Following the object-oriented principles, structure and behavior are modeled at each of the three levels, i.e. at content, hypertext and presentation. The relevance of the structure and behavior models depends on the type of Web application to be implemented. Web applications which make mainly static information available require less behavior modeling compared with highly interactive Web applications, such as for example e-commerce applications which provide search engines, purchase order functions, etc. With respect to mapping the different levels, it is recommended to use a uniform modeling formalism for structure and behavior, which might allow relying on one single CASE tool. Naturally, this modeling formalism has to cope with the specific characteristics of each of the three levels.

3.3.3 Phases

There is no consensus in literature about a general modeling approach for the development of Web applications (see also Chapter 10). In any case, the sequence of steps to model the levels should be decided by the modeler. Depending on the type of Web application, it should be possible to pursue an information-driven approach, i.e., starting with content modeling, or a presentation-driven approach, i.e., starting with modeling of the application's presentation aspects. Model-based development in Web engineering contradicts somewhat the often found practices in Web projects comprising, e.g., short-lived development cycles and the desire for ''agile methods'' (see section 10.5). A model-based approach counters this situation with a comprehensive specification of a solution model and, if appropriate case tool support is available, the possibility to automatically generate the (prototypical) Web application. Models also ensure the sustainability of solution ideas, in contrast to shorter-lived software solutions. In addition, the communication amongst the developers of a team as well as between customers and developers is improved.

3.3.4   Customization

The inclusion of context information in the development of Web applications plays a significant role to allow for e.g. personalization, multi-delivery and location-based services. Customization considers the context, e.g., users' preferences, device characteristics, or bandwidth restrictions, and allows to adapt the Web application accordingly. It influences all three Web modeling dimensions of content, hypertext, and presentation with respect to structure and behavior and should be taken into account in all phases of the development process. Handling context information is, therefore, treated as an independent modeling dimension (see Figure 3-2, Kappel et al. 2003).

Since there is currently no wide-spread modeling method that covers all dimensions discussed here (see also section 3.9), we will use UML as our notation in this chapter and expand it by borrowing a few concepts from a UML-based Web application modeling method, namely *UWE* (*UML-based Web Engineering*) (Koch and Kraus 2002). We suggest using UWE as UWE is compliant with UML. It is defined as a UML profile that is a lightweight extension of UML (see also section 3.9 for a comparison of methods).

3.4   Modeling Requirements

As shown in Chapter 2 various techniques can be used to identify, analyze, describe, evaluate, and manage Web application requirements. Use cases are the preferred modeling technique for functional requirements, not least since they can be represented graphically. The overall functionality of a Web application is modeled as a set of use cases, which describe the Web application requirements from the actors' (people and other systems) perspectives. Additionally, use cases can be supplemented by UML activity diagrams to describe the functional requirements in more detail.

One peculiarity of Web application requirements is navigation functionality, which allows the user to navigate through the hypertext and to find nodes. (Baresi et al. 2001) suggests separating the functional from the navigational use cases, creating two distinct models. Another approach (UWE), selected herein, is to create one single use case model, which uses the UML ≪navigation≫ stereotype to denote the difference between functional and hypertext-specific use cases.

All Web applications have at least one human user, most often anonymous. In our example of an online conference paper reviewing system (referred to as ''the reviewing system'' in the following), four actors can be identified: users of the reviewing system, authors submitting papers to the conference, members of the program committee (reviewers) reviewing papers submitted, and the chair of the program committee (PC chair). Figure 3-3 shows a use case diagram representing part of the use case model for the reviewing system, which will serve as the starting point for further modeling. Navigational requirements supplementing functional requirements are made explicit through the stereotype ≪navigation≫ in the use case diagram.

Use cases should be described in detail. We can describe each use case in textual form or by use of a behavior diagram, e.g. an activity diagram. Activity diagrams are mainly used when use cases are based on more complex application logic. Such a use case, for example, might be implemented as a Web service. Figure 3-4 is an example of a simplified paper submission process.
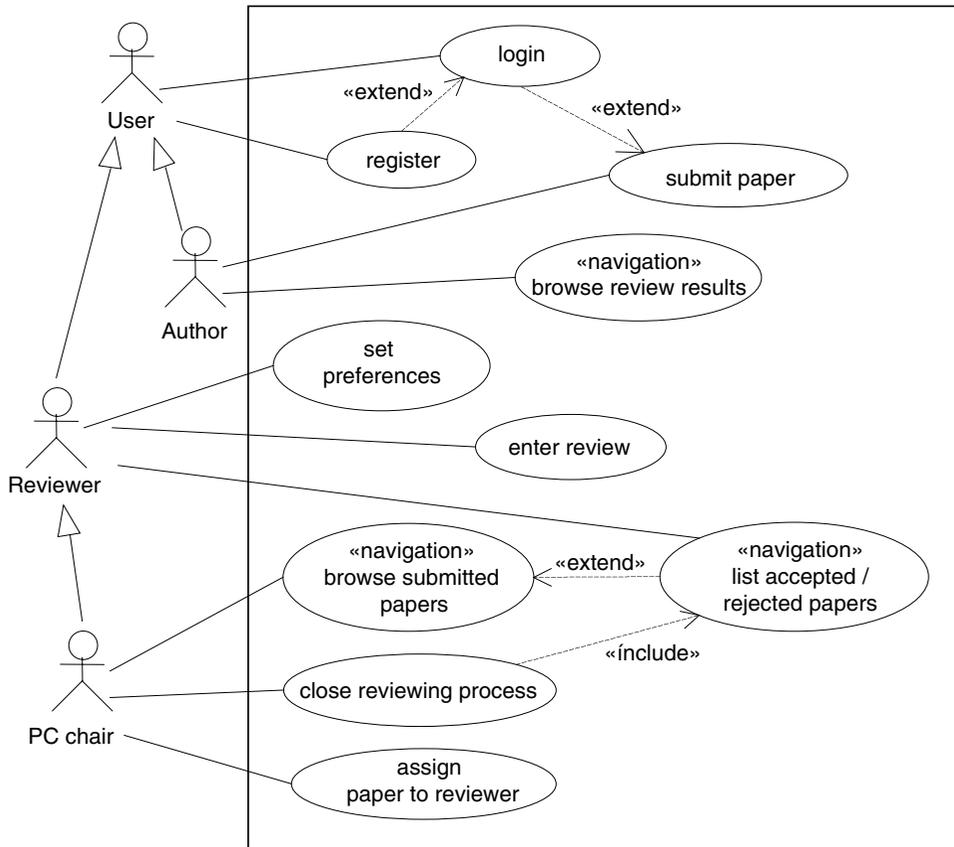
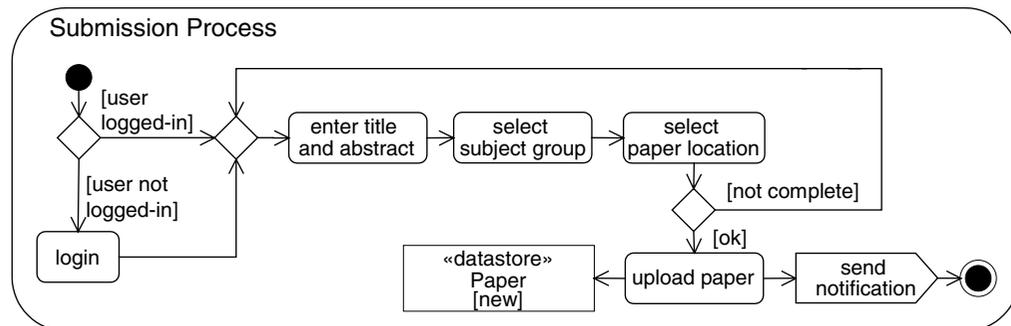**Figure 3-3**   Use case diagram of the reviewing system.

Figure 3-4   Activity diagram of the submission process.

3.5   Content Modeling

The information provided by a Web application is one of the most important factors for the success of that application, not least due to the origins of the Web as an information medium (see also Chapter 11). Modeling the content in the sense of pure data modeling is normally sufficient for static Web applications. Complex Web applications (according to the categorization defined in Chapter 1) additionally require the modeling of behavioral aspects. This means that content modeling includes the creation of the problem domain model, consisting of static and dynamic aspects, as known from traditional Software Engineering. In addition the following Web application characteristics have to be taken into account:

- *Document-centric character and multimedia*: It is necessary to take all kinds of different media formats into account when modeling the content, including the structures the information is based on.
- *Integration of existing data and software*: Many Web applications build on existing data repositories and software components, which were not created for Web applications originally. Content modeling has to satisfy two potentially contradicting objectives, i.e., it should cover the content requirements of the Web application to the best possible extent, and it should include existing data structures and software components.

3.5.1   Objectives

Content modeling is aimed at transferring the information and functional requirements determined by requirements engineering to a model. The hypertext character of a Web application and the requirements of its presentation will not be considered in this effort.

Content modeling produces a model that comprises both the structural aspects of the content, e.g., in the form of a class diagram, and, depending on the type of Web application, the behavioral aspects, e.g., in the form of state and interaction diagrams.

3.5.2   Concepts

As mentioned earlier, content modeling builds on the concepts and methods of data modeling or object-oriented modeling. It strives to ensure that existing information is free from redundancies and reusable.

Figure 3-5 shows a very simplified UML class diagram for the reviewing system example. The diagram models a conference to be held on a number of topics, and users who can sign in to the conference and submit their papers. A paper is subject to a review by three reviewers. Notice the invariant attached to the class ''Paper'': it ensures that authors won't be able to review their own papers. This class diagram will later serve as the basis to model the hypertext and the presentation for the example application.

In addition to the class diagram, Figure 3-6 shows a state machine diagram used to model the various states of a paper in the reviewing system. It shows that a submitted paper will be assigned to three reviewers for review after the submission deadline has expired. If a pre-set threshold value is reached, the paper is accepted; otherwise, it is rejected. In both cases the authors are
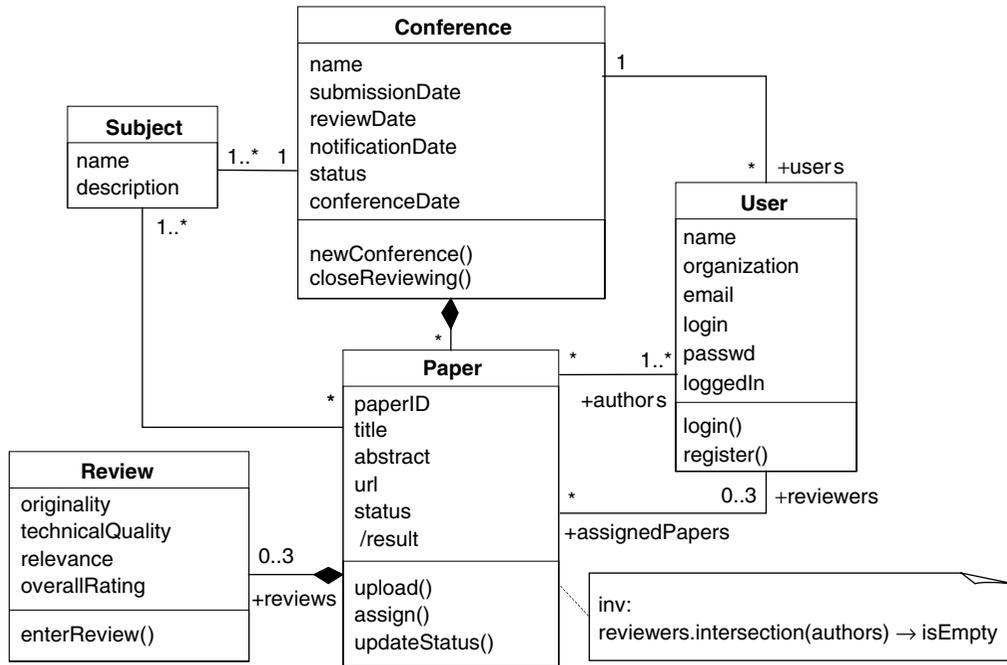
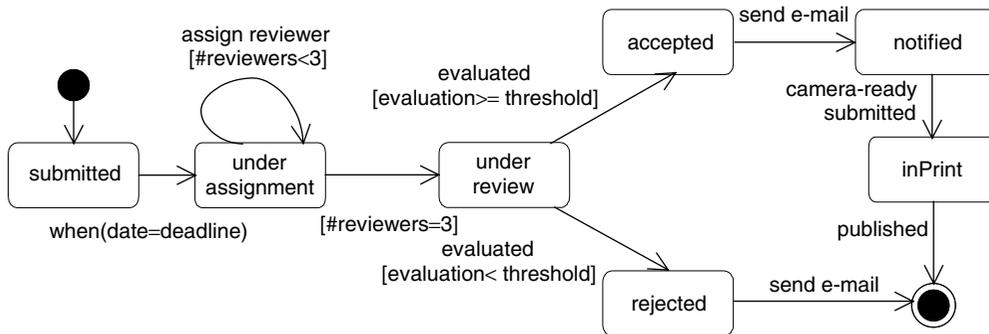**Figure 3-5**   Class diagram for the reviewing system.

Figure 3-6   State machine diagram for the states of a paper.

notified via e-mail about the outcome of the review. Finally, an accepted paper will be printed once the final version has been submitted.

3.6   Hypertext Modeling

The non-linearity of hypertext is one of the most important properties to be taken into account when modeling Web applications. Thus the hypertext structure has to be designed carefully. This can be achieved by using suitable access structures, i.e., navigation options, to avoid the risk of users getting lost and putting them under excessive cognitive stress (see also Chapter 11).

3.6.1   Objectives

The objective of hypertext modeling – also known as navigation modeling – is to specify the navigability through the content of a Web application, i.e., the navigation paths available to the users. Hypertext modeling generates a two-fold result: First, it produces the *hypertext structure model*, also known as *navigation structure model* which defines the structure of the hypertext, i.e., which classes of the content model can be visited by navigation. Second, it refines the hypertext structure model by access elements in the form of an *access model*.

Hypertext modeling focuses on the structural aspects of the hypertext and the access elements. The navigational behavior of a Web application is normally not represented explicitly, because it provides very little additional information for the developer.

3.6.2   Hypertext Structure Modeling Concepts

In contrast to the content level, for which ER diagrams or class diagrams are used, specialized notations are often employed to model the hypertext structure. Hypertext structure modeling is based on the concepts of hypertext, i.e., on nodes (also called pages or documents) and links between these nodes.

The starting point used for the creation of a hypertext structure model is usually the content model which contains the classes and objects to be made available as nodes in the hypertext. Often the hypertext structure model is specified as a view on the content model and is therefore sometimes also called the navigational view. Thereby a node is specified as a view on the content model selecting one or more objects from the content. Some methods even define transformation rules to derive links on the basis of relationships on the content level. Additional links can be added by explicit design decision. Other methods model the hypertext structure independently of the content model. For example the OOHDM (Object-Oriented Hypermedia Design Method) (Schwabe et al. 2002) offers an approach to model scenarios, where the hypertext structure model can be built directly from the navigational requirements identified by these scenarios.

In any case, we can create various hypertext structure models that define hypertext views on the content. For example, if we take into account the access rights of different users for the hypertext structure modeling, we can obtain personalized hypertext views (see also section 3.8).

In the reviewing system example hypertext views are required for the following user roles: author, reviewer, and PC chair. Figure 3-7 shows the hypertext structure model for the PC chair's view. A PC chair can view all submitted papers. In addition, the PC chair can access the list of accepted or rejected papers, and the reviewer profiles. In line with the UWE modeling method, Figure 3-7 shows how the UML stereotype ≪navigation class≫ is used to mark classes representing nodes in the hypertext structure model to distinguish them from content classes. Links are modeled by directed associations with the stereotype ≪navigation link≫.

The literature defines various specific types of links to further refine the semantics of the hypertext structure model. For example, the HDM (Hypertext Design Model) method (Garzotto et al. 1995) specifies the following types of links:

- *Structural links* connect elements of the same node, e.g., from a review summary to the review details.
- *Perspective links* put various views of a node in relation to each other, e.g., the PostScript and the PDF versions of a paper.
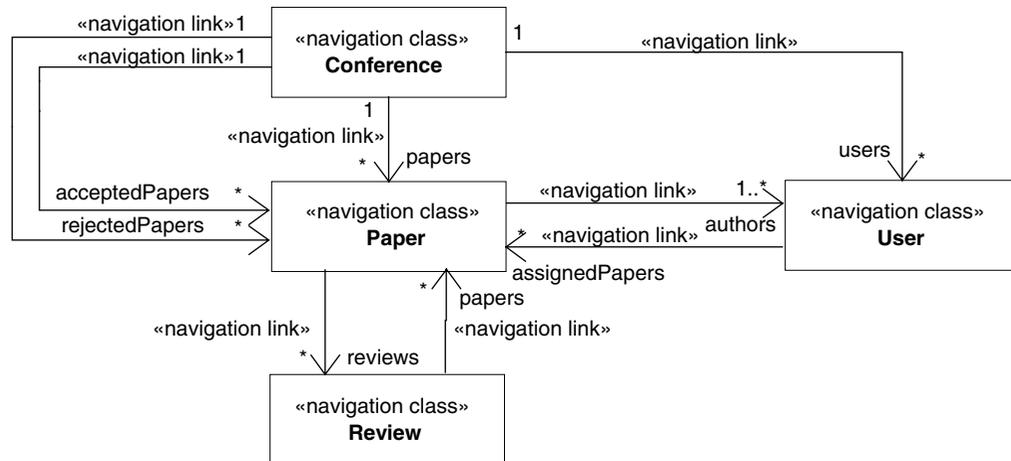
Figure 3-7   Hypertext structure model of the PC's view on the reviewing system.

- *Application links* put different nodes in relation to each other, depending on the application, e.g., a link pointing to ''best paper''.

Other classifications are based on the possible transport of information during navigation. For example, the WebML (Web Modeling Language) method (Ceri et al. 2003) specifies the following types of links:

- *Contextual links* carry context information, e.g., the unique number of a reviewer, to navigate from one reviewer to the reviews he or she created.
- *Non-contextual links* have no associated context information, e.g., links pointing from a single review to the list of all reviews.

With regard to the distribution of nodes on the hypertext level over pages on the presentation level (see section 3.7), WebML specifies additionally the following types of links:

- *Intra-page links* are used when the source and the destination of a link belong to the same page, e.g., when a link allows the user to directly navigate to the summary of a paper, which is displayed further down on the page.
- *Inter-page links* are used when the source and the destination are on different pages, e.g., when detailed information about the authors and their papers are on different pages.

Based on the functional requirements of Web applications, the UWE (Koch and Kraus 2002) modeling method defines the following types of links:

- *Navigation links* are used to navigate between nodes, e.g., links between papers and their authors.
- *Process links* point to the start node of a process, e.g., to the beginning of the review submission.

- *External links* point to a node not directly belonging to the application, e.g., to the formatting guidelines established by the publisher of the conference proceedings, which are not directly stored in the reviewing system.

The OO-H (Object-Oriented Hypermedia) modeling method (Gómez and Cachero 2003) defines five types of links as follows:

- *I-links (internal links)* point to nodes inside the boundaries of a given navigational requirement, e.g., internal links to review details of one of the reviewers.
- *T-links (traversal links)* point to nodes covering other navigational requirements, e.g. from an author to his or her papers.
- *R-links (requirement links)* point to a start of a navigational path, e.g., to add a new review.
- *X-links (external links)* point to external nodes, e.g., to external formatting guidelines.
- *S-links (service links)* point (with their corresponding response links) to services, e.g., to an external search engine.

3.6.3   Access Modeling Concepts

The hypertext structure model built so far alone is not sufficient to describe how nodes can be reached by navigation. To allow users to navigate to nodes the users need navigation and orientation aids. These are formulated in the form of *access structures* refining the hypertext structure model. Recurring access structures are described in (German and Cowan 2000, Lyardet et al. 1999, Rossi et al. 1998, Akanda and German 2005) as design patterns, also called ''hypermedia design patterns'' or ''navigation patterns''. The use of these navigation patterns helps to increase the quality of the hypertext model tremendously.

In our reviewing system example, if one wants to navigate from a reviewer to a paper assigned to this reviewer, one will have to identify this specific paper during navigation. For example, this could be realized in the form of a list showing all papers. Such a selection list for navigational support is also known as an ''index''. An *index* is an access structure which allows users to select a single object (i.e. one object of the content) out of a homogeneous list of objects. In contrast, a *menu* allows users to access heterogeneous nodes, or further menus (i.e. submenus). Other access structures are the guided tour and the query. A *guided tour* allows users to sequentially walk through a number of nodes. A *query* allows users to search for nodes. Most modeling methods offer dedicated model elements for the most frequently used navigation patterns. Special navigation patterns include *home*, which points to the home page of a Web application, and *landmark*, which points to a node that can be reached from within all nodes.

Some of these access structures can be added to the hypertext structure model automatically (Koch and Kraus 2002). For example, indexes can be added automatically whenever we want to allow access to a set (>1) of objects of a node.

Figure 3-8 shows a simplified access model of the PC chair's view specified in the hypertext structure model in our reviewing system. Note that a link's default multiplicity is 1. The PC chair has access to all papers, reviews, and users. To access a specific paper, a unique number is used. Alternatively, the PC chair can search for a paper by title. UWE uses UML stereotypes, i.e., ≪menu≫ (e.g., ''Conference''), ≪index≫ (e.g., ''ReviewingStatus''), ≪query≫
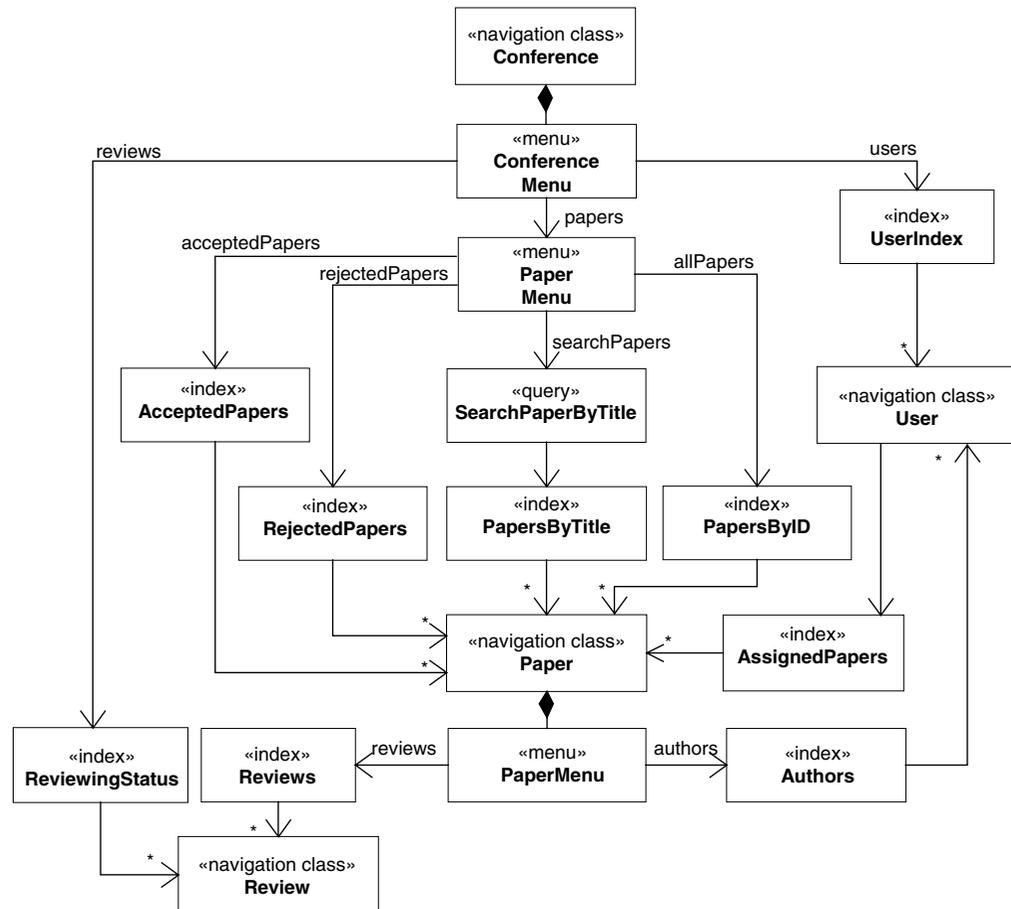
Figure 3-8 Simplified access model of the hypertext structure model from Figure 3-7.

(e.g., ''SearchPaperByTitle''), and ≪guided tour≫, to specify the menu, index, query, and guided tour access structures.

3.6.4 Relation to Content Modeling

Depending on the underlying modeling method, the hypertext model is more or less strongly dependent on the content model. There exists both a dependence at the type level, e.g., which classes in the content model form which node in the hypertext model, and at the instance level, i.e., which sets of objects in the content model populate that node in the hypertext model. Not all methods describe dependencies between the content model and the hypertext model exactly. Nevertheless, some methods specify a direct derivation of the hypertext from the content by defining nodes on the basis of views (in the sense of ''database views'') (Schwabe et al. 2002, Koch and Kraus 2002).

3.7   Presentation Modeling

Similar to traditional Software Engineering, presentation modeling deals with the user interface and thus with the look and feel of a Web application. In contrast to traditional applications, the central element of the presentation in Web applications is the page as a visualization unit.

3.7.1   Objectives

Presentation modeling is aimed at designing the structure and behavior of the user interface to ensure that interaction with the Web application is simple and self-explanatory. In addition, the communication and representation task of the Web application are taken into account. Presentation modeling generates a two-fold result: First, it produces a uniform presentation concept by modeling recurring elements on the pages, e.g., headers and footers. It should ideally show the composition of each page and the design of the fields, texts, images, forms, etc., included in these pages. Second, in addition to the structure of the pages, the presentation model describes the behavior-oriented aspects of the user interface, e.g., which button to click to activate a function of the application logic. Due to the wide variety of navigation options and the inherent risk of getting lost, care should be taken to give users appropriate orientation help on the presentation level. This can be achieved, for example, by displaying the current navigation path, or pages visited during the active session.

Not all methods available for modeling Web applications support technology-independent presentation modeling concepts; some rather use technology-specific concepts, such as Stylesheet languages, e.g., XSL (Extensible Stylesheet Language) (Pineda and Krüger 2003).

Another important factor for Web applications is the graphical layout design of the user interface. It is often produced by a graphic designer based on some basic drawings, or conceptualized by the tool-supported implementation of prototypical pages. Although this task is part of presentation modeling, it is currently not supported by modeling techniques. Chapter 11 discusses useful guidelines to design the user interface.

3.7.2   Concepts

Model elements are described on three hierarchical levels:

- A *presentation page* describes a page presented to the user as a visualization unit. It can be composed of different presentation units.
- A *presentation unit* serves to group related user interface elements, representing a logical fragment of the page. It presents a node stemming from the hypertext model.
- A *presentation element* is the basic building block of the presentation model. Presentation elements represent a node's set of information and can include text, images, audio, etc.

We can visualize the composition of presentation pages on the basis of a nested UML class diagram representation known as ''composition'', as in the example shown in Figure 3-9. This example uses the stereotype classes ≪page≫ and ≪presentation unit≫ to depict presentation pages and presentation units. Notice that all types of presentation elements are also designated by appropriate UML stereotypes. Figure 3-9 shows two presentation pages of our reviewing system. A paper is positioned on the page called ''PaperPage'' with the appropriate fields as well as a
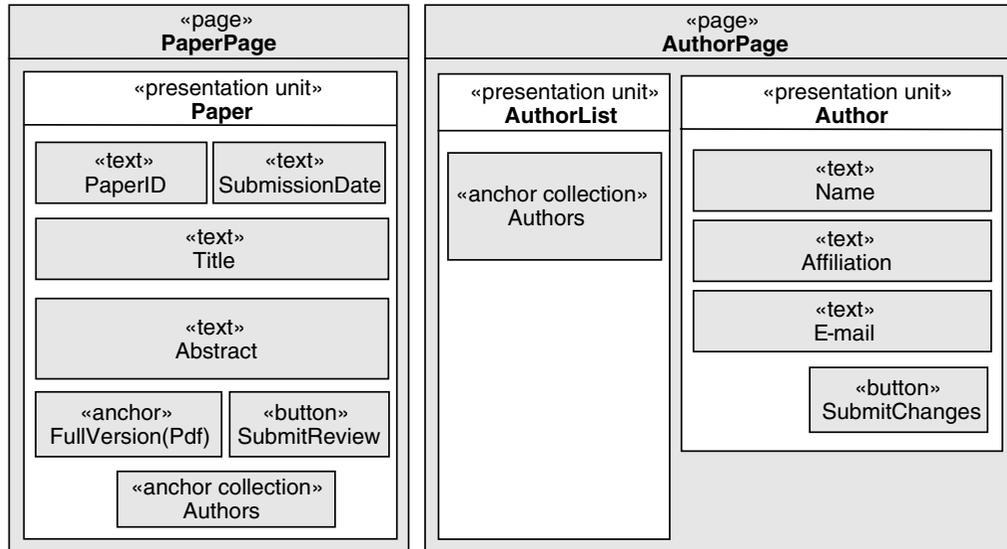
Figure 3-9   Presentation pages of the reviewing system.

link to the paper's full version and a link to display the paper's authors. Moreover, the user can press a button to add a new review. The page ''AuthorPage'' has two presentation units, i.e., the list of all authors and each author's detailed information.

Behavioral aspects of the user interface, such as a reviewer's interaction to navigate to the papers assigned to him or her for reviewing, can be modeled by means of behavior diagrams, as shown in Figures 3-10, 3-11, and 3-12. In general, a user's interaction with the Web application does not only involve the presentation level; it is also forwarded to the hypertext level and the content level, depending on the type of interaction. We can see in the simplified sequence diagrams in Figures 3-11 and 3-12, that a reviewer activates the navigation to the index of assigned papers by using the navigation bar from within the conference home page. This information is, in turn, composed of the relevant papers on the content level. The list allows the user to select a paper out of the list of assigned papers. The user can then navigate to select one paper, which will be displayed in the details view.

3.7.3   Relation to Hypertext Modeling

Similarly to mapping the content model to the hypertext model, we also have to specify how hypertext elements should be mapped to presentation elements. This is normally done under the assumption that all instances of a node will be displayed on the presentation level.

As mentioned before, the interactions triggered by a user are not necessarily limited to the presentation level only. For this reason, we have to additionally consider their correspondences to the other links. This correspondence may be in the form of objects and application logic on the content level, and for navigation on the hypertext level.
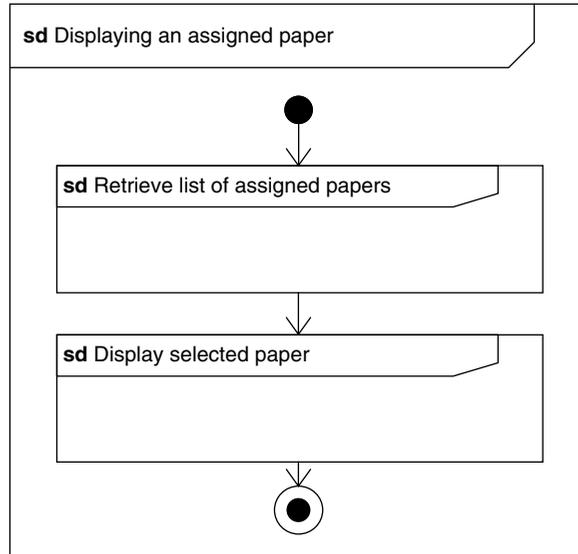
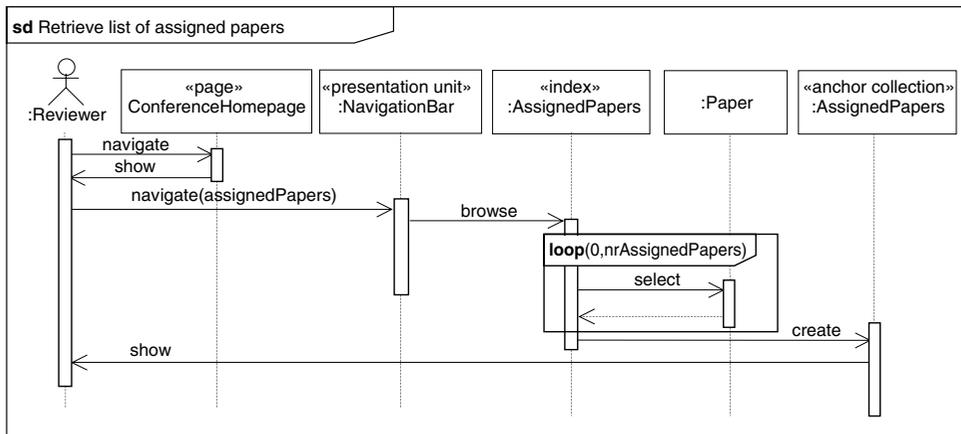**Figure 3-10** Interaction overview diagram of the reviewing system.

Figure 3-11 Sequence diagram for retrieving a list of assigned papers.

3.8 Customization Modeling

Since ubiquitous Web applications increasingly gain importance, the consideration of context information and an appropriate adaptation of the application as early as possible in the modeling phase (see also section 1.3.2) are required. Relevant proposals for *customization* originate from the fields of personalization (Kobsa 2001, Brusilovsky 2001) and mobile computing (Eisenstein
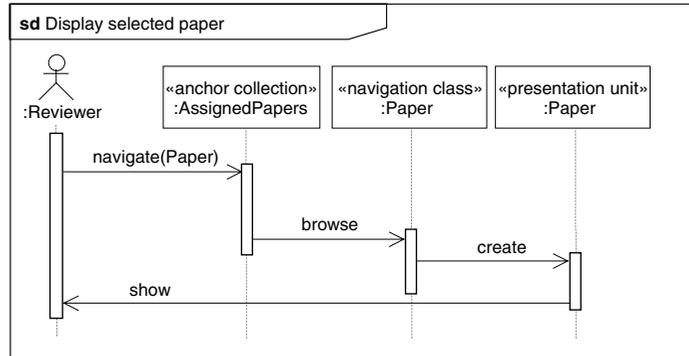
Figure 3-12   Sequence diagram for displaying selected papers.

et al. 2001, Want and Schilit 2001). For a more detailed overview on the origin of customization the reader is referred to (Kappel et al. 2003).

However, customization modeling is still a very young field, and only a few existing methods for Web application modeling offer a form of customization modeling (Fraternali 1999, Kappel et al. 2000, Ceri et al. 2003, Garrigós et al. 2005). For the UWE methodology (Baumeister et al. 2005) have recently proposed an aspect-oriented approach to deal with customization.

3.8.1   Objectives

Customization modeling is aimed at explicitly representing *context* information, and the *adaptations* derived from it. Depending on the modeling method the result is not always an explicit customization model. In most cases, customization modeling is intermingled with content, hypertext, and presentation models.

Customization has to be distinguished from maintenance or re-engineering. Customization modeling considers context information that can be predicted at modeling time which can assume different values when the Web application is run. In contrast, adaptation due to changes in the organizational or technological environment is part of maintenance or re-engineering activities.

3.8.2   Concepts

Customization requires examining the Web application's usage situation, i.e., dealing with the questions of ''what'' should be adapted and ''when''. To be able to personalize a Web application we have to model and manage the preferences and characteristics of a user in a so-called *user profile*. For example, to adapt a Web application in the field of mobile computing, we have to consider *device profiles*, *location information*, and *transmission bandwidth*. This information is then represented within the context model in form of a class diagram. At runtime, context can change, e.g., users change their preferences, or the application is ''consumed'' at different locations. This situation, in turn, is the reason why we have to adapt the Web application.

With regard to the abstraction level of the context information, one can distinguish between *physical context* and *logical context*. The physical context results from the respective usage

situation (e.g., a user's login name or the GSM cell in which a user is currently located). The logical context provides additional context knowledge (e.g., address at work versus address at home, working hours versus spare time). This context information can also be provided to the Web application by external sources. One example of such an external source that provides information for a more detailed specification of the location context are Geographic Information Systems (GIS). Initial approaches, such as the *ContextToolkit* (Abowd 1999) or the *NEXUS* project (Fritsch et al. 2002), have been proposed to support universal components capable of supplying different types of physical and logical context information.

The adaptation to a context can be modeled in either of two fundamentally different ways. First, it can be modeled in a *result-oriented* way by creating various models or model variants with regard to the different set of variants of the context information. This approach is known as *static adaptation*. The hypertext modeling example shown in Figure 3-7 describes a statically adapted hypertext structure to the context of the ''PC'' user role. The drawback of static adaptation is the exponential growth of model variants to be considered. Second, *dynamic adaptation* can be used. In contrast to static adaptation, dynamic adaptation adds context-dependent transformation rules to the content, hypertext, and presentation models. These transformation rules describe the variants to be created at runtime. For example, dynamic transformation rules, e.g., formulated as ECA (Event/Condition/Action) rules, could specify the addition or removal of model elements, or the filtering of instances, to create a personalized list with papers on the topics a user is interested in. Whether dynamic or static adaptation is the better approach depends on the use case. Dynamic adaptation has the benefit that it avoids the combinatory explosion of model variants. Its drawback is that the result, i.e., the model's variant adapted to the context, is not available directly, but will actually be created ''at runtime'', which makes it more difficult to understand the model. The reader will find a detailed discussion of proposals for customization modeling in (Kappel et al. 2003).

Figures 3-13 and 3-14 show how the hypertext and presentation levels of the review-ing system example can be dynamically adapted. We use annotations – stereotyped with ≪customization≫ – to add customization rules to the adapted class. The rules described informally in this example can be specified in more detail by using a formal language, e.g., the Object Constraint Language (OCL) (Baerdick et al. 2004), in further refining steps. Figure 3-13 shows an example of how the hypertext structure can be customized so that the papers a user can read are limited to those with topics of interest to that user. The elements of the access structure, ''InterestingPapers'', are adapted dynamically by transformation rules based on personal topics of interest.

The example in Figure 3-14 shows how elements of the presentation model can be adapted by the use of transformation rules. Specifically, the button ''Enter Review'' should be visible only for users with the ''Reviewer'' role.

Most of the currently existing methodologies tackle the modeling of customization by defining rules or a filter for each point in the Web application where customization applies as has been shown in the previous examples. A different approach is to consider customization as a cross-cutting concern. UWE follows such an approach using aspect-oriented modeling (AOM) techniques (Baumeister et al. 2005). AOM allows on the one side for a systematic separation of the system functionality from the customization aspects, and on the other side it allows for reduction of redundancy.
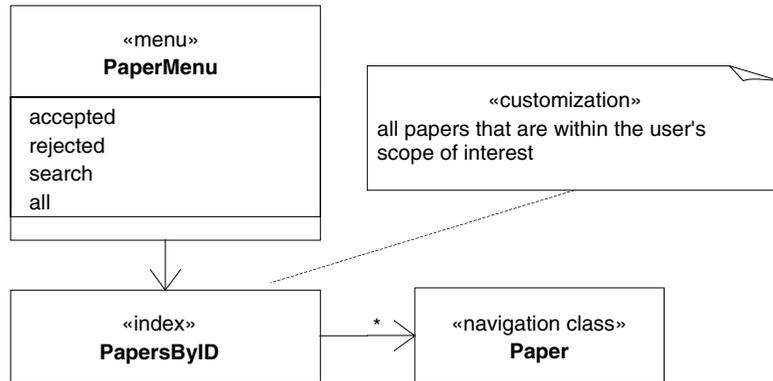
«menu»
PaperMenu

accepted
rejected
search
all

«customization»
all papers that are within the user's
scope of interest

«index»
PapersByID

*

«navigation class»
Paper

Figure 3-13   Dynamic adaptation of an index in the hypertext model.

«page»
PaperPage

«presentation unit»
Paper

«text»
PaperID

«text»
SubmissionDate

«text»
Title

«text»
Abstract

«anchor»
FullVersion (Pdf)

«button»
EnterReview

«anchorcollection»
Authors

«customization»
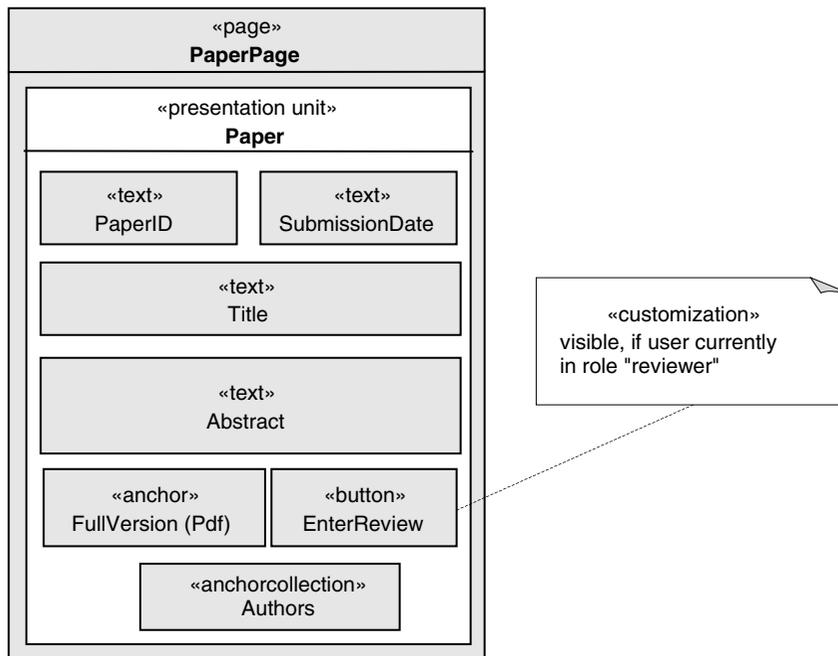visible, if user currently
in role "reviewer"

Figure 3-14   Dynamic adaptation of a page in the presentation model.

UWE materializes the cross-cutting concern by the use of stereotyped UML packages for the *pointcut* part and the *advice* part of an aspect. An aspect is a (graphical) statement saying that in addition to the features specified in the principal models, each model element of the package pointcut also has the features specified by the advice. In other words, a complete description including both general system functionality and additional, cross-cutting features is given by the composition – so-called weaving – of the main model and the aspect.

3.8 Customization Modeling

UWE distinguishes between customization at content, hypertext and presentation level (as represented in Figure 3-2). For example, links can be annotated, sorted, hidden or generated dynamically according to the current status of a user or context model. The approach consists of extending the UWE metamodel with a modeling element *NavigationAnnotation* that can be attached to any navigation link (Baumeister et al. 2005). Figure 3-15 shows how the designer used the *NavigationAnnotation* to add an attribute (*PresStyle*) included in the advice part to the set of links included in the pointcut part. Figure 3-16 shows the results of the weaving process.
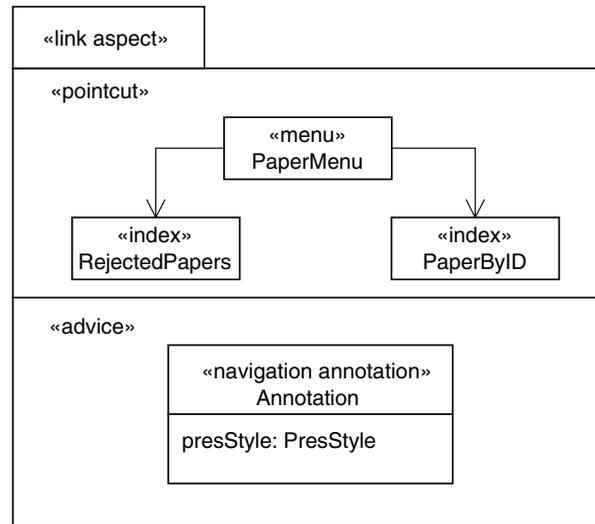
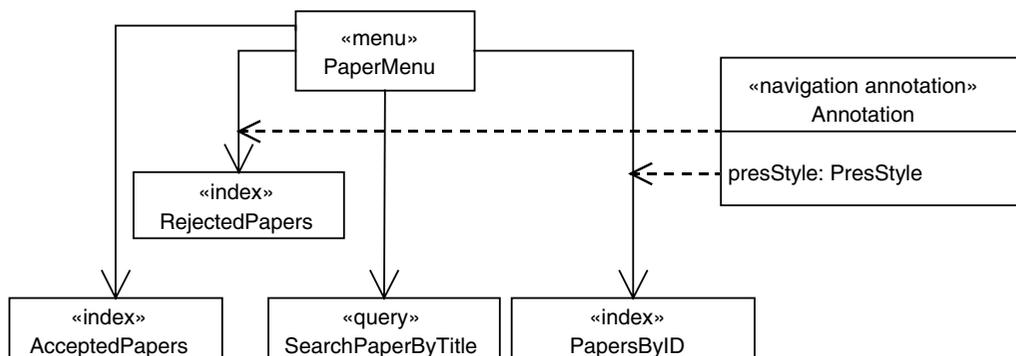**Figure 3-15**   Modeling adaptation with aspects.

Figure 3-16   Results of the waving process.

Dynamic customization can be achieved including rules written in OCL in the advice part of the aspect.

3.8.3   Relation to Content, Hypertext, and Presentation Modeling

As shown in Figure 3-2, customization can influences all levels of a Web application modeling process. Changes can be limited locally to one level or affect several levels. Separation of the customization model from the content, hypertext, and presentation models are recommended with regard to changeability, flexibility, and encapsulation, but most existing methods don't provide for this separation. And sometimes, as in the reviewing system, such a separation is difficult. For example, the user of a Web application and his or her preferences are context modeling issues, but the user may also be an issue in content modeling, as the reviewing system example shows.

3.9   Methods and Tools

All modeling methods offer a set of modeling elements tailored to the requirements of Web applications (Retschitzegger 2000). Almost all methods offer a specific notation for the modeling elements. In addition, many methods define a process and are supported by a tool that (semi) automatically generates an implementation from the created models (see Table 3-1).

3.9.1   Modeling Methods: An Overview

Methods available for Web application modeling are normally based on traditional methods, such as *ER*, or they enhance an object-oriented modeling language, e.g., *UML*. More recent methods usually build on the strengths of earlier methods. Consequently, we can describe the history of Web application modeling methods as shown in Figure 3-17 (based on Escalona 2004).

Modeling methods follow different paradigms, depending on their origin and focus:

- *Data-oriented methods* originate from the field of database systems; they are mainly based on the ER model enhanced by specific concepts for modeling on the hypertext level. The primary focus of these methods is the modeling of database-driven Web applications. Examples of data-oriented methods include the *Relationship Management Methodology* (*RMM*) (Isakowitz et al. 1998), *Hera* (Houben et al. 2004) and the *Web Modeling Language* (*WebML*) (Ceri et al. 2003, Brambilla et al. 2005).
- *Hypertext-oriented methods* center on the hypertext character of Web applications; they emerged mainly from the field of hypertext systems (Lowe and Hall 1999). Representatives of this group are the *Hypertext Design Model* (*HDM*) (Garzotto et al. 1995), which has been extended into *W2000* (Baresi et al. 2001), and *HDM-lite* (Fraternali and Paolini 1998), or the *Web Site Design Method* (*WSDM*) (De Troyer and Decruyenaere 2000, Plessers et al. 2005).
- *Object-oriented methods* are based on either OMT (for the very early methods) or UML. UML is the preferred notation when a standard language for modeling is selected. This category includes the *Object-Oriented Hypermedia Design Method* (*OOHDM*) (Schwabe and Rossi 1998), *UML-based Web Engineering* (*UWE*) (Koch and Kraus 2002, Koch and Kraus 2003), *Object-Oriented Web Solutions* (*OOWS*) (Pastor et al. 2005) and the *Object-Oriented Hypermedia* (*OO-H*) method (Gómez and Cachero 2003).
- *Software-oriented methods* look at Web applications mainly from the perspective of traditional software development, using techniques that strongly follow classical Software

3.9 Methods and Tools 59

Engineering. Good examples for this category are *Web Application Extension* (*WAE*), or *WAE2*, its enhanced version (Conallen 2003).
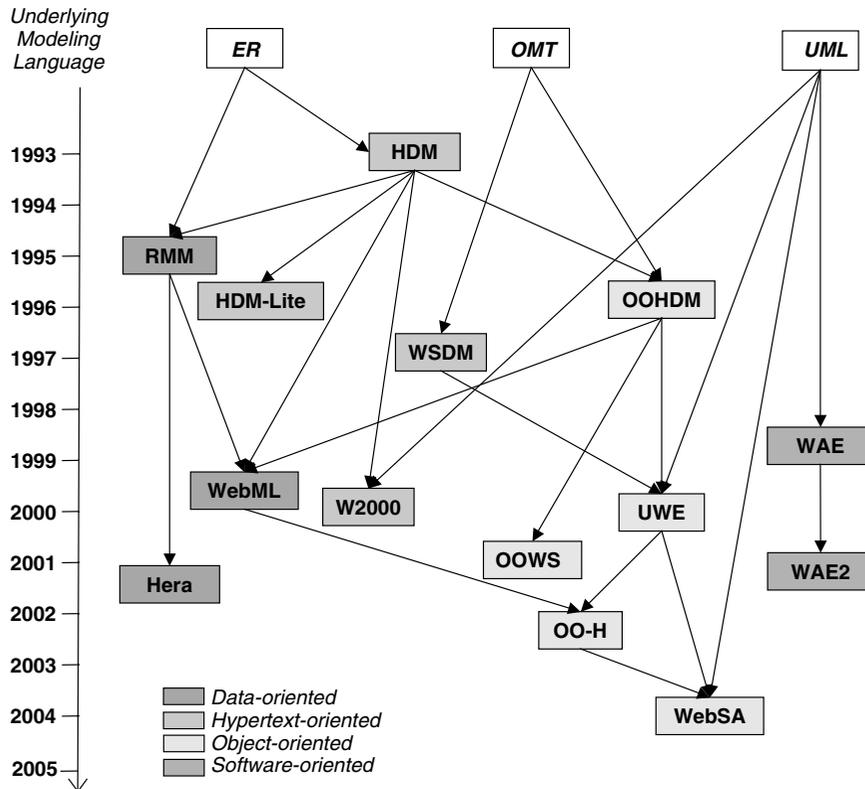

Figure 3-17 Historical development of methods for Web application modeling.

Table 3-1 gives a more detailed overview of these methods. In summary, it shows the notations used, whether the methods are still being developed further, and what modeling dimensions (see section 3.3) they support. Moreover, the table shows whether there is tool support for modeling and/or code generation purposes. Finally, the table shows the strengths of the methods.

HDM-lite – the successor of HDM – was designed focusing on automation of the development process and with automatic generation of Web applications in mind. *W2000*, another enhancement of *HDM*, models a Web application from hypertext-centric and user-centric perspectives. *RMM* is a method that builds on the ER model and defines a gradual process for the successive refinement of models. Another approach based on the ER paradigm is *Hera* which uses the RMM notation. *WebML* is an easy-to-understand and mature modeling language for data-intensive Web applications which provides support for all essentials of Web application modeling. This method uses the *WebRatio* tool (see below) to support both modeling and the automatic generation of code. Due to the fact that *OOHDM* heavily emphasizes the concept of navigation context, this method is recommended for Web applications which use a wide range of different

Table 3-1   Overview of methods for Web application modeling.

Modeling Method	Modeling Paradigm	Notation	Evolving	Requirements Modeling	Content Modeling	Hypertext Modeling	Presentation Modeling	Customization Modeling	Structure and Behavior	Process / Approach	Tool Support	Generation	Strengths
HDM-lite	HT	ER + own notation	×	×	×	✓	✓	×	s	own	generation tools	auto	process for model transformation, automatic generation
Hera	DB	ER + RMM+ own notation	✓	×	✓	✓	✓	✓	s + b	own	authoring & generation tool	semi	model-driven development
OO-H	OO	UML + own notation	✓	✓	✓	✓	×	pers	s + b	own	modeling- & generation tool	auto	tool for automatic generation
OOHDM	OO	UML + own notation	✓	✓	✓	✓	✓	pers	s + b	own	×	×	powerful concepts for contextual navigation, personalization
OOWS	OO	UML + own notation	✓	✓	✓	✓	✓	×	s + b	own	modeling- & generation tool	auto	advanced (commercial) tool for automatic generation
RMM	DB	ER + own notation	×	×	✓	✓	✓	×	s	own	authoring tool	semi	hypertext modeling based on ER-model, predefined process
UWE	OO	UML	✓	✓	✓	✓	✓	pers	s + b	RUP	extended UML tool & generation tools	semi	UML-based method, model-driven development, aspect-oriented customization
W2000 (HDM)	HT	UML	✓	✓	×	✓	✓	pers	s	×	extended UML-tool	×	user-centric hypertext modeling
WAE2 (WAE)	SW	UML	✓	✓	✓	×	✓	×	s + b	RUP	standard UML-tools	×	implementation design, architectural design
WebML	DB	ER, UML	✓	✓	✓	✓	×	pers	s + b	own	modeling- & generation tool	auto	well-elaborated notation, database integration, generation
WS DM	HT	own notation	✓	×	✓	✓	×	×	s + b	own	×	×	user-centric approach for analysis

✓ supported	pers personalization	RUP Rational Unified Process		DB data-oriented
× not supported		own own process model / approach		HT hypertext-oriented
	s structure modeling			OO object-oriented
	b behavior modeling	auto automatic generation		SW software-oriented
		semi semi-automatic generation		

navigation access options. OOHDM has been expanded to support personalization, framework modeling (Schwabe et al. 2001), Web application architectures and diagrams to capture user interaction scenarios, now offering many innovative concepts. *WSDM* focuses on a methodic approach oriented towards user requirements. *UWE*, on the other hand, is an approach that offers a UML-based notation and meta-model based model consistency checking. *OO-H* is one of

the more recent methods, combining the benefits of WebML, OOHDM, and UWE. The OO-H method uses a tool called *VisualWADE* (see below) to support model-driven automatic code generation. *OOWS* in the same way as OO-H, is an object-oriented approach that use some concepts of the is partially based on the UML, but it mainly uses its own notation. *WAE2* is a UML approach that focuses on the distribution of the application logic. And finally, *WebSA*, is an approach for modeling Web architectures (cf. section 3.9.2).

3.9.2   Model-Driven Development

The Model-Driven Development (MDD) approach not only advocates the use of models (such as those described in the previous sections) for the development of software, but also emphasizes the need of transformations in all phases of the development, from system specification to implementation and testing. Transformations between models provide a chain that enables the automated implementation of a system in successive steps right from the different models defined for it.

The development of Web applications is a specific domain in which MDD can be successfully applied, due to the Web specific characteristic separation of concerns – content, hypertext, presentation and customization. Methods such as WebML, OO-H and UWE constitute a good basis for model-driven approaches for the development of Web applications. They already include some semi-automated model-based transformations.

WebSA (Web Software Architecture) is another specific model-driven approach for the Web domain, which is based on the MDA (Model-Driven Architecture) paradigm. WebSA – analogous to MDA (OMG 2005) – emphasizes the construction of platform independent models (PIMs) and the subsequent automated building of platform specific models (PSMs), where PSMs constitute the basis for the generation of executable code (Meliá and Cachero 2004).

The WebSA methodology proposes a development process made up of a set of UML models used to represent the Web application and QVT[1] (OMG 2005) transformations as shown in Figure 3-18. In a first step the transformations (T1) are a mechanism to integrate architectural models and the structural and behavioral models of a Web application (such as those described in sections 3.5 to 3.8). These structural and behavioral models are called functional models in the WebSA approach. They focus on the functional requirements of a Web application in contrast to the architectural models which are based on the non-functional requirements of the application. A UML profile for Web specific architectural elements is part of WebSA (Meliá et al. 2005). The result of the merge of the functional (content, hypertext and presentation) and the architectural models (configuration and subsystem view of the system) in the first step is an integration model. This integration model is still a platform independent model. The next step consists in PIM to PSM transformations (T2) in order to obtain platform specific models, such as for J2EE or .NET (see Figure 3-18).

3.9.3   Tool Support

Due to short development cycles and the complexity of Web applications, it is recommended to use tools that support not only the modeling itself, but also and particularly automatic

1   Query/Views/Transformations.
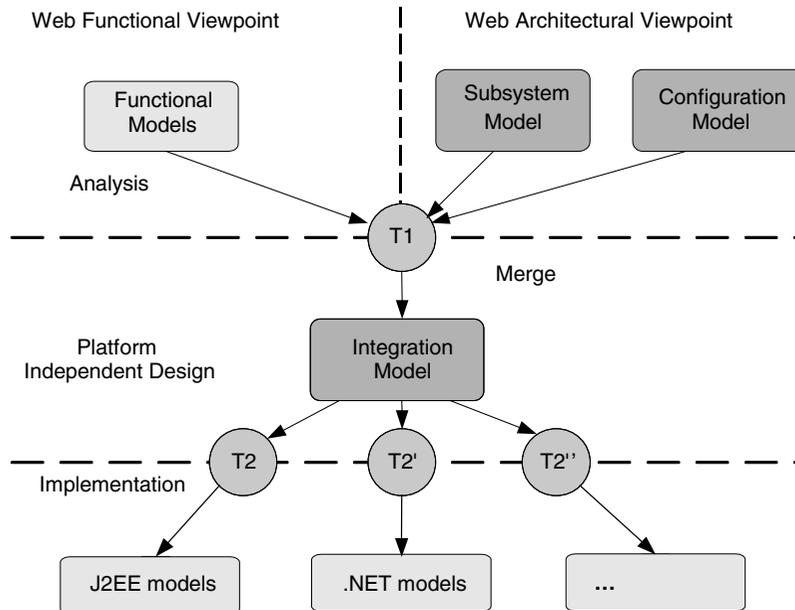
Figure 3-18   WebSA Development Process (Meliá and Cachero 2004).

code generation and model consistency check. The following subsections describe *WebRatio Site Development Studio, VisualWADE*, and the *OpenUWE Suite* as examples for this type of tools.

WebRatio Site Development Studio

The *WebRatio Site Development Studio* (www.webratio.com) is a model-based development tool that builds on the *Web Modeling Language* (*WebML*) (www.webml.org). This tool uses its own notation for hypertext modeling and additionally supports the ER notation and UML. The tool's code generator uses XSL to transform content and hypertext models represented in XML into the required database representation and database connections as well as software components and different output formats (HTML, WML, PDF, Microsoft Word). WebRatio uses a tool called *EasyStyle* to generate the presentation of pages, which will transform annotated pages into XSL stylesheets automatically, without additional programming activities. The Web application generated by WebRatio is deployed in a runtime framework based on a set of Java components, which can be configured by use of XML files. The runtime architecture is based on the MVC-2 design pattern (see Chapter 4) and is suited for the *Jakarta Struts* open-source platform and for *JSP* tag libraries.

VisualWADE

The *VisualWADE* (www.visualwade.com) tool is based on the OO-H method. This tool supports modeling and automatic generation of applications based on XML, ASP, JSP, and PHP.

VisualWADE augments a UML model with two additional models: ''Navigation View'' is used to model the hypertext aspect of a Web application, and ''Presentation View'' represents interaction elements of the user interface with regard to its structure and behavior using a number of template structures. This yields a device-independent description of the user interface. This description can be used by generators to automatically generate the Web application for different runtime environments and devices.

OpenUWE

The *OpenUWE* (`www.pst.ifi.lmu.de/projekte/uwe`) tool suite is a development environment for the design and generation of Web applications using the UWE methodology. The main feature of this suite is its open architecture based on established standards. These standards are supported by both open-source and commercial tools. In its current version (still under development), this development environment includes the ArgoUWE case tool and the UWEXML framework, consisting of a model consistency checker, a layout editor, and code generators for the Cocoon XML Publishing Framework (Ziegeler and Langham 2002) and Java Server Pages. The common data exchange language within this architecture is based on the extensible UWE meta-model.

The ArgoUWE case tool is based on the open source CASE tool ArgoUML (`www.argouml.org`). ArgoUWE supports not only the UWE notation, but also checks the consistency of the models according to the OCL constraints specified for the UWE metamodel. Consistency checking is embedded into the cognitive design critique feature of ArgoUML and runs in a background thread, so that the user is warned of model deficiencies but not interrupted. A mapping mechanism defined between the meta-model and the UWE UML profile can be used to alternatively create models with any other standard UML CASE tool. Model exchange is based on XMI (XML Metadata Interchange).

3.10   Outlook

A large number of different methods for Web application modeling have been developed in the last decade. However, some methods will probably converge during the course of further development. It is currently hard to predict how far this converging trend will go and whether it will eventually lead to a ''Unified Web Modeling Language'', similarly to the development of the UML. It is, however, uncontested that there is a trend towards using UML as notation language. Some methods are moving from their proprietary notation to a UML compliant one and introduce a UML profile for their method.

The methods that will succeed will be determined by the tool support offered for their modeling method. In the future, tools will systematically support not only the notation, but also the development process allowing for a model-driven development approach. However, this means that the methods will have to define clear guidelines and approaches, in contrast to the current situation. It also means that agile approaches (Ambler 2002) will most likely have to be considered, but they will have to be harmonized in view of (semi) automatic generation.

Furthermore, novelties in the field of model-driven development following the OMG standards for model-driven architecture (MDA) and Query/View/Transformation (QVT) will have an impact on further developments of methods and tools. This is not least since we have to especially deal with Web applications that use different heterogeneous runtime platforms and publishing frameworks.

Another point of interest for model reuse will be the modeling of Web application families and Web application frameworks. The introduction of OOHDM frame (Schwabe et al. 2001) represents an initial step in this direction.

Particular attention has recently been placed on the inclusion of workflow concepts in Web application modeling to meet the increasingly transactional requirements of Web applications (Brambilla et al. 2003).

The consideration of a Web application's context as early as in the modeling stage will become more and more important as the use of mobile devices for Web applications continually increases. Currently, however, only a few approaches consider customization comprehensively (Kappel et al. 2003, Baumeister et al. 2005). It can be expected that, as the complexity of models increases, their quality and quality assurance will become virulent issues (Comai et al. 2002).

The inclusion of Web services in model-based Web application development projects will bring new challenges, the most critical probably being the interplay between top-down modeling and bottom-up integration of existing services and adequate tool support (Brambilla et al. 2003), Brambilla et al. 2005).