The Authoring Process of the UML-based Web Engineering Approach

Nora Koch [1,2], Andreas Kraus[1], Rolf Hennicker[1]

[1] Institute of Computer Science
Ludwig-Maximilians University of Munich
Oettingenstr. 67, D-80538 München, Germany
{hennicke,kochn,krausa}@informatik.uni-muenchen.de

[2] F.A.S.T. Applied Software Technology GmbH
Arabellastr. 17, D-81925 München, Germany
koch@fast.de

Abstract

We propose a precise UML-based authoring method for Web applications. This authoring method is part of the UML-based Web Engineering (UWE) approach. Starting with a requirement analysis done by use cases technique, it focuses on the design phase. The conceptual model of the application is used as guideline for modeling the navigation space. From the navigation space model we derive, in a next step, a navigation structure model which shows how to navigate through the navigation space using access elements like indexes, guided tours, queries and menus. Finally, a presentation model is constructed based on the navigation structure model. It provides an appropriate UML notation to support sketching and storyboarding. In addition, we suggest to use UML interaction diagrams to represent the presentation flow. During the whole development process we identify steps that can be performed in an automatic way thus providing the basis for a generation mechanism for Web application design. The different models of the design process are represented by using a UML conform extension of UML for Web applications (UML profile). The authoring process is illustrated with an example: a Web–based conference review system.

The strength of the presented Web engineering approach is given by the fact that we use exclusively the UML notation and techniques. Moreover, our specification of constraints with OCL (part of UML) allows augmenting the exactitude of the models. In the same way our methodology has a high degree of precision in the description of guidelines of the authoring process of Web application, which can even partially be automated.

Keywords: Web Engineering, Unified Modeling Language, Web applications, Authoring Process, Design Method, Systematic Development, UML Extension

1 Introduction

The development of Web applications demands, as far as certain aspects are concerned, a different process than the software engineering approaches in use for traditional software. Web engineering is the new and improving discipline that deals with this development process. Web engineering can be defined, adapted from the definition given by the IEEE (1991) for software engineering, as the systematic, disciplined, quantifiable approach to the development, operation and maintenance of Web applications (Lowe & Hall, 1999).

The UML-based Web Engineering (UWE) approach presented by Koch (2000) supports Web application development with special focus on systematisation and personalisation. It is an object-oriented approach based on the two dimensions principle, i.e. time and content, of the Unified Process (Jacobson, Booch & Rumbaugh, 1999). The process that covers the whole life-cycle of Web applications is defined on the basis of phases, workflows and milestones. The phases are the inception, elaboration, construction, transition and maintenance. The building stones used in the process definition are workers, activities and artifacts (results). Workers are expert roles responsible for the activities to be performed during each phase and in each iteration with the goal to produce defined and incremental results. The set of artifacts includes models, pieces of code and documents.

Part of the UWE approach is a systematic authoring process that consists of a notation and a method. The notation used is the Unified Modeling Language (UML, 1999) and a “lightweight” UML profile we developed in previous works (Baumeister, Koch & Mandel, 1999, and Hennicker & Koch, 2000, and Koch, 2000). A UML profile is a UML extension based on the extension mechanisms defined by the UML itself. This profile includes stereotypes defined for the modeling of navigation and presentation aspects of Web applications. They are used to indicate the descriptive and restrictive properties of the modeling elements.

The method provides guideline for the systematic and stepwise construction of models. The construction is performed in an iterative and incremental design process. The modeling activities are the requirements analysis, conceptual, navigation, and presentation design. The method recommends the use of constraints written in the Object Constraint Language (OCL) to augment the precision of the models. Within UML, OCL is the standard for specifying invariants of classes and, pre-conditions and post-conditions of operations.

Many similarities with other methods for hypermedia and Web design are not a coincidence. Our goal is not to provide a new methodology defined from scratch, i.e. “yet another method”. In contrast, our objective is to combine well proved aspects of existing methodologies and to improve the resulting combination with some new ideas. For example, for our notation we use some graphical elements of RMM (Isakowitz, Stohr & Balasubramanian, 1995). We keep the separate construction of conceptual, navigation and presentation models that stems from OOHDM (Rossi, Schwabe & Lyardet, 2000) and continue with the user-centred approach of WSDM (De Troyer & Leune, 1997). We formalise sketching and storyboarding techniques widely used by user interface designers without a precise notation (Preece et. al., 1994, Sano, 1996, and Schneiderman, 1998).

The main aspects of our approach are:

- the use of a standard notation, i.e. UML through all the models,
- the precise definition of the method, i.e. the description of detailed guidelines to follow in the construction of the models,
- the specification of constraints, i.e. augmenting the precision of the models.

The use of UML has the advantage of being a well-documented modeling language, which is a de facto industrial standard and the most used object-oriented notation nowadays. The advantage of using a UML profile, in our case an extension with restrictive stereotypes (Berner, Glinz & Joos, 1999), is that any practitioner with a general UML background is able to understand a model based on this specialisation. Selic (1999), the author of the UML profile for the real-time domain, stresses that the resulting language of a UML profile remains compact, because the refinements fully respect the general semantics of their parents concepts and retain its “universal” quality. Conallen (1999) provides a UML extension that focuses on current implementation techniques and architecture aspects of Web applications. It does not consider separation of navigation and presentation aspects.

The specification of constraints has the advantage to augment the models precision, e.g. navigation and presentation restrictions. Another currently available formal language, such as Objective-Z or VDM++ could have been chosen. According to Warmer and Kleppe (1999), OCL is easy to learn for people who have not strong mathematical background, although it is underpinned by mathematical theory and logic.

The definition of the method is based on detailed guidelines for the development It allows a precise and systematic, even in some steps automated construction of Web Applications.

As a running example to illustrate the UWE authoring process, in this work the Web application of a *Conference Review System* is built. The purpose of the Web application is to support the process of submission, evaluation and selection of papers for a conference. A complete description of the sample problem is given in (Schwabe, 2001). The characteristics of the application needed to understand the models presented in each section are explained in the corresponding subsections "sample problem".

This paper is structured as follows: Section 2 gives an overview of the authoring process. Section 3 describes the requirement analysis performed with use cases. In Section 4, a UML representation of the conceptual model is shown. Section 5 gives guidelines for the systematic building of a navigation space model, also represented with UML class diagrams. Section 6 presents the generation of a UML navigation structure model based on the navigation space model. In Section 7, the presentation model is constructed showing sketches, storyboards, and the dynamic interaction between user and Web application. UML static view and interaction views are used in the presentation design. Finally, Section 8 presents some concluding remarks and an overview of future work.

2 Overview of the UWE Authoring Process

The authoring process consists of four steps. These steps are the requirements analysis, conceptual, navigation and presentation design. They produce the following artifacts:

- use case model
- conceptual model
- navigation space model and navigation structure model
- presentation model

These models are refined in successive iterations of the UWE development process. Figure 1 shows the models represented as UML packages related by trace dependencies (process relationship).

The goal of the requirements analysis is to find the functional requirements of the Web application and to represent these requirements as use cases.

The objective of the conceptual design is to build a conceptual model of the application domain taking into account the requirements captured with use cases. Traditional object-oriented techniques are used to construct the conceptual model, such as finding classes and associations and defining inheritance structures. The conceptual model is represented by an ordinary UML class diagram.

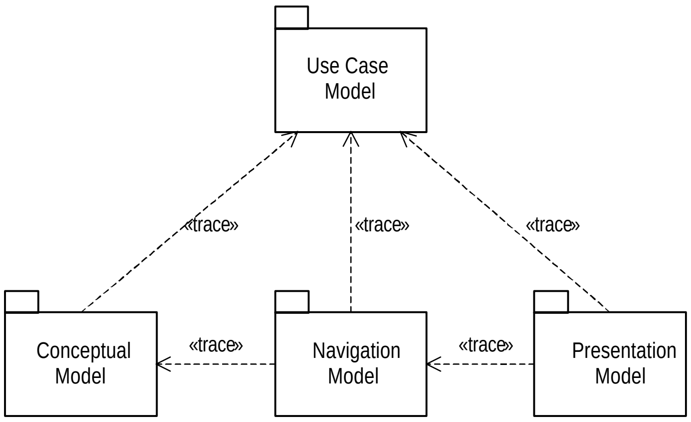

Fig. 1: Models Built during the Authoring Process of the UML-based Web Engineering Approach

Based on the conceptual model the navigation method proposes a set of guidelines to construct a navigation model which represents the navigation space and the navigation structure by adding access elements that can be used for navigation. The method includes a set of UML stereotyped modeling elements for navigation design, like indexes, guided tours, queries and menus. These stereotypes are used in the construction of UML class diagrams to represent the navigation space model and the navigation structure model.

Presentation modeling aims at the design of abstract user interfaces and the design of the user interaction with the Web application. It consists of two steps: The first step in the presentation design defines user interface views which sketch the content and the look and feel of the nodes. These user interface views can then be combined to storyboarding scenarios. The second step focuses on the dynamics of the presentation represented with UML sequence diagrams.

3 Requirement Analysis with Use Cases

Following the Unified Software Development Process of Jacobson, Booch and Rumbaugh (1999) we propose use cases for capturing the system's requirements. It is a user-centred technique that forces to define who are the users (actors) of the application and offers an intuitive way to represent the functionality an application has to fulfil for each actor.

3.1 Modeling elements

The main modeling elements used for use case modeling are: *actors* and *use* cases. They can be related by *inheritance*, *include* or *extend* relationships. All these modeling elements as well as the package and view mechanisms are used with the semantics defined in the UML (1999) and graphically represented with the UML notation.

3.2 Sample Problem

Based on the textual description of the requirements of the *Conference Review System* (Schwabe, 2001) we recognise that users can act in the following roles: program committee chair

(*Chair* for short), program committee member (*Member* for short), *Reviewer*, *Author* and *Co-author*. These are therefore the actors of the Use Case Model. *Reviewers* can be modelled as a generalisation of *Members* (see Figure 4).

The number of identified use cases lead to a set of views of the use case model or, in other words, the division of the use case model in packages. Figure 2 shows three use case packages – *Submission*, *Review* and *Administration*, which are detailed – in Figure 3 to Figure 5.

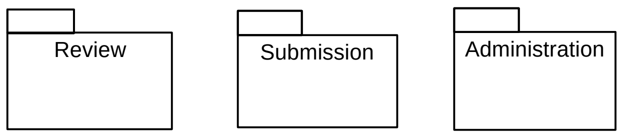

Figure 2: Use Case Model

Figure 3 depicts the use cases related to the submission of papers. These use cases are initiated by the *Author*. Figure 4 shows the actors and use cases that are identified as relevant for the review process performed by the *Members* and the *Reviewers*. Figure 5 shows the *Administration* package. The *Chair* is responsible for the use cases included in this package.

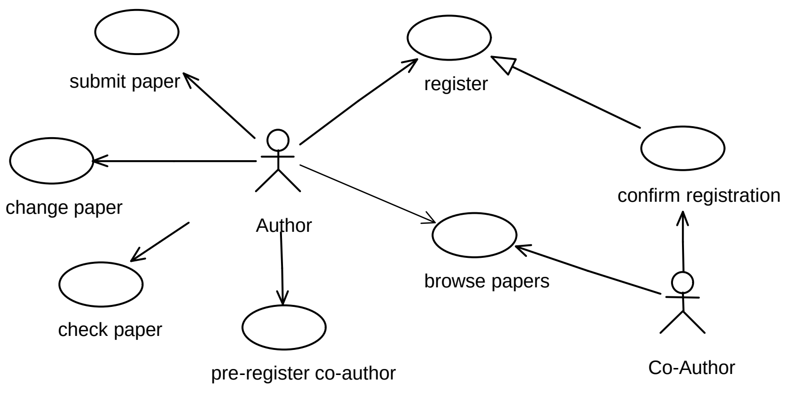

Figure 3: Submission Use Case Package

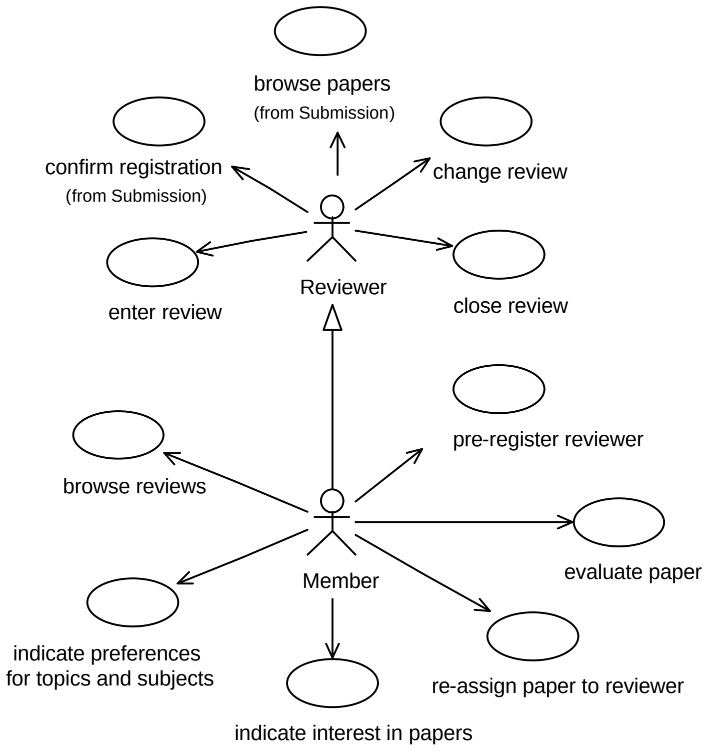

Figure 4: Review Use Case Package

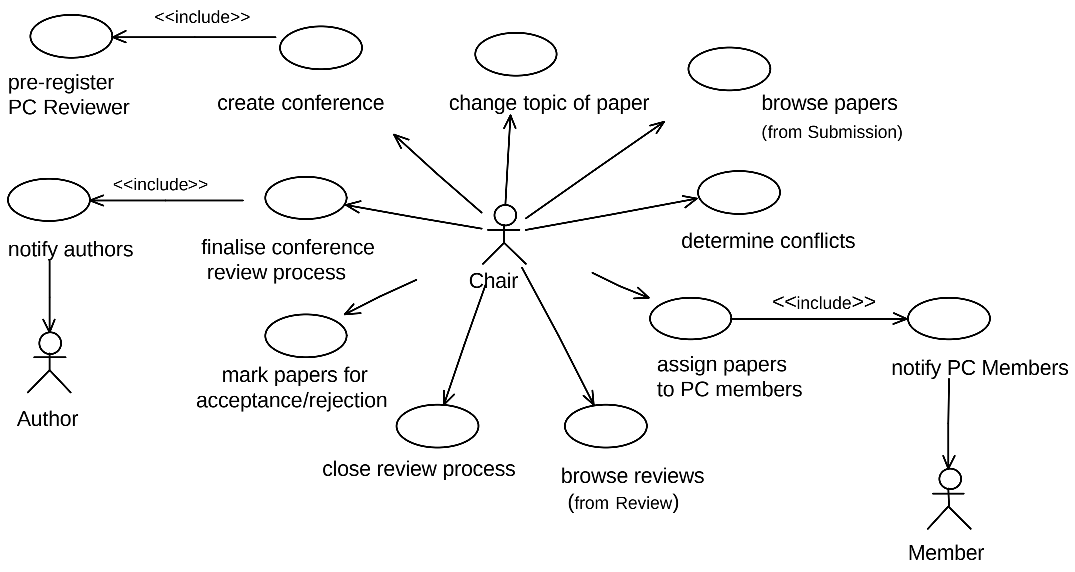

Figure 5: Administration Use Case Package

3.3 The Method

We apply the steps suggested by many use case driven processes (Kruchten 1999, Conallen, 1999, etc.) to build the use case model of a Web application. These steps are:

1. Find the actors.
2. For each actor search the text for activities the actor will perform.
3. Group activities to use cases.
4. Establish relationships between actors and use cases.
5. Establish “include” and “extends” relationships between use cases.
6. Simplify the use case model by defining inheritance relationships between actors and/or between use cases.

For each use case a detailed description can be provided in terms of (primary and secondary) scenarios, for instance following the guidelines of Schneider and Winters (1998). The activities flow of activities related to a use case can be represented by a UML activity diagram.

4 A UML Representation of the Conceptual Model

The conceptual design is based on the requirement analysis of the previous step. It includes the objects involved in the interaction between user and the application, i.e. specified in the use cases. The conceptual design aims to build a class model with these objects, which attempts to ignore as many of the navigation paths, presentation and interaction aspects as possible. These aspects are postponed to the navigation and presentation steps of the authoring process.

4.1 Modeling Elements

The main UML modeling elements used in the conceptual model are: *class, association and package.* These are represented graphically using the UML notation (1999). If the conceptual model consists of many classes, it is recommended that they be grouped using the UML *package* modeling element.

- *Class*

  A class is described by a name, attributes, operations and variants. The optional compartment named *variants* can be added to classes (Koch, 2000). It contains additional information used for *adaptive content* functionality, i.e. to present different or additional content to the users according to their user profile. The graphical representation of a class with variants is depicted in Figure 6.

- *Association and Package*

  Associations and packages are used as in standard UML class diagrams.

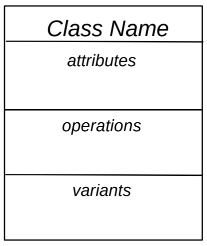

Figure 6: Class with Additional Compartment Variants

Classes defined in this step are used during navigation design to derive nodes of the hypermedia structure. Associations will be used to derive links.

4.2 Sample Problem

The *Conference Review System* offers information about papers, paper reviews and users of this Web application. The users are the actors identified during the requirement analysis. Which information is presented to each user of the system depends on the role of user and on the current status of the conference, i.e. the system administrates time and user dependent access permissions.

Based on the textual description and the use case based requirements analysis of the previous step we identify objects, relationships and operations required to construct the conceptual model of the system. We present three views of the conceptual model of the *Conference Review System:* the *Paper View*, the *User View* and *the Session View*. They are represented with UML class diagrams, which are shown in Figures 7 to Figure 9. We use views follows the only purpose of a more intuitive and clearer modeling.

Figure 7 depicts the *Paper View,* which includes the main classes Conference, Paper, Review and User. The conference class contains information, such as name of the conference and the deadlines of the conference. The status of the conference is a derived attribute (time dependent). Its value changes when submission deadline, review deadline or notification deadline are reached.

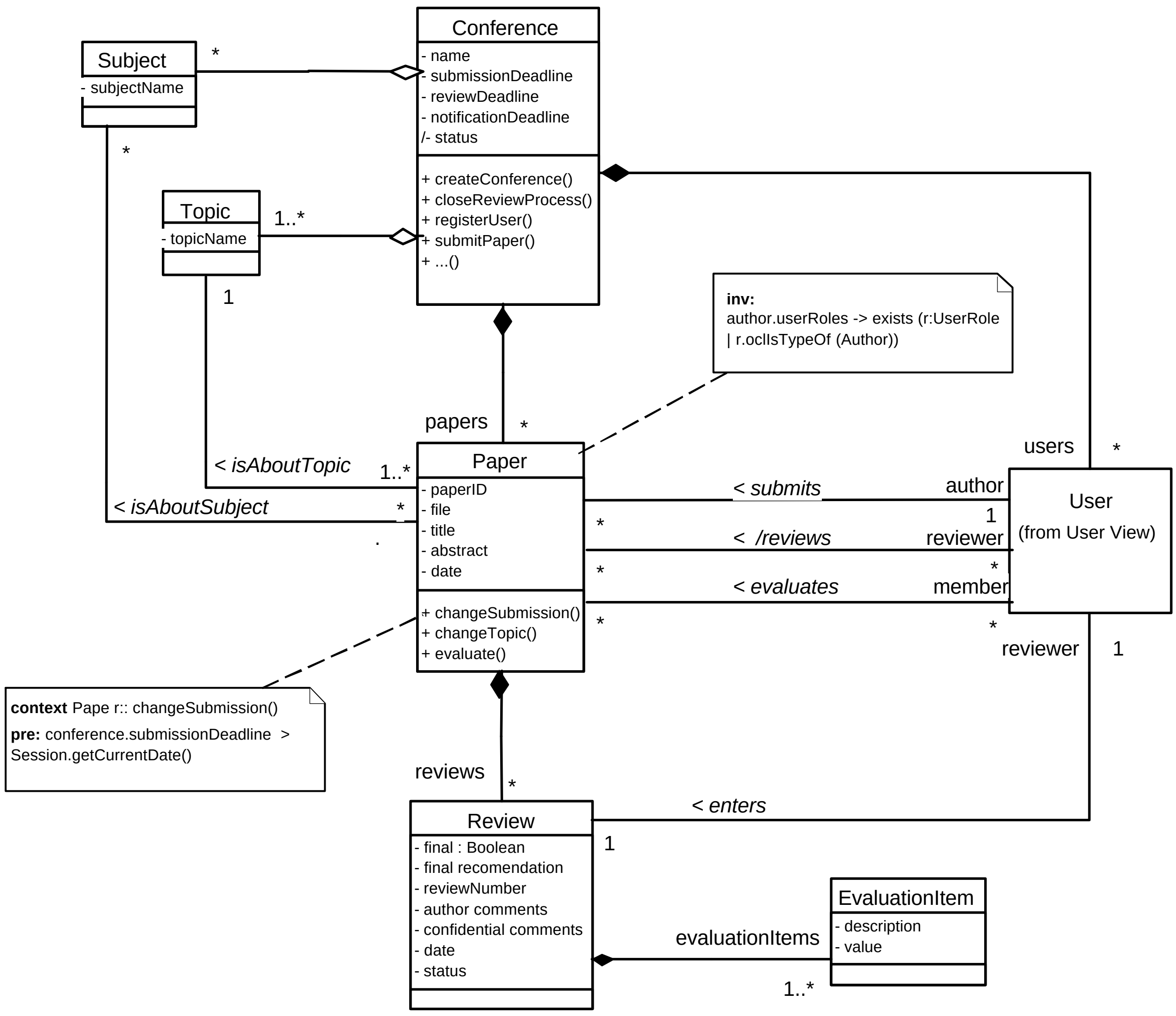

Figure 7: "Paper View" of the Conceptual Model

A paper is described by a *title*, an *abstract,* and a *submission date.* It has associated a file that contains the uploaded document and a paper ID generated by the system. For each paper at least three reviews will be provided by the reviewers. Each review is defined by a number (generated by system when the chair assigns the paper), a set of valued evaluation items, comments for the author and confidential comments, a review date and a status indicating whether this review is final or is still in progress. *Variants* in class *Paper* can be used to present or hide information about a paper in case of conflicts of interests between a member and a paper.

The OCL invariant added to the class *Paper* expresses that only the users in the role author can submit papers. In analogy, we can add other OCL invariants expressing rights for the roles chair, reviewer and member. As an example for pre-conditions/post-conditions for methods a pre-condition for *changeSubmission()* is included in Figure 7.

Figure 8 depicts the classes that represent the *User View* of the conceptual model. In a personalised (adaptive) Web application terminology this view is called the user model or user profile (Koch, 2000). The user is modelled by tracking her interest in conference subjects, expertise in conference topics and preferences on papers to review (in case of reviewers).

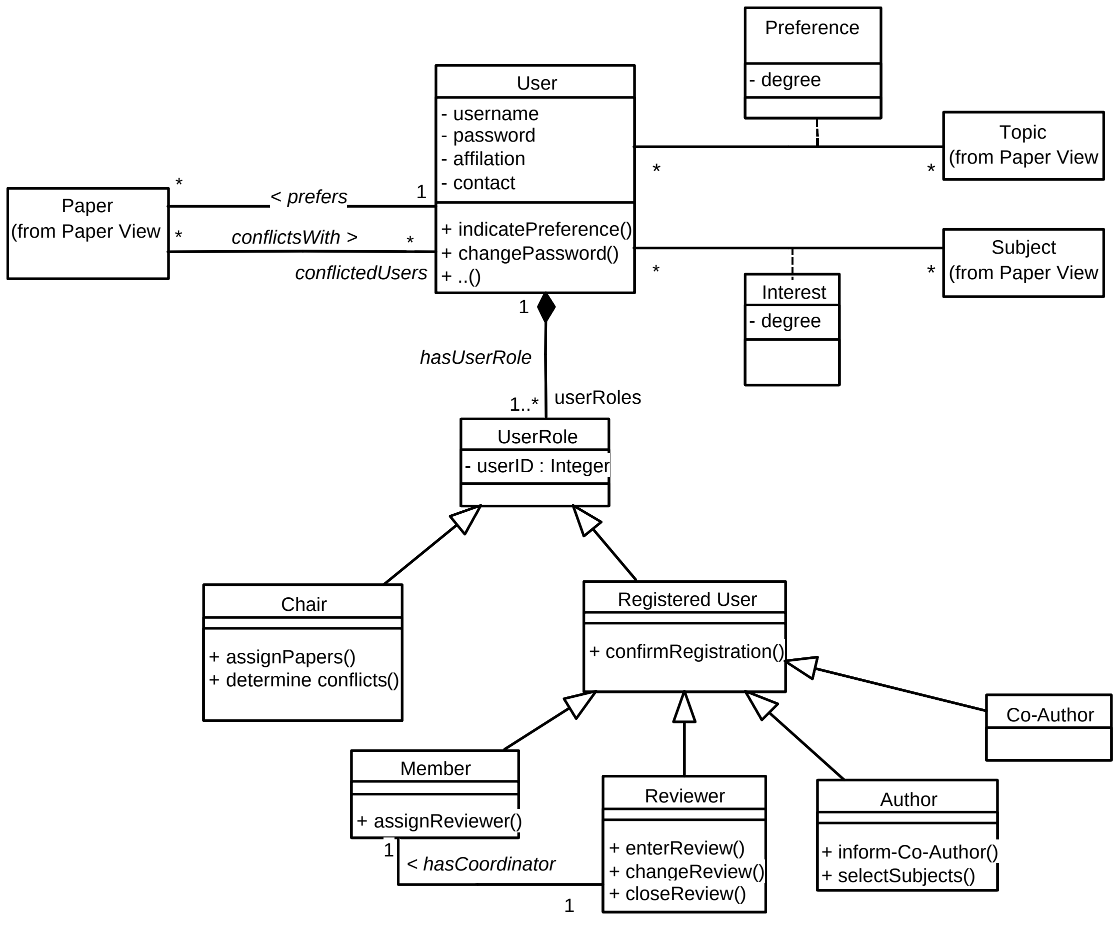

Figure 8: "User View" of the Conceptual Model

At a particular session a user of the system uses the system in one of the following roles: chair, member, reviewer or author, but he can be registered to the system for more than one role, e.g. author and member. The user role is modelled with the aggregated class *UserRole* to the class *User*. Constraints are added to some associations to show that these associations only are valid if the invariants expressed by the constraints are satisfied, e.g. a conference has only one chair.

The third view is the *Session View* that shows the associations between the conference, a current session and a current user of a session. This view models run-time information that is relevant for the conceptual model. The *Session View* is depicted in Figure 9.

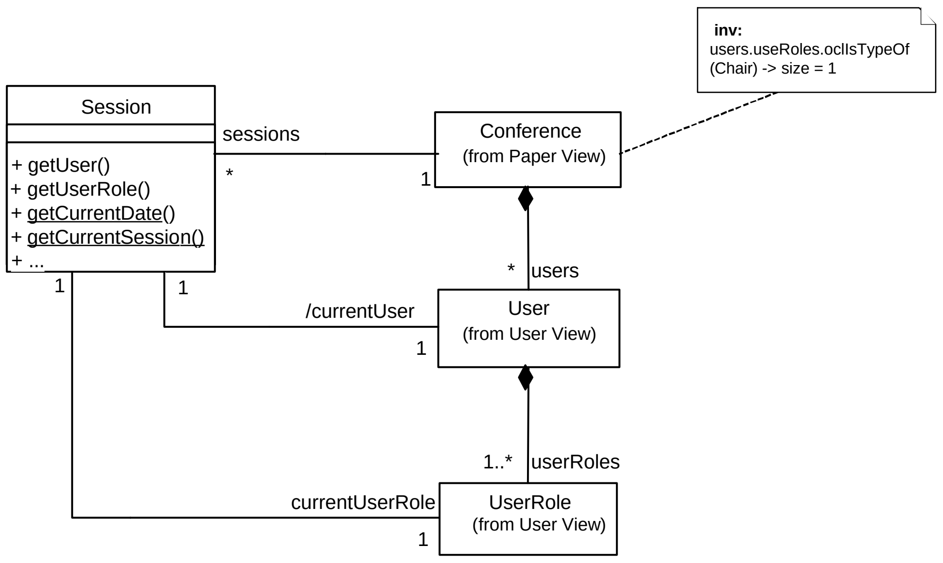

Figure 9: “Session View” of the Conceptual Model

4.3 Method

The developer can follow well-known object-oriented modeling techniques to construct the UML class model for the domain (see Figures 7 to Figure 9), such as:

1. Finding classes, such as *Conference, User, Paper,* and *Review.*
2. Specifying the most relevant attributes and operations, e.g. *title, abstract, paperID,* etc.
3. Determining associations between classes, such as *submits, enters, isAboutTopic,* etc.
4. Aggregating classes and identifying composition of classes, such as between *Conference* and *Topics* and between *Review* and *EvaluatingItems*.
5. Defining inheritance hierarchies, e.g. the user role hierarchy.
6. Defining constraints, such as the precondition for the operation *changeSubmission* of class *Paper*.

5 Systematic Building of a Navigation Space Model

Navigation design is a critical step in the design of a Web application. On the one hand, links improve navigability; on the other hand, however, they increase the risk of loss of orientation. Building a navigation model is not only helpful for the documentation of the application structure, it also allows a more structured increase in navigability. The navigation model comprise the navigation space model and the navigation structure model. The former specifies *which* objects can be visited by navigation through the Web application. *How* these objects are reached is defined by the navigation structure model. In this section we present the modeling elements used and the method applied to construct the navigation space model. The steps to follow for the construction of the navigation structure model and the UML stereotypes defined and used in this construction are described in the Section 6.

In the process of building the navigation space model the developer takes crucial design decisions, such as which view of the conceptual model is needed for the application and what navigation paths are required to ensure the application’s functionality. The designer’s decisions are based on the conceptual model and the application requirements defined in the use case mode.

A set of guidelines is proposed for modeling the navigation space. A detailed specification of associations, their multiplicity and role names, establish the base for a semi-automatic generation of the navigation structure model.

5.1 Modeling Elements

Two modeling elements are used for the construction of the navigation space model: navigation classes and navigation associations, which express direct navigability. They are the pendant to page (node) and link in the Web terminology.

- *navigation class*

  A navigation class models a class whose instances are visited by the user during navigation. Navigation classes will be given the same name as the corresponding conceptual classes. For their representation the UML stereotype «navigation class» is used (see Figure 10). Navigation classes may contain derived attributes. These attributes are derived from conceptual classes that are not included in the navigation model. The formula to compute the derived attribute can be given by an OCL expression. A derived attribute is denoted in UML by a slash (/) before its name.

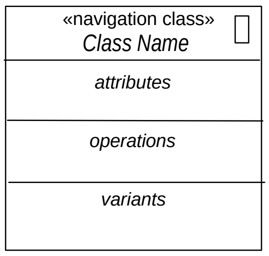

Figure 10: Navigational Class

- *direct navigability*

  Associations in the navigation space model are interpreted as representing direct navigability from the source navigation class to the target navigation class. Hence, their semantics are different from the associations used in the conceptual model. To determine the directions of the navigation the associations of this model are directed (possibly bi-directed). This is shown by an arrow that is attached to one or both ends of the association. Moreover, each directed end of an association is named with a role name and equipped with an explicit multiplicity. If no explicit role name is given, the following convention is used: if the multiplicity is less than or equal to one, the target class name is used as the role name; if the multiplicity is greater than one, the plural form of the target class name is used. In the UML diagrams shown in Figures 11 to Figure 15 all associations are implicitly assumed to be stereotyped by «direct navigability».

5.2 Sample Problem

Each actor, i.e. *Chair, Member, Reviewer* and *Author* has a different view of the navigation space. These views are represented as UML class models built with navigation classes and direct navigability stereotypes. OCL constraints are added for a precise description of classes and associations. Some of these constraints are shown in the diagrams within the UML note symbol that are attached to the corresponding modeling elements.

Starting from the conference review system homepage an author can only navigate to the papers submitted by himself. He can eventually change the submitted version or some data informed about the paper before submission deadline is reached. Figure 11 shows the navigation space of an author. For a more intuitive visual representation we suppress attributes and operations in the UML class diagrams shown in Figure 12 to Figure 15.

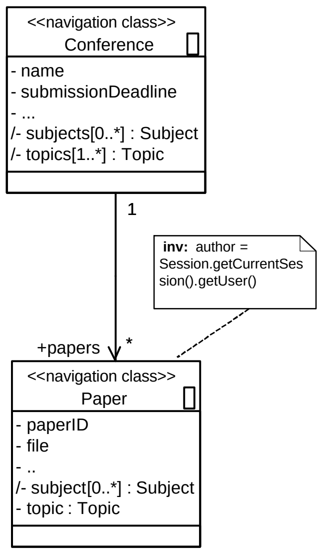

Figure 11: Navigation Space Model of an Author

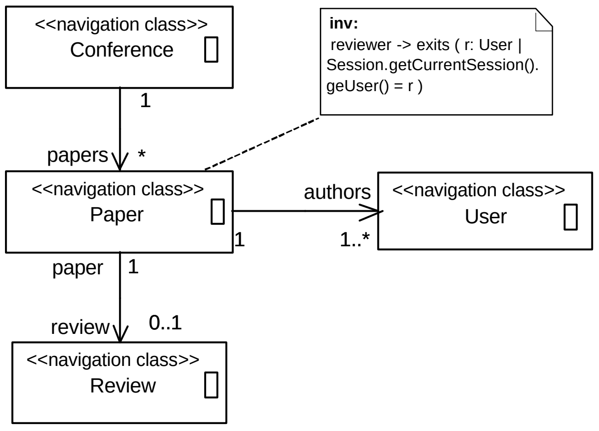

Figure 12: Navigation Space Model of a Reviewer

A reviewer can navigate through a space that includes the start page of *the Conference Review System* , the papers the he reviews and the review comments and evaluation items he already entered. The navigation space model of a reviewer is shown in Figure 12.

The UML class diagram for the navigation space of a program committee member (see Figure 13) includes the start page of the conference review, all papers and the reviews corresponding to papers assigned to him by the Chair under the pre-condition that he entered and marked as final his own review (if he is an assigned review of this paper). This constraint is included as a note in the navigation space model shown in Figure 13. After the review process is closed by the chair the members can access to the list of accepted and rejected papers.

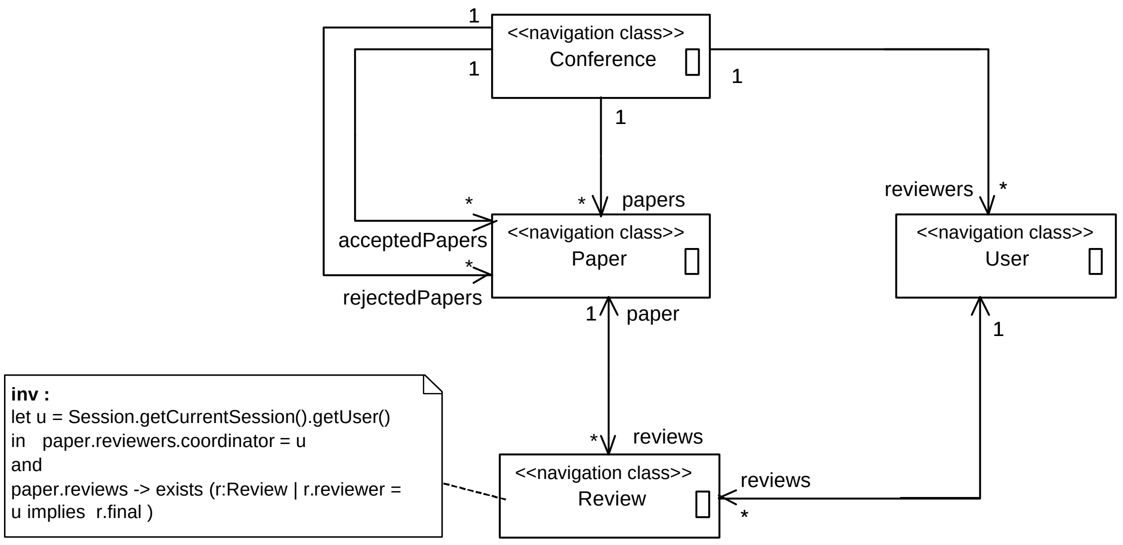

Figure 13: Navigation Space Model of a Member

The navigation space of the chair allows for administration of the conference information and the papers as it includes conference start page, all papers, reviews and the list of users of the system. The navigation space of the Chair is shown in Figure 14.

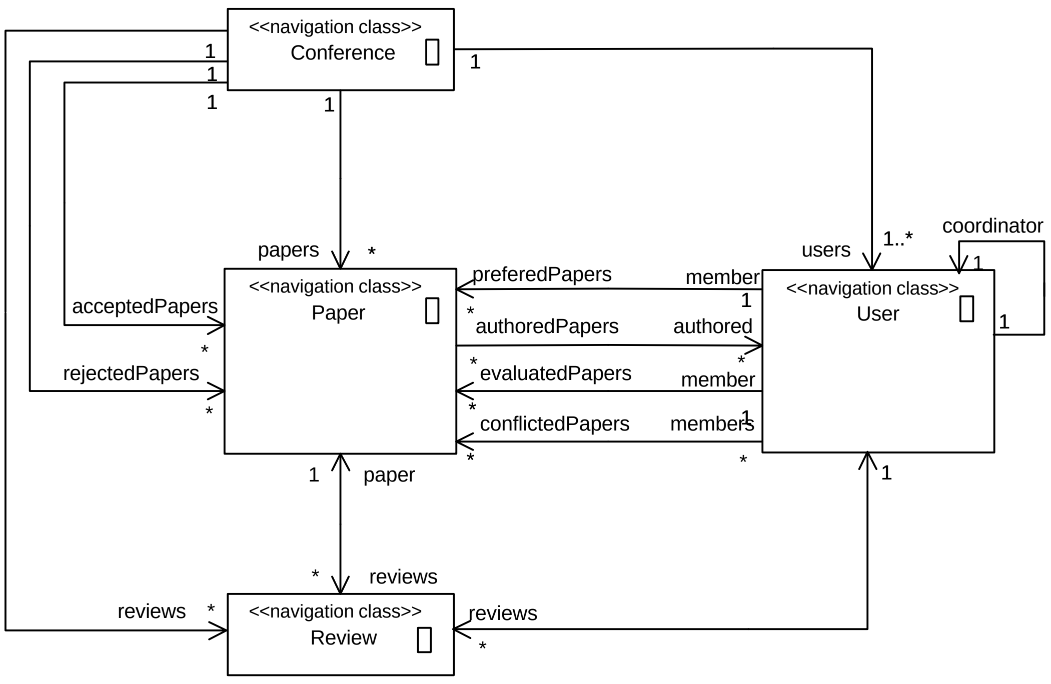

Figure 14: Navigation Space Model of the Chair

The UML class diagram shown in Figure 15 summarises the navigation space views of authors, reviewers, members and the chair in a global navigation space model for the *Conference Review System*.

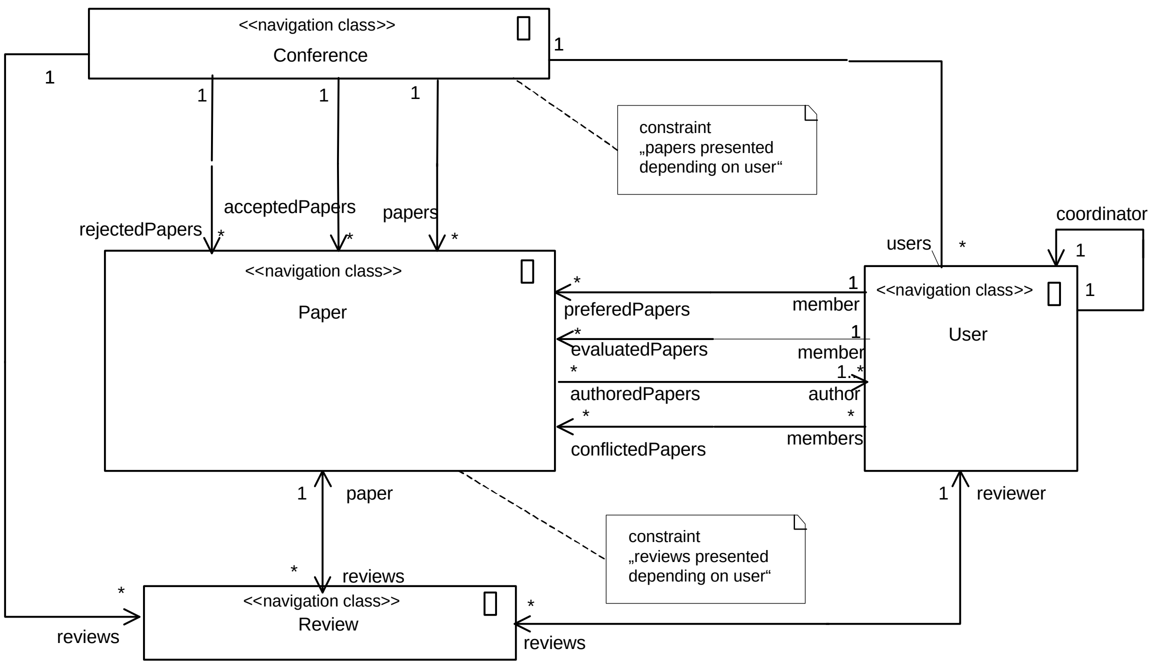

Figure 15: Navigation Space Model of the Conference Review System

The constraints associated to the navigation associations are only referenced by an invariant name in the diagram due to space problems. The complete specification of these two OCL constraints referenced in the diagram are listed below.

```
context Conference
inv  "papers presented depending on current user":
let ur = Session.getCurrentSession().getUserRole()
    in  let  u= Session.getCurrentSession().getUser()
        in  if ur.oclIsTypeOf(Author)
            then papers-> excludesAll (p:Paper | p.author <> u )
            else  if ur.oclIsTypeOf (Reviewer)
                   then papers  -> excludesAll (p:Paper | p.reviewer<> u )
                    else  true
                    endif
              endif
```

```
context Paper
inv  "reviews presented depending on current user":
let ur = Session.getCurrentSession().getUserRole()
    in  let  u= Session.getCurrentSession().getUser()
        in  if ur.oclIsTypeOf(Author)
            then reviews  -> isEmpty
            else  if ur.oclIsTypeOf (Reviewer)
                   then reviews ->excludesAll (r:Review | r.reviewer<>u )
                   else if ur.oclIsTypeOf (Member)
                        then  reviews -> excludesAll (r:Review | r.reviewer.coordinator<>u )
                                                  excludesAll (r:Review | r.paper.reviews
                                                  -> exists (r1:Review | r1.reviewer = u and  not r1.final)
                        else  true
                        endif
                   endif
             endif
```

The number of OCL constraints that can be added within notes to an UML diagram is limited. If the models require more than two or three constraints it is recommended to list them separately. In such a case, the model element that the constraint applies to has to explicitly be mentioned.

5.3 Method

The navigation space model that is built with the navigation classes and navigability associations are graphically represented by a UML class diagram as shown in Figures 11 to Figure 15. Although there is obviously no way to automate the construction of the navigation space model, there are several guidelines that can be followed by the developer:

1. Include classes of the conceptual model that are relevant for the navigation as navigation classes in the navigation space model (i.e. navigation classes can be mapped to conceptual classes). If a conceptual class is not a visiting target in the use case model, it is irrelevant in the navigation process and therefore omitted in the navigation space model, such as the classes *Topic, Subject* and *UserRole* in this example.

2. Keep information of the omitted classes (if required) as attributes of other classes in the navigation space model (e.g. the newly introduced attributes *evaluationItems* of the navigation class *Review*). All other attributes of navigation classes map directly to attributes of the corresponding conceptual class. Conversely, exclude attributes of the conceptual classes that are considered to be irrelevant for the presentation in the navigation space model, e.g. user password.
3. Associations of the conceptual model are kept in the navigation model. Additional associations can be added for direct navigation to avoid navigation paths of length greater than one. Examples are the newly introduced navigation associations between *Conference* and *User* (see Figure 14).
4. Add additional associations based on the requirements description or the scenarios described by the use case model. Hence associations for *accepted papers* and *rejected* papers (see Figure 15).
5. Add constraints to specify restrictions in the navigation space, e.g. invariants for classes conference and paper.

6 Generation of a UML Navigation Structure Model

The navigation structure model describes how the navigation is supported by access elements such as indexes, guided tours, queries and menus. Technically, the navigation paths together with the access elements are presented by a class model which can be systematically constructed from the navigation space model in two steps: The first step consists in enhancing the navigation space model by indexes, guided tours and queries. The second one consists in deriving menus directly from the enhanced model. Menus represent possible choices for navigation. The result is a UML class diagram built with UML stereotypes, which are defined according to the extension mechanism of the UML.

6.1 Including Access Primitives

Access primitives are additional navigation nodes required to access navigation objects. The following access primitives are defined as UML stereotypes: index, guided tour, query and menu. In this section the first three are described and used to refine the navigation space model. Menu is treated separately in the next subsection.

6.1.1 Modeling Elements

The following modeling elements are used for describing indexes, guided tours and queries. Their stereotypes and associated icons are defined in (Koch & Mandel, 1999); some of the icons stem from Isakowitz, Stohr and Balasubramanian (1995).

- *index*

  An index allows direct access to instances of a navigation class. This is modelled by a composite object, which contains an arbitrary number of index items. Each index item is in turn an object, which has a name that identifies the instance and

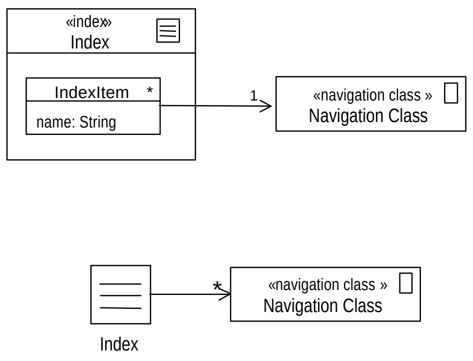

Figure 16: Index Class and Shorthand Notation for Index Class

owns a link to an instance of a navigation class. Any index is a member of some index class, which is stereotyped by «index» with a corresponding icon. An index class must be built conform to the composition structure of classes shown in Figure 16 (above). Hence the stereotype «index» is a restrictive stereotype in the sense of Berner, Glinz and Joos (1999). In practice, we will always use the shorthand notation shown in Figure 16 (below).

Note that in the short form the association between *Index* and *NavigationClass* is derived from the index composition and the association between *IndexItem* and *NavigationClass*.

- *guided tour*

A guided tour provides sequential access to instances of a navigation class. For classes, which contain guided tour objects we use the stereotype «guidedTour» and its corresponding icon depicted in Figure 17 (above). Any guided tour class must be built conform to the composition structure of classes shown in Figure 17 (above). Each *NextItem* must be connected to a navigation class. Guided tours may be controlled by the user or by the system. Figure 17 (below) shows the shorthand notation for a guided tour class.

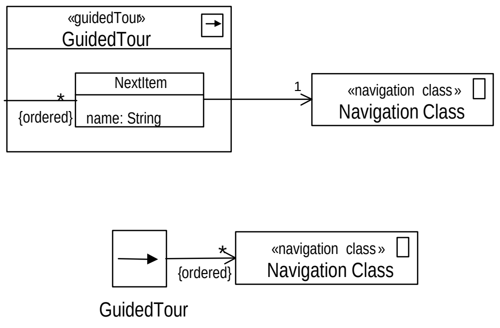

Figure 17: Guided Tour Class and Shorthand for Guided Tour Class

- *query*

A query is modeled by a class which has a query string as an attribute. This string may be given, for instance, by an OCL select operation. For query classes we use the stereotype «query» and the icon depicted in Figure 18 (above). As shown in Figure 18 (above), any query class is the source of two directed associations related by the constraint {xor}. In this way a query with several result objects is modelled to lead first to an index supporting the selection of a particular instance of a navigation class. The query results can alternatively be used as input for a guided tour.

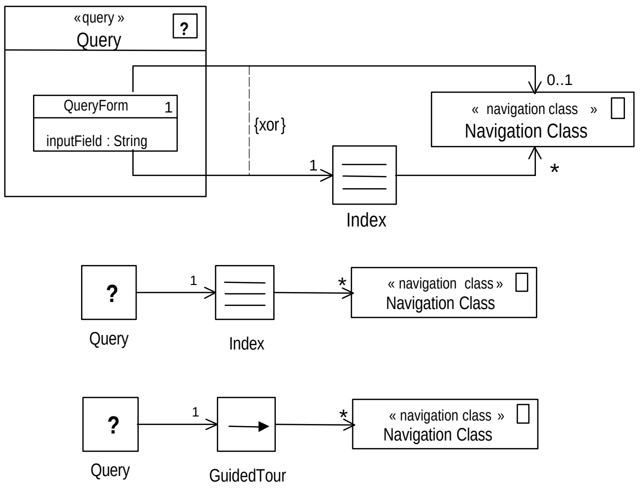

Figure 18: Query Class and Shorthand for Query

Figure 18 (below) also shows the shorthand notation for a query class in combination with an index class or with a guided tour.

6.1.2 Sample Problem

Figure 19 shows how the navigation space model of the can be enhanced by indexes, guided tours and queries.

Note that the access to all papers submitted to the *Conference Review System* is modelled with two indexes and two search items: search by title and by subject, and lists of indexes by paper ID and by topics.

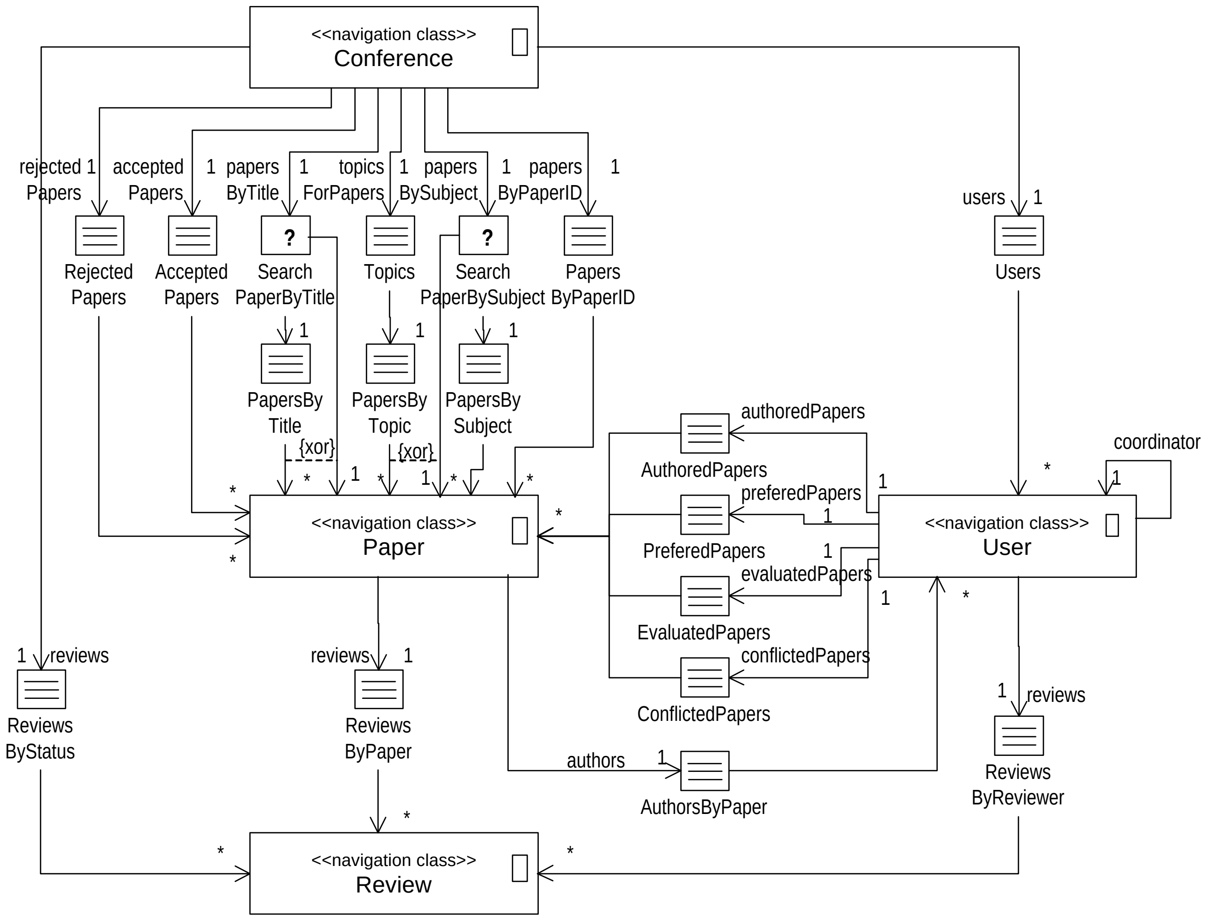

Figure 19: Navigation Structure Model (First Step) of the Conference Review System

6.1.3 Method

The navigation space model is enhanced by access elements of type index, guided tour and query following certain rules, which can be summarised as follows:

1. Replace all bi-directional associations, which have multiplicity greater than one at both associations ends by two corresponding unidirectional associations.
2. Replace all bi-directional associations, which have multiplicity greater than one at one association end with one unidirectional association with an directed association end at the end with multiplicity greater than one. The navigation in the other direction is guaranteed by the use of navigation trees introduced later in the design.
3. Consider only those associations of the navigation space model, which have multiplicity greater than one at the directed association end.

4. For each association of this kind, choose one or more access elements to realise the navigation. In our example, e.g. for the association with role *papers* two queries and two indexes are selected. The query allows for the search of papers by subject and by title. One index offers a list of topics, where each topic is a link to papers of that topic. The other index is a list of the papers ordered by paper ID.
5. Enhance the navigation space model correspondingly. Role names of navigation in the navigation space model are now moved to the access elements (compare Figure 15 and Figure 19). If two or more alternatives are introduced in step 3, distinguish them by changing the role names of the associations by the search or index criteria used.

In item 2 the task of the designer is to choose appropriate access elements. However, note that it is also possible to fully automate this step by making the choice of an index a default design according to an attribute with the property {key} of the target navigation class.

6.2 Adding Menus

In this step, access primitives of type menu are added to the navigation structure model.

6.2.1 Modeling Elements

The modeling element *menu* is an additional access primitive that can be added to the list presented in the previous step. Its UML stereotype is defined by Koch and Mandel (1999).

- *menu*

  A *menu* is an index of a set of heterogeneous elements, such as an index, a guided tour, a query, an instance of a navigation class or another menu. This is modelled by a composite object which contains a fixed number of menu items. Each menu item has a constant name and owns a link either to an instance of a navigation class or to an access element. Any menu is an instance of some menu class which is stereotyped by «menu» with a corresponding icon.

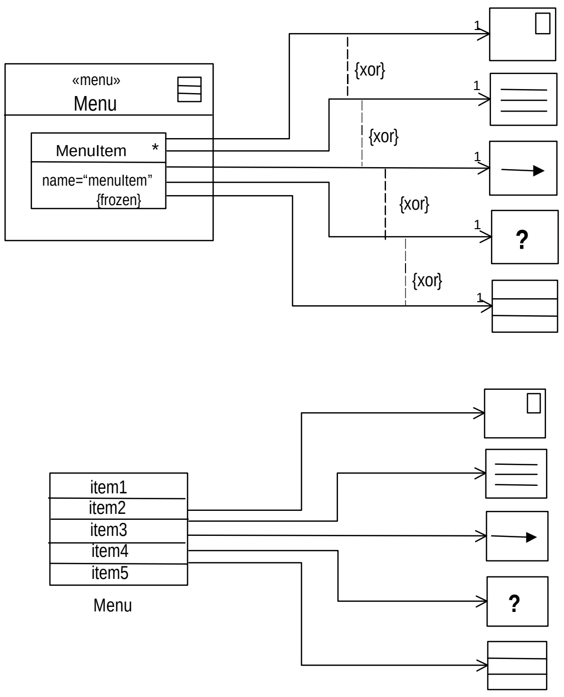

Figure 20: Menu Class and Shorthand for Menu Class

  A menu class must be built in accordance with the composition structure of classes shown in Figure 20 (above). Hence the stereotype «menu» is again a restrictive stereotype according to the classification of stereotypes given by Berner, Glinz and Joos (1999). The property {frozen} is attached to each name attribute in a menu item class to indicate that menu items have fixed names. Nevertheless, the same menu item class may have different instances since there may be menu items with the same name but linked to different objects.

For a convenient notation of menu classes in navigation structure models the shorthand notation shown in Figure 20 (below) is used. This is a somewhat more flexible extension than the extension mechanisms of UML allows, since it includes a variable number of compartments with the names of the menu items. To remain hundred percent UML compatible, we have to use the notation shown in Figure 19 (above) with the disadvantage of being space consuming. This shorthand is the only not strict UML notation used in this work.

6.2.2 Sample Problem

Figure 21 shows how the navigation structure model of the previous subsection of the *Conference Review System is* enriches by menus. Each menu class is associated with a composition association to a navigation class. Note that the role names occurring in the previous model (Figure 20) are now names of corresponding menu items.

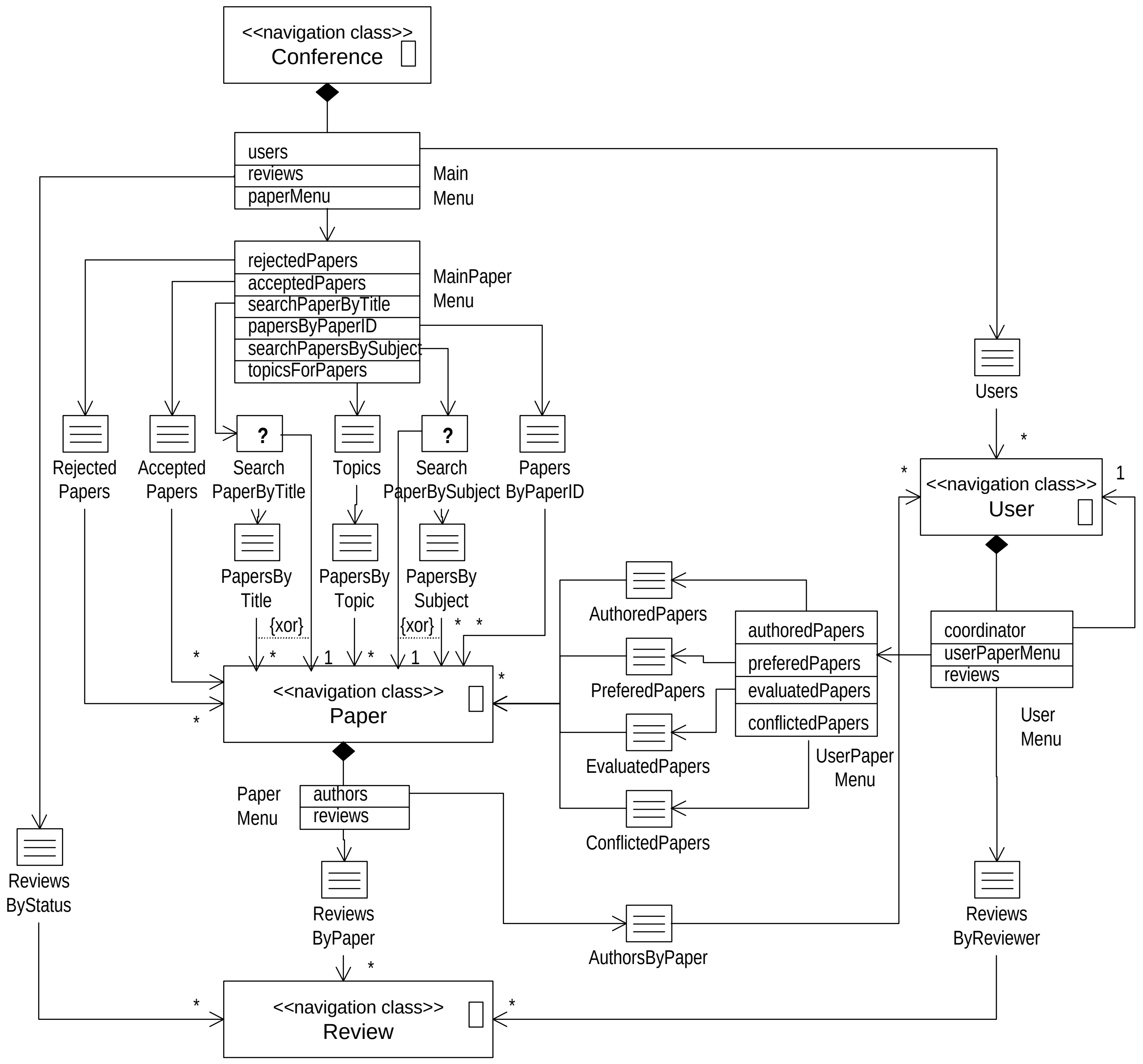

Figure 21: Navigation Structure Model (Second Step) of the Conference Review System

6.2.3 Method

The navigation space model is enhanced by access elements of type menu following certain rules which can be summarised as follows:

1. Consider those associations, which have as their source a navigation class.
2. Associate to each navigation class, which has (in the previous model) at least one outgoing association, a corresponding menu class. The association between a navigation class and its corresponding menu class is a composition. In our example the navigation classes that require a menu are *Conference, Paper* and *User*. We introduce a *MainMenu*, a *PaperMenu*, and *UserMenu.*
3. Reorganise a menu in a menu with sub menus. In our example this is done for the MainMenu, where the MainMenu item paper is replaced by the sub menu MainPaper menu.
4. Introduce for each role, which occurs in the previous model at the end of a directed association a corresponding menu item, such as *acceptedPapers, searchPaperBySubject* and *preferredPapers*. By default, the role name is used as the constant name of the menu item.
5. Any association of the previous model which has as its source a navigation class now becomes an association of the corresponding menu item introduced in step 4.

Note that all the steps in the above method can be performed in a fully automatic way. As a result we obtain a comprehensive navigation structure model of the application. The method guarantees that this model conforms to the pattern shown in Figure 22.

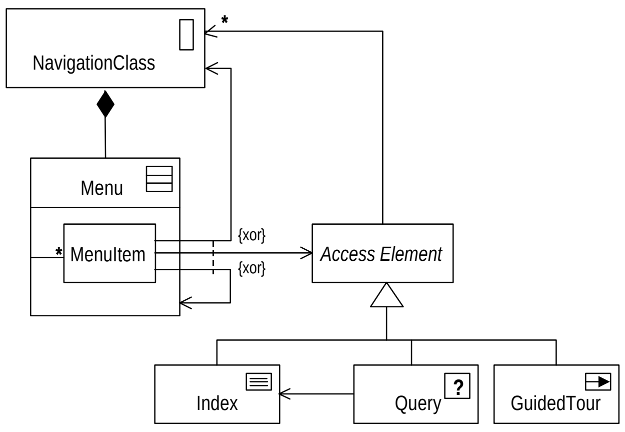

Figure 22: Pattern for Access Structures

7 Using UML for Storyboarding and Presentation Flow

Presentation design supports the construction of a presentation model based on the navigation structure model and additional information collected during the requirements analysis. The presentation model consists of a set of views that show the content and the structure of the single nodes (i.e. how each node is presented to a user) and how the user can interact with them. We propose the construction of sketches, storyboards and a presentation flow model.

First, the Web designer proposes a sketch of each main user interface view, i.e. the design of abstract user interfaces. These are rough drawings of a couple of relevant elements of each navigation node. This sketching technique is frequently used by Web designers, but without having a precise notation for it as described by Sano (1996). We propose to use an appropriate extension of UML for this purpose. These sketches are used for the storyboard model that consists of storyboarding scenarios.

In the second step based on the storyboard model the designer can decide whether he wants to choose a multiple-window technique and/or whether he wants to use frames. The objective of the presentation flow model is to show *where* the user interface views of the storyboard model are presented to the user, i.e. in which frame or window they are displayed. It also shows which contents are replaced when the user interacts with the system.

7.1 Storyboarding

The storyboarding design may be considered as an optional step as the design decisions related to the user interface can also be taken during the realisation (prototyping) of the user interface. Sketches give a first look and feel of the user interface. After having produced the different user interface views (sketches) storyboarding scenarios can be developed which show sequences of user interface views in the order in which a user can navigate from one view to another (Pinheiro da Silva & Paton, 2000). The objective is to visualise the organisation of the Web application structure in a more intuitive manner than the navigation structure model does.

Both, the sketches of user interface views as well as the storyboarding scenarios are a useful means for the communication between a customer and a Web designer. In particular, they can be validated w.r.t. the use cases identified during the analysis phase. We model sketches with UML classes and UML associations of type composition. For the visualisation we choose the nested composition graphical representation, which is an alternative offered by UML to the composition drawn with the black diamond symbol.

7.1.1 Modeling Elements

For the construction of the sketches we propose a set of modeling elements, some of which are shown in Figure 23. As for the navigation elements in Sections 4 and Section 5, each class in Figure 23 defines a stereotype which will be used in concrete sketches. The associations and the inheritance relation show again the well-formedness rules, the notation of the interface elements in form of icons stems mainly from Baumeister, Koch and Mandel (1999).

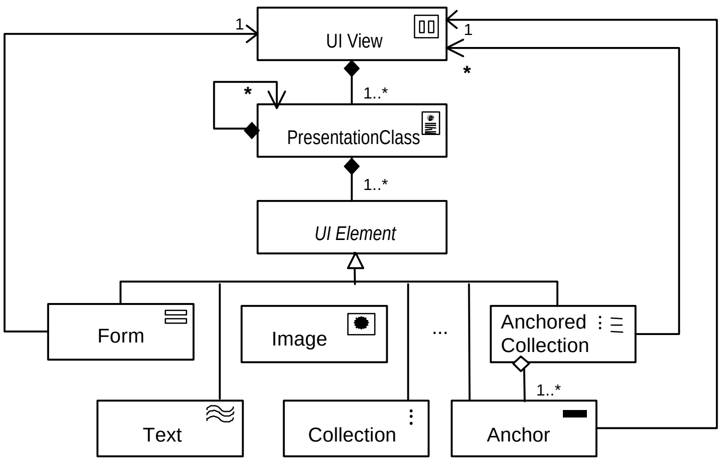

Fig. 23: Metamodel for the Abstract User Interface Elements

- *user interface view*

  A user interface (UI) view specifies that each instance of this class is a container of all the abstract user interface elements which are presented simultaneously (i.e. at one moment in time in one window) to the user. For user interface view classes we use the stereotype «UI view» and its corresponding icon depicted in Figure 23.

- *presentation class*

  A presentation class is a structural unit which allows to partition a user interface view into groups of user interface elements. For presentation classes we use the stereotype «presentation class» and its corresponding icon depicted in Figure 23.

- *user interface element*

  A user interface element is an abstract class which has several specialisations describing particular interface elements (Figure 23).

For instance, the stereotyped classes «text», «image», «video», «audio», «anchor», and «form», are subclasses of UI elements for modeling texts, images etc.. The classes «collection», and «anchored collection» are also subclasses of user interface element which provide a convenient representation of frequently used composites. Anchor and form are the basic interactive elements. An anchor is always associated with a link for navigation. Through a form a user interacts with the Web application supplying information and triggering a submission event.

7.1.2 Sample problem

Figure 24 show a UI view of the *Conference Review System*. It is the composite of the presentation of a paper (showing information about a paper, such as paperID, title, abstract) and a navigation tree build on the basis of the main menu and the paper menu (which in turn is

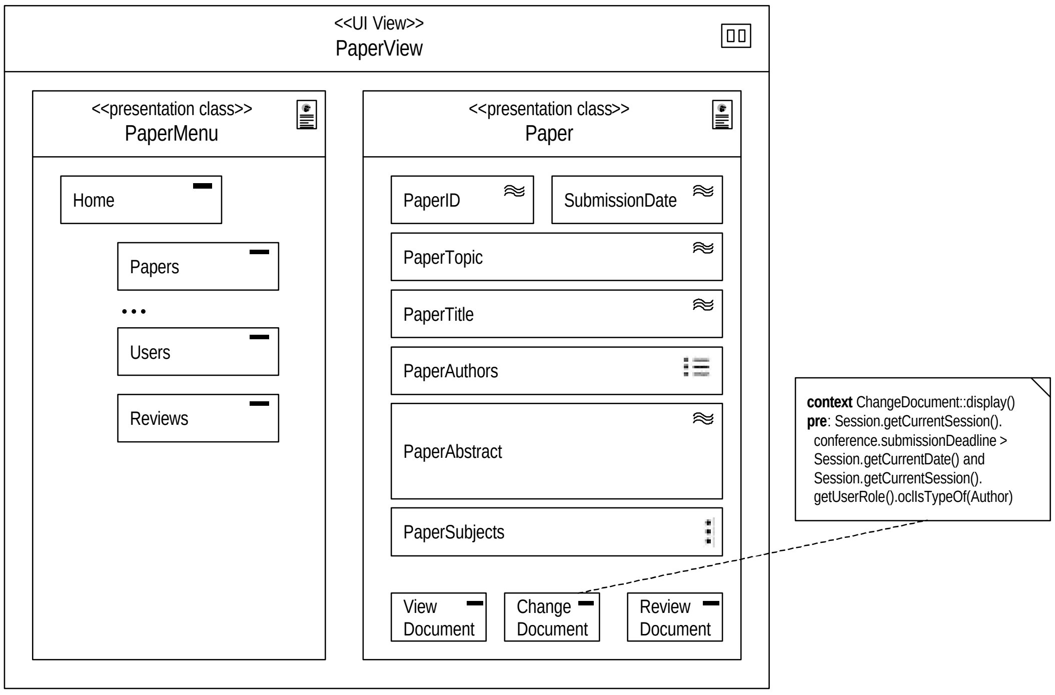

Figure 24: User Interface View of a Paper

composed by a set of anchors).

Figure 25 shows a storyboard scenario which shows how a chair can find information about papers and authors of papers.

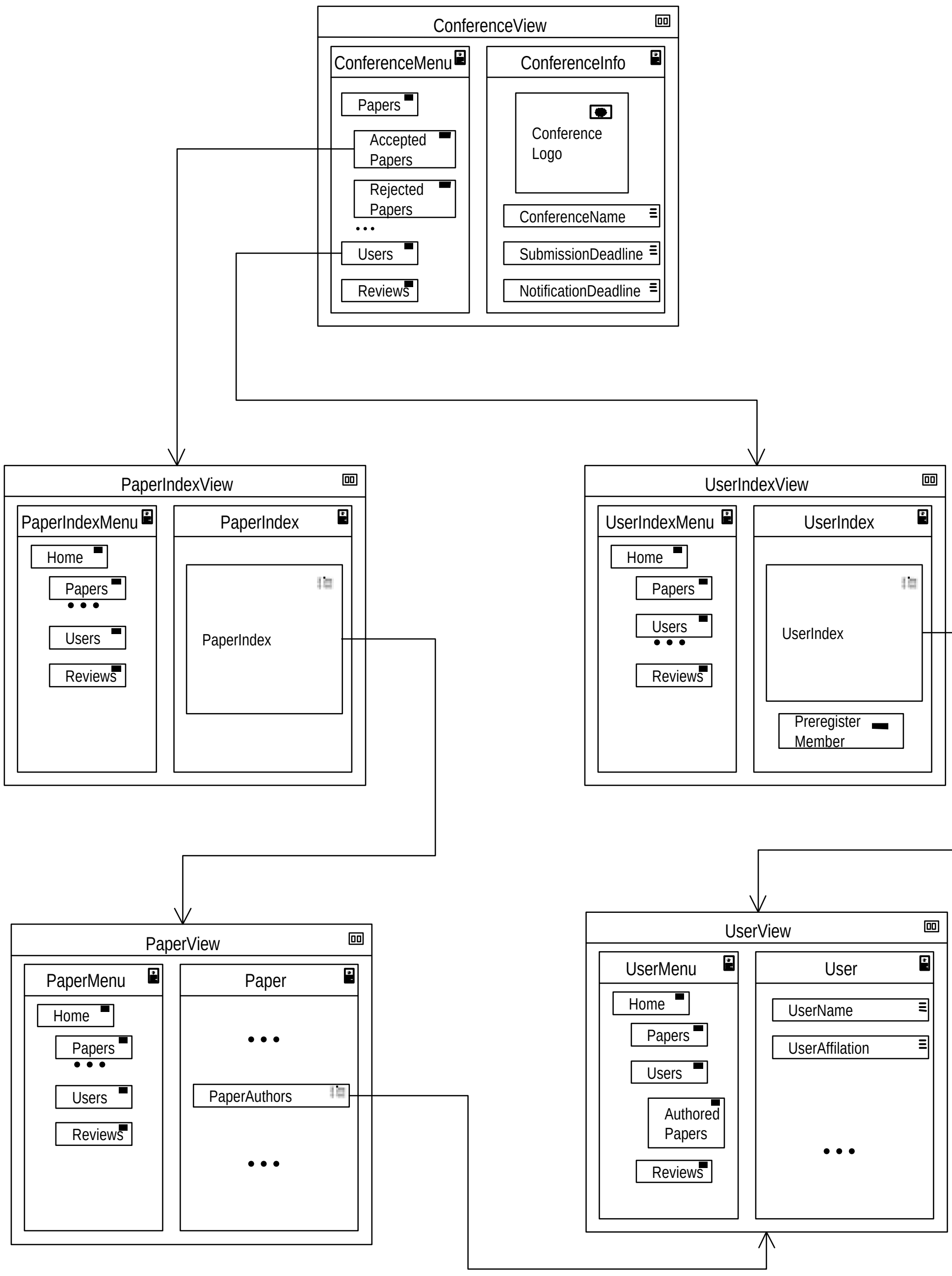

Fig. 25: Example of Storyboard Scenario for the Conference Review System

7.1.3 The Method

To design a storyboard model we start with a first navigation model of the Web application. Each abstract user interface is represented as a composition of classes. The following rules can be used as guidelines for the construction of the storyboard model (UML class diagram) based on the user interface views:

1. Construct a presentation class for each navigation class occurring in the navigation structure model. The presentation class defines a template suited to present the instances of the class by taking into account the given attributes. Stereotyped interface elements, such as «text», «image», «audio», «video» are used for attributes of primitive types and «collections» are used for lists, etc. Figure 24 depicts the presentation class for a paper.
2. Construct a presentation class for each menu and index occurring in the navigation structure model. The presentation of a menu or an index class usually consists of a list of anchors. Use stereotypes «anchor» or «anchored collection» for this purpose. An example for the presentation of a menu is the in Figure 24.
3. Construct a presentation class for each query and guided tour. For queries use a «form» stereotype and for guided tours use a menu with items "next" and "previous" (allow to navigate to the next and to the previous object within a guided tour).
4. Construct presentation classes for navigation support as composition of the presentation classes derived from the access structures. They are used to reflect the navigation path. It is the designer's decision where to include these derived presentation classes (see Figure 24).
5. Add anchors to the presentation classes to allow creation, destruction and execution of operations on objects of the conceptual model – for example the submission of a paper by an author, the upload function of a review (Figure 24) or when the chair pre-registers a member. The functional requirement of these anchors stem from the use case model.
6. Determine which presentation elements should be presented together to the user (in one window). The corresponding presentation classes must be composed in a user interface view (stereotyped by «UI view»). Since the user needs always a combination of conceptual data and navigation facilities, typically a user interface view consists of the presentation class constructed for a navigation class and of a presentation class constructed for navigation facilities.
7. Add OCL constraints, if needed, as shown in Figure 24 for the *ChangeDocument* anchor.
8. Construct storyboarding scenarios represented by sequences of user interface views (optional). For this purpose introduce links that connect an anchor (within a UI view) with another UI view thus showing the possible flows of presentations that can be caused by user interactions. An example for a storyboard model is the scenario shown in Figure 25.

7.2 Building the Presentation Flow

The focus of this step is to model the dynamics of the presentation showing *where* the navigation objects and access elements will be presented to the user, i.e. in which frame or window the content is displayed and which content will be replaced when a link is activated. First of all, the designer has to specify whether a single or multiple-window technique is used, whether frames are used and, if so, into how many frames framesets are divided. In the case of one window without frames the result is obvious from the storyboard model and no further graphical representation is needed. Each click produces just a complete replace of the window content by a new content. The presentation flow is visualised with UML interactions models, i.e. UML sequence diagrams.

7.2.1 Modeling Elements

A presentation flow model of a Web application is built with stereotyped classes «window», «frameset» and «frame». We use these stereotypes to indicate the location of the presentation. The user interface elements defined in Section 6 are used as well in the messages sent in an UML interaction model.

- *Window*

  A window is the area of the user interface where presentation objects are displayed. A window can be moved, maximised/minimised, resized, reduced to an icon and/or closed. For performing these actions a window contains special buttons. In addition, windows include two scrollbars: a horizontal and a vertical scrollbar that allow visualisation of the whole content of the window. Any window is an instance of a class stereotyped by «window» with a corresponding icon (see Figure 26).

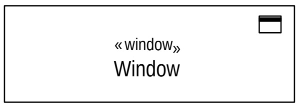

Figure 26: Window

- *Frameset*

  A frameset is a modeling element used to define multiple visualisation areas within a window. It is divided into lower level location elements – so called frames – and may also contain an arbitrary number of nested framesets. A frameset is an instance of a frameset class stereotyped by «frameset» with a corresponding icon (see Figure 27).

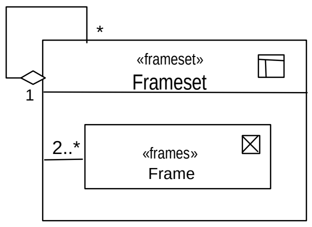

Figure 27: Frameset and Frame

- *Frame*

  A frame is always part of a frameset, it defines an area of the corresponding frameset where content is displayed. A frame is an instance of a frame class stereotyped by «frame» with a corresponding icon (see Figure 27).

7.2.2 Sample Problem

Figure 28 shows the selected windows and the structure of the frameset. This representation gains in relevance when the Web application has several windows and many different framesets.

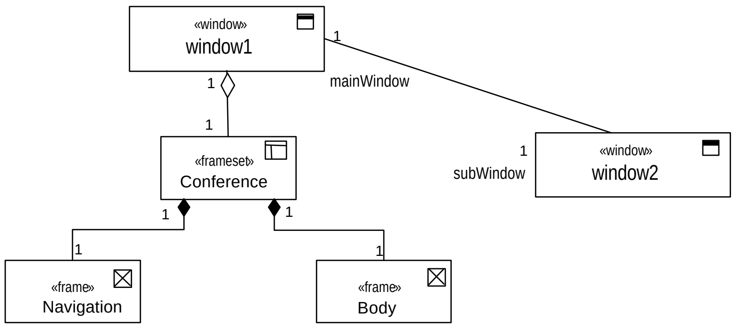

Figure 28: User Interface Location Elements for the Conference Review System

Figure 29 shows a presentation flow representing a scenario for a sequence of possible navigation activities that can be performed by the chair of a conference.

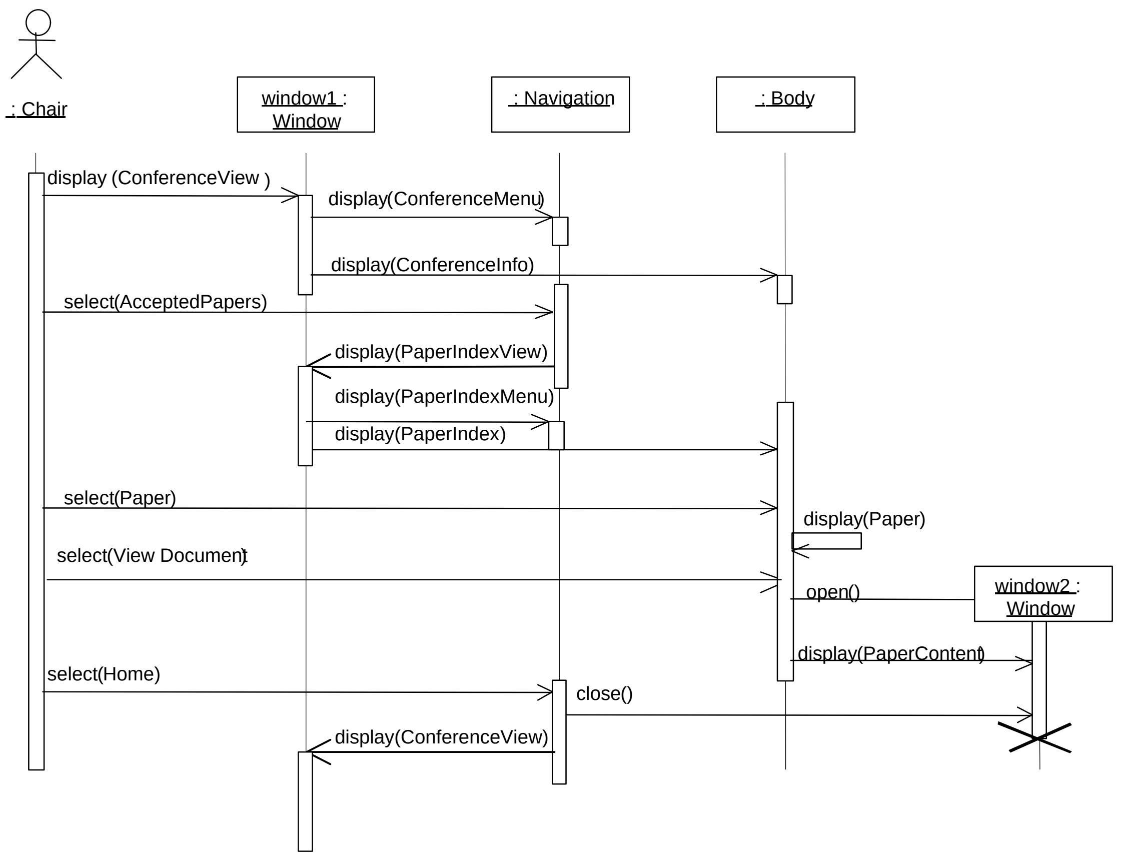

Figure 29: View of the Presentation Flow of the Conference Review System

7.2.3 The Method

The presentation model requires that the designer takes some decisions, such as number of windows to be used and number of frames each frameset is divided into. Hence the construction of the presentation structure cannot be fully automated, but there are certain guidelines that the designer can follow:

1. Choose between a single or multiple-window technique. In case of a multiple-window technique plan how many windows will be used.
2. Choose the frame style, i.e. with or without frames. In the first case specify how many frames each frameset has.
3. Represent the presentation structure with a UML class diagram (optional).
4. Set the scenario for the interaction model, i.e. define which navigation path of the navigation structure diagram will be modelled. A navigation path is always related to a use case.
5. Represent the user, the windows and/or frame objects in the horizontal dimension.

6. Specify a *display* message for each presentation object that should be presented to the user (in a window or frame). The parameter of the display message is the corresponding presentation object (described in previous sections).
7. Include a *select* message for each user action which selects an anchor or a button. The anchor or button names are the parameters of the message.
8. Specify a *fill* and a *submit* message for each user action, which consist of supplying data in a query form. This form is the parameter of the message.
9. Include a message for each open and each close of a window.
10. Use "balking" to specify the period of time that a window or frame is active.

UML sequence diagrams are used to represent the presentation control flow. Note that this representation does not include additional classes needed in the implementation, such as client pages and server pages (Conallen, 1999). These more implementation-oriented modeling elements can be introduced in further refinements.

8 Conclusions and Future Work

In this paper we describe the authoring process of the UML-based Web Engineering (UWE) approach using the running example of a *Conference Review System* (Schwabe, 2001). This approach focuses on a systematic construction of the design models for a Web application using exclusively UML notation and techniques. In particular, we use stereotypes of our UML profile for Web applications, which is constructed according to the extension mechanisms defined by the UML.

Our methodology is built on many concepts, modeling elements and steps defined in other methods for hypermedia and Web design, which proved to be useful to support Web development. A comparison of hypermedia design methods is presented in (Koch, 1999).

What does distinguish our approach from other methods? Our methodology adds precision to the notation, to the models and to the process. Precision in the notation is obtained by the use of UML. Even more precision in the models is obtained by the use of OCL constraints applied to the modeling elements used in the diagrams. Precision in the process is obtained by detailed guidelines that support the construction of the models.

What does a systematic authoring process help for? Some steps can be performed in a semi-automatic way, thus providing the basis for a generation mechanism in Web development.

An important next step is to extend existing Case Tools, such as ArgoUML (2001), with the set of stereotypes of our Web Profile to allow visualisation of the stereotypes through the corresponding icons. In addition, a Case Tool supporting the development of Web applications based on our UWE authoring process is planned. It will consist of tools for the semi-automatic transition between the different conceptual models and a mapping to specific implementations like HTML or JSPs. Further work will be done on modeling concerning the dynamical behaviour on client and server side.

References

ArgoUML (2001). http://www.tigris.org

Baumeister H., Koch N.& Mandel L. (1999). Towards a UML Extension for Hypermedia Design. *Proceedings of The Unified Modeling Language Conference: Beyond the Standard (UML´99).* France R. and Rumpe B. (Eds). LNCS 1723, Springer Verlag, 614-629.

Berner S., Glinz M. & Joos S. (1999). A Classification of Stereotypes for Object-oriented Modeling Languages. *In Proceedings UML'99 – The Unified Modeling Language: Beyond the standard Conference*. France R. and Rumpe B.(Eds.).LNCS 1723. Springer Verlag, 249-264.

Conallen J. (1999). *Building Web Applications with UML*. Addison-Wesley.

De Troyer O. & Leune C. (1997). WSDM: A User-centered Design Method for Web Sites. *Proceedings of the 7th International World Wide Web Conference.*

Hennicker R. & Koch N. (2000). A UML-based Methodology for Hypermedia Design. *Proceedings of the Unified Modeling Language Conference, UML´2000,* Evans A. and Kent S. (Eds.). LNCS 1939, Springer Verlag, 410-424.

IEEE (1991). Standard Glossary of Software Engineering Terminology. Springer Edition

Isakowitz T., Stohr E. & Balasubramanian P. (1995). A Methodology for the Design of Structured Hypermedia Applications. *Communications of the ACM,* 8(38), 34-44.

Jacobson I., Booch G., & Rumbaugh J. (1999). *The Unified Software Development Process.* Addison Wesley.

Koch N. (1999). A Comparative Study of Methods for Hypermedia Development. Technical Report 9905, *Institute of Computer Science, Ludwig-Maximilians-University Munich*.

Koch N. (2000). Software Engineering for Adaptive Hypermedia Applications. PhD. Thesis.

Koch N. & Mandel L. (1999) Using UML to Design Hypermedia Applications. *Ludwig-Maximilians-University Munich, Institute of Computer Science,* Technical Report 9901.

Kruchten P. (1999). *The Rational Unified Process: An Introduction.* Addison Wesley.

Lowe D. & Hall W. (1999). *Hypermedia & the Web: An Engineering Approach.* John Wiley & Sons.

Pinheiro da Silva P., Paton N.: UML*i:* The Unified Modeling Language for Interactive Applications. In Proceedings «UML» 2000, Evans, A., Kent, S. (Eds), LNCS, Vol. 1939. Springer-Verlag (2000) 117-132.

Preece J., Rogers H., Benyon D., Holland S., Carey T.: Human-Computer Interaction. Addison Wesley (1994).

Rossi G., Schwabe D., & Lyardet F. (2000). Web Applications Models are More than Conceptual Models. *Web Engineering Workshop at WWWCM´99.*

Sano D.: Lare-Scale Web Sites – A Visual Design Methodology. Wiley Computer Publishing (1996).

Schneider G. & Winters J. (1998). *Applying Use Cases: A Practical Guide*. Addison-Wesley, Object Technology Series.

Schneiderman B. (1998). Designing the User Interface: Strategies for effective Human-Computer Interaction. Addison Wesley.

Schwabe D. & Rossi G. (1998). Developing Hypermedia applications using OOHDM. *Proceedings of Workshop on Hypermedia development Process, Methods and Models, Hypertext´98.*

Schwabe D. (2001). A Conference Review System. 1st *Workshop on Web-oriented Software Technology. Valencia, June 2001.* http://lasanya.dsic.upv.es/iwwost/

Selic B. (1999). Using UML in the Real-Time Domain. *Communications of the ACM,* 42 (10), 46-54.

UML Version 1.3 (1999). Unified Modeling Language. The Object Management Group. http://www.omg.org

Warmer J. & Kleppe A. (1999). *The Object Constraint Language.* Addison Wesley.