Extending UML for Modeling Web Applications

Luciano Baresi, Franca Garzotto, and Paolo Paolini
Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza Leonardo da Vinci, 32 - 20133 Milano (Italy)
{baresi, garzotto, paolini}@elet.polimi.it

Abstract

Web sites are progressively evolving from browsable, read-only information repositories to web-based distributed applications. Compared to traditional web sites, these web applications do not only support navigation and browsing, but also operations that have affects their contents and navigation states. Compared to traditional applications, web applications integrate operations with the built-in browsing capabilities of hypermedia.

These novelties make web application design a complex task that requires the integration of methods and techniques developed in different "worlds". This integration is achieved in this paper by extending and customizing the Unified Modeling Language (UML) with web design concepts borrowed from the Hypermedia Design Model (HDM). Hypermedia elements are described through appropriate UML stereotypes. UML diagrams are also tailored to model operations and relate them with hypermedia elements. The approach is exemplified by describing the design of a web-based conference manager.

Keywords
Web applications, Web Design, UML, HDM

1 Introduction

The main purpose of this paper is the introduction of W2000, a framework for designing web applications based on two preexisting assets: UML ([4]), the standard notation for modeling object-oriented systems, widely accepted by the software engineering community, and HDM ([12]), an hypermedia model recognized as the ancestor of a family of several design models.

With the advent of the web, hypermedia and information systems, traditionally very far apart, are converging to define a new area of interest: web sites are becoming complex operational environments, and information systems on the web are adopting navigation as a fundamental interaction paradigm.

Web applications are different from “traditional” hypermedia for three main aspects:

- Users do not only navigate, but also activate operations and transactions;
- The hypermedia structure itself may evolve, as the application evolves;
- Different users may have different visibility of the information and different capabilities for the operations.

The essence of these features is not completely new, but their relevance is becoming the key factor: they are crucial and characterizing aspects, rather than marginal as it was in the general case. Consider for example an e-commerce web site and its shopping bag*:* while users are browsing the catalog, they can bookmark the products of interest, move products in their cart, evaluate the total and, providing additional information, complete the buying transaction.

The need for information systems to become *navigational* was anticipated several years ago. The motivation was to augment the power given to users for accessing the information, allowing them to fully exploit the linked nature and interface facilities of hypermedia. Again, consider for example the main legacy systems to which web-based interfaces have been added recently.

The merging of these evolutions leads to web applications. Their sophisticated dynamics and evolution make operations affect their contents and navigation state. They do not only modify the contents of individual pages, but also add or delete groups of pages and links. Shopping carts are typical examples of dynamic elements that can be created and destroyed by users.

Web applications address a potentially huge variety of different users with different navigational and functional requirements. Thus they must face the problem of providing different visibility levels – on contents, navigation, and functionality - to different categories of users. Again this is not completely new [5], but it is assuming an increasing relevance and importance, since huge communities of non-expert users interact with web applications, like modern portals, with different needs, skills, and customization requirements [17].

The above considerations show that web application design is a complex task that requires the integration of different methods and techniques. Analogously to hypermedia design, it requires the ability of organizing large structured or semi-structured contents in a non-linear way, and of defining the multiple navigation paths across them. Analogously to traditional application design, it requires the ability of specifying the functional and evolutional aspects.

Unfortunately, the design methods, models and techniques

from hypermedia and software engineering cannot just be borrowed and piled up: they must be integrated to create a viable and usable conceptual framework. This is why this paper proposes W2000, an integrated framework that blends together the Unified Modeling Language (UML, [4]) and the last version of HDM (the Hypermedia Design Model, [12]), which was the first design model proposed for hypermedia-hypermedia applications and inspired, among the others, OOHDM [20], RMM [16], HDMlite [11], WebML [7]). UML has been chosen because of its being a standard, its graphical and intuitive representation, and its extensibility for representing domain-specific notations. The integration between UML and HDM consists in:

- Defining several stereotypes and customizations of diagrams to render HDM with UML;
- Specifying guidelines to use UML as a way to specify some of the dynamic and operational aspects of web applications;
- Refining use case diagrams to describe high-level user requirements, related to both functional and navigational aspects.

We consider interface design as a separate task, with high complexity on its own: by no means it is a sub product of the other design activities. We do not consider interface design in this paper, but we pave the ground for a possible further extension of W2000 in this direction.

In this paper, we exemplify W2000 through the design of a web-based conference manager system. The case study is complex enough to highlight all the design requirements raised by a real life application, and to exemplify the use of the various modeling concepts and notational primitives; lack of space obliges us to discuss only a little portion of it.

The rest of this section shortly introduces our running example. Section 2 provides an overview of the different design activities for web applications. Sections 3, 4, 5, 6 describe the design activities in detail and exemplify them on the case study. Section 7 compares our approach to other similar approaches. Section 8 concludes the paper and outlines our future work.

1.1 Running Example

A web-based conference manager system should guide all different users involved in a conference to accomplish their tasks. The application has to suitably promote and advertise the conference and its structure. In addition, it has to support authors while submitting papers, to guide program committee members in reviewing papers, and to help the general (program) chair selecting papers and setting up a program. Involved roles impose a set of trivial constraints on the way they can use the application. Generic users should be allowed to browse only through the general information pages; they should never browse through submitted papers and reviews. Authors should be able to access information about their papers, but not those of other papers nor the information about the reviewing process. PC members should see all papers and maybe reviews, except those for which they have declared conflicts. The chair must be able to do everything. After accepting papers, the application should notify all authors, asking all authors of accepted papers for the camera-ready version.

2 W2000 Design Framework

W2000 organizes the design activity in a number of interdependent tasks, as summarized in Figure 1. Each activity produces a model (i.e., a set of related diagrams), which describes some aspects of the web application. Figure 1 does not define “yet another design process”, but it identifies mutual dependencies among design tasks. A number of activities can run in parallel, and designers may need to rework several times the same issue, refining or modifying a portion of the specification with respect to design decisions taken during other tasks.

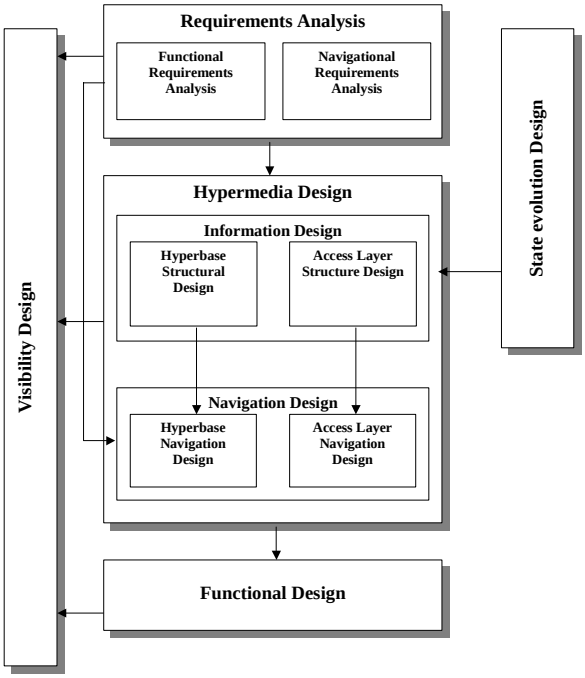

Figure 1: W2000 Design Framework

Each block of Figure 1 identifies a design activity:

Requirements analysis. It extends “conventional” requirements analysis to hypermedia applications. It consists of two sub-activities: *navigational and functional requirements analysis*. The former aims at highlighting the main information and navigation structures needed by the different users of the application. The latter identifies the main user operations. Both activities borrow *UML use-case diagrams* to clearly represent their outcome.

State evolution design. It supplements requirements analysis and defines how the application contents evolve. This activity is not mandatory, but it is required only for applications with complex behaviors. The straightforward way of representing evolution is through *UML statecharts diagrams*. Besides their usual purpose of making explicit the state through which an element evolves, statecharts diagrams become a means to reason on some peculiar aspects of the application and on "temporal" constraints.

Hypermedia design. It consists of *information and navigation design*. The information design specifies and organizes the application contents. According to HDM, the *hyperbase structural design* structures the "core" information that must be available to users. The *access structure design* organizes the contents into higher-level structures (collections in HDM), integrated if needed with "superimposed information" [9], to support access paths to the base contents. The navigation design defines how users can navigate the information elements and access structures.

All delivered diagrams are based on *UML class diagrams*, tailored with appropriate stereotypes.

Functional design. It specifies the main user operations of the application. It extends the specification of standard functions with some peculiarities specific to hypermedia applications. Designers have to provide scenarios for all the main activities in functional use-case diagrams. Being able to explain how identified information objects cooperate to complete an operation is twofold: it is a check on the completeness of the hyperbase design and identifies all operations that must be associated with selected objects.

Extended *UML interaction diagrams* (both sequence diagrams and collaboration diagrams) describe how the information objects identified so far cooperate to provide promised services.

Visibility design. It is a key feature of many web applications. Different users, in general, have a *different perspective* of the application, its contents, and operations. The purpose of visibility design is thus to specify which operations, information structures and navigation paths must be visible to whom.

Lack of space prevents us from taking into account visibility design in detail. The reader should bear in mind that for most design specifications there could be additional versions tailored to specific roles, which specify what is available to whom.

3 Requirements Analysis

Requirement analysis can start focusing on the different categories of users, hereafter roles, which can interact with the application. Roles elicit and help organize both navigation paths and operations. To exemplify requirements analysis, we refine the informal description of the web-based conference manager presented in Section 1.1. The main roles involved in this application are:

- *Generic users* browse only public conference information;
- *Authors* submit papers and browse all relevant information on their papers;
- *PC members* submit reviews, browse all papers, and discuss paper acceptance;
- *Conference chair* assigns papers to PC members and defines the program.

Notice that physical users are not statically associated with single roles. A PC member, for example, can also be an author, or an author is also a generic user. Specifying roles is the best way to make user profiles explicit and to avoid duplicating functionalities and navigation paths for all users that could utilize them.

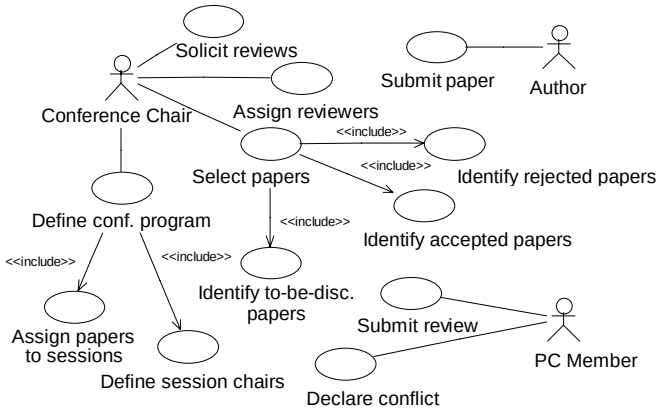

Figure 2: Functional use-case model

Figure 2 presents the *functional use-case model* for the conference manager system[1]. It identifies the main functionalities and associates them with roles. Generic users do not appear in the functional use-case model, but only in the navigational use-case model, because they can only browse through public pages, but they do not perform particular actions. The conference chair *assigns reviewers* to each submitted paper, *solicits reviews* from late reviewers, *selects papers*, and *organizes the conference program*. Selecting papers consists of (<<include>> in the diagram) identifying accepted, rejected, and to-be-discussed papers. Organizing the conference means assigning papers to sessions and associating each section with a session chair.

defines the *navigational use-case model*, that is, the navigation capabilities associated with each role. Generic users can only *browse public conference site*, *browse "accepted" authors*, and *browse selected papers*. Browsing the conference site means surfing through general information, conference program, and program committee.

[1] In this simple example, each use-case model comprises a single use-case diagram. More complex applications could require more than a single diagram.

Authors can see all information related to their papers (*browse paper info*) and *browse reviews*. Authors can only see the reviews of their papers, thus the note associated with the relation between the role and the functionality constraints the visibility of the operation. PC members can *browse papers*, *browse reviews*, except those of papers for which they declared conflicts, and *browse authors*. Browsing papers can be refined (<<extend>> in the diagram) in *browse rejected papers*, *browse accepted papers*, and *browse to-be-discussed papers*.

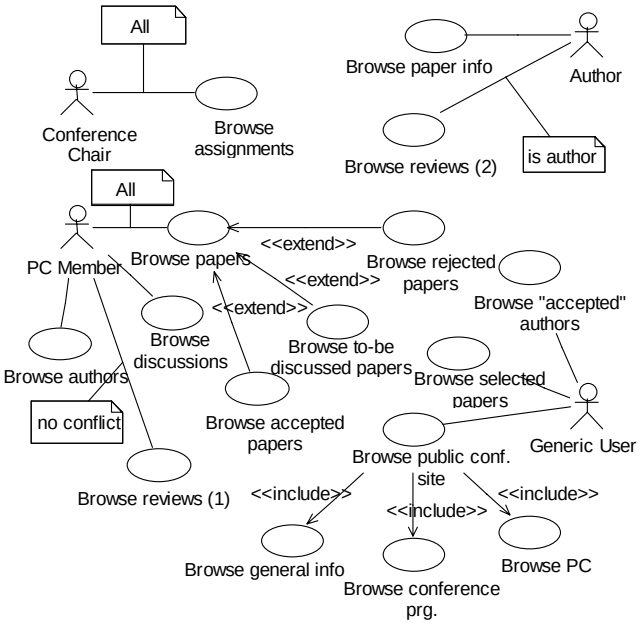

Figure 3: Navigational use-case model

The conference chair can *browse assignments*, that is, all assignments between papers and reviewers.

After defining the requirements models, designers should have a rough, but complete, view of what they are about to do. All other models should use this first set of diagrams as reference and context. As reference, because consistency must be enforced through the whole design process; as context, because here the whole application is available at a glance, the other models are much more focused and detailed.

4 State Evolution Design

A distinctive feature of web applications is that their hypermedia schema is subject to evolution. Potentially every information object and every navigation path is subject to evolution, i.e., could have a number of different states, with transitions defined to move from one state to another. W2000 supplies *state evolution diagrams* to describe how applications evolve. State evolution diagrams are not mandatory for all application elements. They are required only for those specific cases where a significant evolution is foreseen.

State evolution diagrams are rendered with UML statecharts diagrams, which are exemplified in Figure 4. The diagram describes how *papers* evolve within the application. A new *paper* element is created when a paper is submitted to the conference. The creation corresponds to the first arrow and moves the paper from state *Start* (the bullet in the upper part of Figure 4) to state *Submitted*. When the deadline for submitting papers expires, all submitted papers move to state *TBR* (To Be Reviewed) and wait for PC members to submit their reviews. Each paper requires that three reviews be available. When submitted, the application computes its preliminary score. If it is greater than 12 (the predefined upper bound), the paper is automatically accepted and it moves to state *Accepted*. If the score is less than 6 (the lower bound), the paper is rejected, and thus it enters state *Rejected*. If the score is between the two bounds, the paper has to be discussed by the PC members (state *TBD* – To be discussed). Finally, after discussion, it is either accepted or rejected and thus its state becomes *Accepted* or *Rejected*, respectively. Rejected papers move to state *Notified* as soon as the application notifies corresponding authors with the rejection. Authors of accepted papers are asked for the camera-ready version of their papers (state *WCR* - Waiting for Camera Ready) and when received, the paper is *Ready* and can be presented. After being *Presented*, a paper can be moved to the end state.

The state sequence of Figure 4 does not only clarify the behavior of *papers*, but provides also designers with interesting feedbacks to structure the hyperbase. States can impact design in different ways. For example, they can require:

- A particular flag, attribute, to mark the current state of each information element;
- A special-purpose filter that extracts, from all elements of a given type, only those with specific properties;
- A specific container (collection, in the HDM jargon) that includes only selected elements.

5 Hypermedia Design

The purpose of hypermedia design is to specify the information structures (*information design*) and navigation paths (*navigation design*) needed by the various classes of users. Notice that hypermedia design is user-centered: It is intended to interpret and model information and navigation as they are perceived by users, rather than capturing structures for implementation. As such, the hypermedia design schema is substantially different from a database schema.

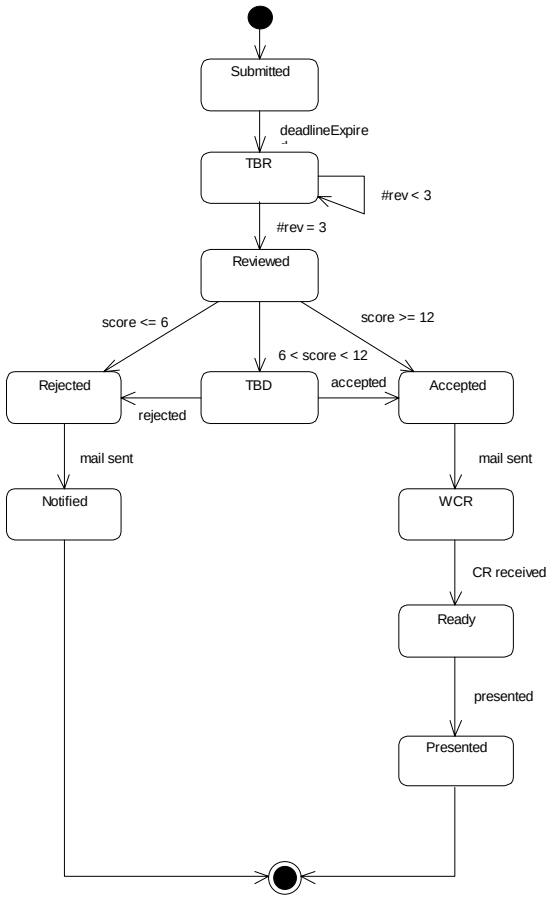

Figure 4: State evolution diagram for conference papers

HDM prescribes that hypermedia design is organized in two distinct layers: the *hyperbase layer* and the *access layer*. The hyperbase layer defines the base information objects, their mutual associations and the navigation paths across them. The access layer introduces alternative groupings and organizations of the base information elements and organizes the ways users initiate their trips within the information space. These two layers are further organized in: design *in-the-large*, which frames the information structures and navigation, and *in-the-small*, which completes all missing details.

The rest of this section illustrates the hypermedia design of the web-based conference manager through several examples.

5.1 Hyperbase information design

HDM organizes the hyperbase information in *entity types* and *semantic association types*[2]. An entity type describes a class of information objects perceivable by users. For example, the conference manager requires: *paper*, *review*, *PC member*, *OC members*[3], and *author*.

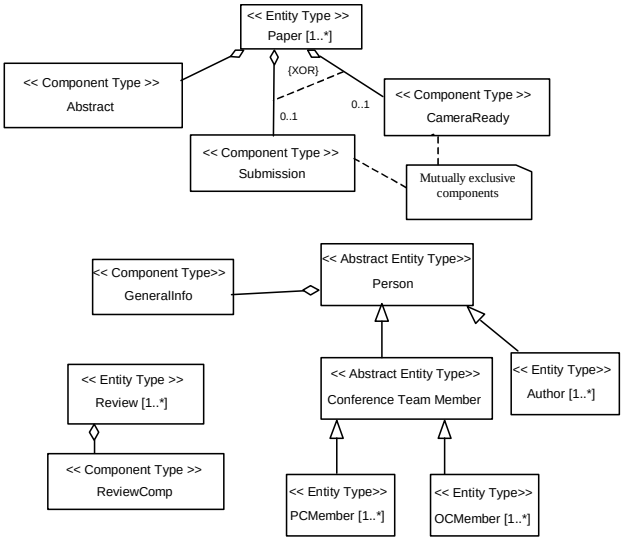

Figure 5: An excerpt of the information design schema (in-the-large)

All HDM concepts are rendered with special-purpose stereotypes as presented by the excerpt of the information structures, defined for the running example, of Figure 5. It shows that entity types can be organized in inheritance hierarchies and are structured in components.

Components are in part-of relations with the entities they belong to. They are information subunits that are not self-contained, but have well-defined roles within entities. Components may have their own subcomponents. In Figure 5, the *paper* entity type comprises three components: *abstract*, *submission*, and *camera ready*. Not all three components simultaneously exist during the life of a *paper* entity since the state of an entity may change as the application evolves: Abstract and submission exist after submitting the paper. The camera ready replaces the initial submission after acceptance and submission of the final version. This property is expressed in Figure 5, by the *xor* label between the two component types and by the associated comment. Notice that this behavior implements the requirements of the state evolution diagram of Figure 4.

Semantic associations denote domain-specific binary relationships that are of interest from the user perspective. They are defined in particular diagrams (see Figure 6) using UML associations. A semantic association can connect entities, components, or other associations. For example, the semantic association *reviews* relates a PC Member to the association *has*, which in turn connects each paper assigned to a PC Member with the review he or she has done for that paper.

[2] HDM allows also users to model singletons as isolated entities without defining ad-hoc types. This modeling option is not further addressed in this paper.

[3] *OC member* stands for organizing committee member.

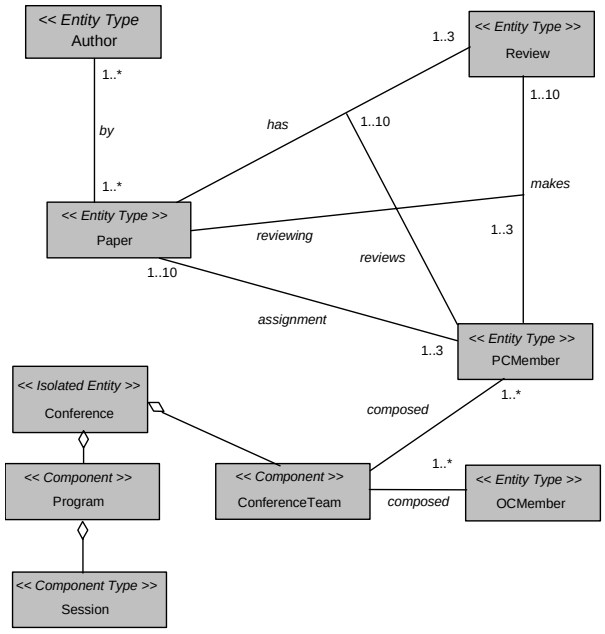

Figure 6: Information design semantic associations (in-the-large)

Figure 5 and Figure 6 concentrate on the design in-the-large, without any specifications of the actual contents of the various objects. As to the design-in-the-small, HDM allows contents to be associated with components and semantic associations. A data value is called *slot*, which can both be a primitive piece of information (e.g., a string, a date, an integer) or be a complex value (e.g., a video or a sound track). In both cases, slots are regarded as atomic: the designer is not interested to its inner structure (defined at implementation level). Slots are typed: They can be *simple* or *composite*, i.e., aggregations of other slots, and are associated with a multiplicity. For example, Figure 7 shows the in-the-small specification of the *paper* entity type.

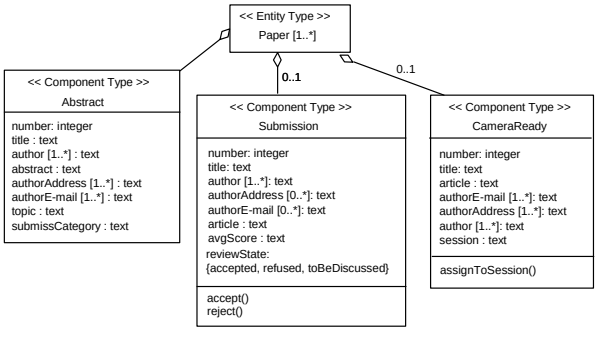

Figure 7: Information design of the *paper* entity type (in-the-small)

Within semantic associations, slots are introduced as part of the description of their *centers*. The center of an association is an information structure that users exploit for navigating the association. For example, the center may contain slots to describe which are the linked objects to help users properly select the one of their interest before actually navigating to it. In most cases, the slots within association centers are borrowed from the linked objects; in some cases, however, additional slots must be introduced (e.g., for explanation purposes). Figure 8 shows the design in-the-small of the *has* semantic association, where slots are specified explicitly.

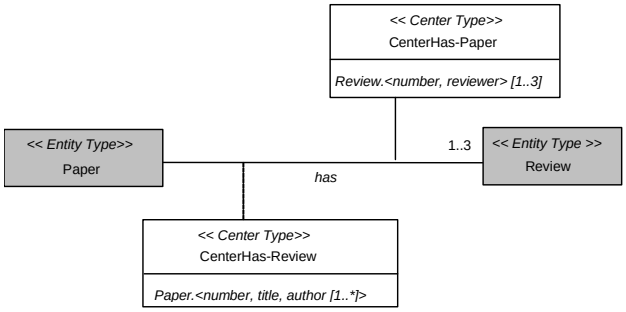

Figure 8: Information design of the *has* semantic association (in-the-small)

5.2 Hyperbase navigation design

Navigation design defines the *navigational nodes* (*nodes* for short) and *navigational links (links)* of the application.
Nodes, which are derived from the hyperbase structural elements (entities, components, association centers), are the information units as perceived and navigated by users. Usually, they are rendered as web pages, or as well identifiable logical blocks in a page.
Nodes, rendered using the UML stereotype *node type*, are derived from the structural design through a set of rules and design decisions. In the simplest case, nodes correspond to leaf entity components and to association centers. Usually, additional nodes can be introduced to facilitate the access to the various information structures.
A link is a path that connects two nodes. Links, represented with UML arrowed associations, are organized in two categories: *structural* links, which are induced by part-of associations among components, and *semantic* links, which are induced by semantic associations.
Figure 9 shows the node-link structure for the *paper* entity type. The navigational topology is hierarchical: The root corresponds to the *abstract* component. The symbol "@" indicates that this node is a default, that is, all users, who access a paper, must start navigation from this node.
Navigation design cannot be automatically derived from the information design, but must be consistent with it. Navigation design implies design decisions and can benefit from *navigation design patterns* ([15], [18]), which

provide readily usable solutions to well-known navigation problems.

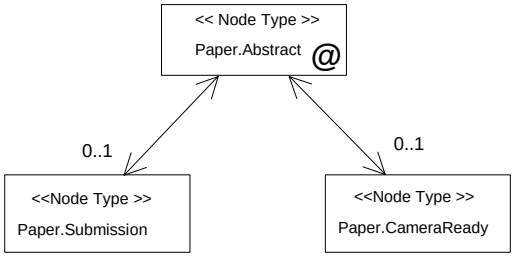

Figure 9: Navigation design of the *paper* entity type

For example, Figure 10 describes the navigation induced by the one-to-many semantic association *has:* From each review, a link may lead directly to the corresponding paper. From a paper, an intermediate node, which corresponds to the association center, can be reached. This node lists the reviews available for the paper. Users must select an item in the list if they want to get all details of a particular review. To access another review, users must return to the center node and make another selection. This is the straightforward navigation pattern called *index*, expressed with the symbol .

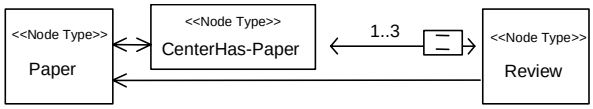

Figure 10: Navigation design of the *has* semantic association

A different navigation pattern can be used to relate papers to authors: The centers are not independent nodes, but are included within the default nodes of the respective entities (and will be presented in the same page). The navigation follows the *guided tour* pattern: Users can scan sequentially all authors of the same paper without navigating back to the center. These examples are only two variants of the many navigation patterns available in the literature.

It is important to notice that the navigational diagrams presented so far give only an in-the-large specification of the navigation, that is, the core navigational decisions of the designer. A number of details are still missing and they should be specified as part of the *navigation design in-the-small*. For example, we should clarify what happens when users reach the last node in a guided tour sequence. Navigation design in-the-small should specify if users can only go back to the previous node, or also return to the first node or to the center node.

5.3 Access layer information design

The access layer design describes the information structures that are super-imposed [9] to the hyperbase and provides alternative organizations of the base contents to users to facilitate the understanding of the hyperbase. The access layer structural design consists of a number of *collections types*. A collection represents a container of entities, components, associations, or other collections. These elements are called collection *members*, and can be selected and organized according to different criteria. A collection may include (and usually does include) a distinguished element called *collection center*. Its main purpose is to help users understand what the collection is about and which are its members, and to serve as starting point for navigation.

The in-the-large specification of collections consists of the types of its members and (if exists) its center. Optionally, the definition includes also the ordering and sub-grouping criteria to organize the members, and filtering criteria by which members can be algorithmically selected.

Once more, collections are rendered with the UML stereotype *collection type*, while their centers are represented using *center type*. An example of this notation is shown in Figure 11.

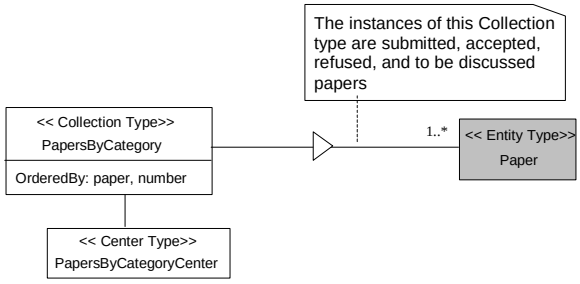

Figure 11: Information design of the *paper by category* collection type

Designing a collection in-the-small means specifying the actual contents of the collection center. A collection center should contain enough information to allow users identify the collection members. It usually includes a list of slots borrowed from the members, to outline what the members are in a compact, but comprehensible way. The collection center may also introduce *new* structured or unstructured information, such as comments, explanations, content maps or orientation maps, and similar visual or textual cues, to improve usability and understandability of the collection itself.

5.4 Access layer navigation design

The access layer navigation design consists of defining the navigational links (called *collection links)* for each collection. These links define how users can navigate both between the center and members of a collection, and across members. We assume that navigation within a collection starts from the node corresponding to the center, or from the default node of the first member element, if the center

is missing. The multiple ways to access the member nodes must be described both in-the-large (by means of navigation patterns) and in-the-small, by providing the details of all collection links. The index and guided tour patterns described in the previous section, and all their possible variants, apply to collections as well, and we use the same stereotypes to represent them in UML. An example of navigation in-the-large within the *reviews* collection is shown in Figure 12, which exploits the "Index + Guided Tour" pattern.

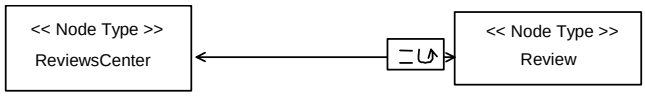

Figure 12: Navigation design of the *reviews* collection

6 Functional Design

The functional design describes how the different elements of the hypermedia cooperate to accomplish the actions defined in the functional use-case model. Functional design is strictly related to both state evolution design (since the applicability and the "meaning" of operations depend upon states) and hypermedia design (since information objects and navigation paths are used by the operations).

Scenario diagrams are rendered using suitably extended *UML interaction diagrams* (both sequence and cooperation diagrams). Extensions concern both objects and message passing. Involved "objects" are organized in *entities (components, nodes), semantic associations*, and *collections.* Message passing is not only pure invocation of methods (functions) associated with identified elements. Web applications require also *free navigation*, represented with dotted lines, and *constrained navigation* through a particular link, represented with lines with diamonds in the middle. Free navigation allows users to "move around" without following specific navigation paths. In contrast, constrained navigation forces users to follow predefined navigation paths.

In this paper, we exemplify *scenario diagrams* through the extended *UML sequence diagram* of Figure 13: It shows how papers are selected, that is, what information is necessary and how it is used to compute the preliminary scores associated with submitted papers. The example refers to a single paper; to address all papers, the same operation should be applied to all members of the *submitted papers* collection.

The conference chair selects a paper through a free navigation (dotted line), which means that he/she is not obliged to follow a particular navigation path. Once the paper is selected, he/she explores the associated reviews through constrained navigations (lines with diamonds), traversing the links induced by the *has* semantic association. Before computing the score (the user wants to *evaluate* the paper), the application removes the paper from the collection of *submitted paper*s. Then, it *compute*s the preliminary score associated with the paper. The outcome of this operation can be any number between 0 and 18 and the possible actions depend on its actual value. UML sequence diagrams allow alternatives to be represented on the same diagram by drawing multiple arcs exiting from the same point and by associating each arc with a predicate to decide which path must be followed. In this case, if the score is greater than 12 the paper is added to the collection of *accepted papers* and the state becomes *accepted*. If the score is less than 6, the paper is added to the collection of *reject papers* and its state becomes *rejected*. Finally, if the score is between 6 and 12, the paper is added to the collection of *TBD papers* and its state becomes *TBD*.[4]

A complete specification for a web application should contain at least one scenario diagram for each operation identified in the functional use-case model. Usually a single scenario is sufficient for simple and sequential operations. More complex functions, with nested alternatives and deep modifications of the hyperbase, usually require several scenarios to represent all possible and meaningful alternatives. Squeezing all alternatives in a single diagram would lead to unreadable models.

An important cross check is the consistency between the navigation paths, as they are used within these diagrams, and their definition in the hypermedia design.

7 Related Work

To the best of authors' knowledge, W2000 is the first approach that integrates in a single unified framework most of the powerful structural and navigational abstractions, introduced by hypermedia web models, with functional and behavioral primitives provided by UML. Other approaches do not cover completely either hypermedia modeling or behavior specification and thus they all lack in specifying the mutual relations between the two aspects.

If we consider UML extensions specific to hypermedia modeling, UHDM [2] renders OOHDM (an object-oriented extension of HDM [20]) by extending UML with suitable stereotypes and OCL constraints. It defines a *conceptual model*, which is a "standard" class diagram that identifies the domain entities together with their relationships, a *navigation model*, which is a customized class diagram that identifies the classes of the conceptual model and the navigation among them, and a *presentation model*, which describes the abstract user interface by means of UML object and statecharts diagrams.

[4] Notice that the diagram represents also the different states in which the paper can be.

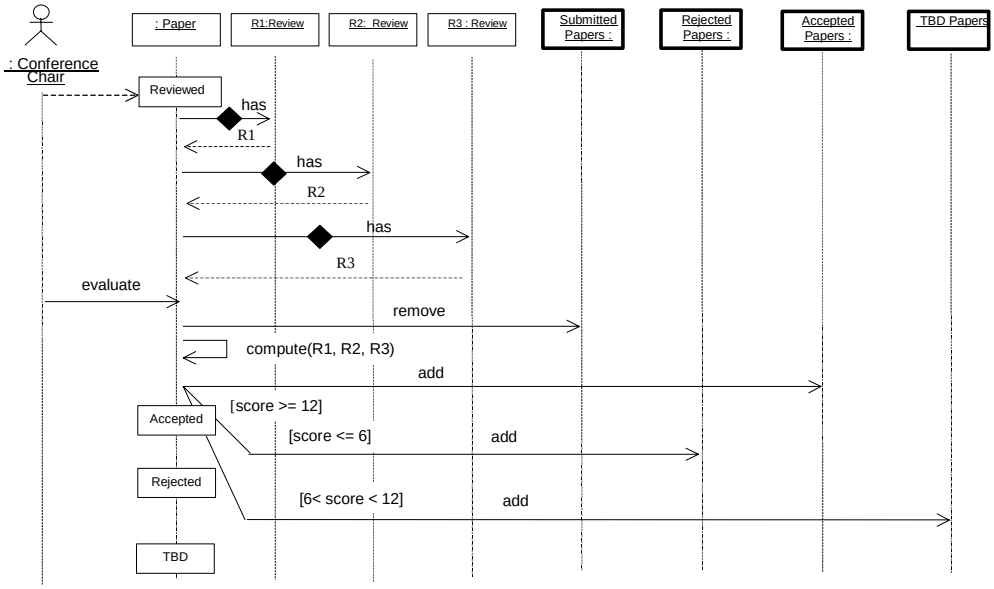

Figure 13: Scenario diagram for *Select papers*

Since UHDM is based on OOHDM, the main differences between UHDM and our approach are the same as those between OODHM and HDM2000.

Given the generality and extensibility of UML, all well-known hypermedia models (e.g., HDM [12], HDM-lite [11], RMMM [16], OOHDM [20], Araneus [1], WSDM [10], WebML [6] [7]) could be represented with suitable customizations of UML, and, in principle, they could have been adopted for hypermedia design in our framework. In contrast, we have chosen HDM2000 for a number of reasons. HDM2000 decomposes the information and navigation schemas in two layers -- hyperbase and access layer -- and allows the designer to define them at two levels: in the large and in the small. These abstraction levels do not appear in any of the above-mentioned models, but, to the best of authors' experience, they are extremely useful to plan the design activity, to make it more organized, and to structure design documentation more effectively. In addition, HDM2000 introduces a number of novelties with respect to HDM, and other models, that are useful to specify complex design situations. For example, it provides isolated entities and isolated collections, identifies different roles for elementary data units (slots), includes content attributes in the definition of associations (see the notion of center), and incorporates a set of sophisticated navigation patterns as built-in design primitives [15].

A different approach to web modeling is presented in [19]: UML is not extended, but it is simply used to associate a standard graphical interface with Jessica, an existing object-model for designing web-services. Besides UML syntax, Jessica elements are defined by means of XML code to allow designers to define web services by specifying a (too simple) UML model and automatically obtaining the XML equivalent. The direct mapping between Jessica (UML) and XML precludes any modeling abstraction and functional behavior is not addressed.

Considering web applications modeling, the use of UML has been discussed by Conallen in [8], which aim at proposing a workable solution for releasing web applications. The proposal privileges client-server interactions and underestimate the logical vs. physical design of both information and navigation structures. It defines stereotypes, tagged values, and OCL constraints to model web pages and hyperlinks, forms, frames, and client-server components at a concrete level (for example, a web page is seen as a fragment of html code). Conallen adapts also all classical phases of software development to web architectures, and tailors almost all UML diagrams to render web-related concepts. For example, use-case diagrams are used to represent requirements and interaction diagrams both exemplify use cases and show how objects interact. But, neither use-cases nor interaction diagrams are extended to address navigation issues.

8 Conclusions and Future Work

This paper introduces W2000, an UML-HDM integrated framework for the design of web applications. The underlying assumption is that web applications are complex, in their nature, thus their design is necessarily a complex task. Web applications are not standard applications, with "more data" to take into account, nor are hypermedia applications with a few operations added. Web applications design requires the integration of two distinct but interrelated activities: hypermedia design, which focuses on information structures and navigation paths, and functional design, which focuses on operations. To integrate the two

"different worlds" W2000 proposes:

- Special-purpose extensions to use-cases to capture both operational and navigational requirements;
- An UML-based description of HDM, the supporting hypermedia design model;
- An extension to standard UML dynamic diagrams to model web operations

In this paper, all presented concepts have been exemplified only on a web-based conference manager system, but the effort for the definition of W2000 is paying off. We are apparently able to specify a number of interesting (and difficult) operational features, and to integrate them with the hypermedia design. However, our future agenda includes:

- To keep using W2000 for defining further web applications to have a large base set of case studies.
- To fine-tune the definition of W2000 to improve expressiveness without impairing readability (tradeoff always difficult to keep under control).
- To extend the approach to address presentation design issues.
- To restructure the suite of design tools JWEB ([3]), which was previously based on the original HDM, to support all steps of W2000.

9 References

[1] P. Atzeni, G. Mecca, and P. Merialdo: "Design and Maintenance of Data-Intensive Web Sites", Proc. EDBT 1998, pp. 436-450.

[2] H. Baumeister, N. Koch, and L. Mandel: "Towards a UML Extension for Hypermedia Design", in Proceedings of UML´99 The Unified Modeling Language - Beyond the Standard, LNCS 1723, Fort Collins,USA, October 1999, Springer Verlag

[3] M. A. Bochicchio, R. Paiano, and P. Paolini, "JWeb: an HDM Environment for Fast Development of Web Applications", in Proceedings of IEEE Multimedia Computing and Systems 1999 (ICMCS '99), Vol.2, pp.809-813.

[4] G. Booch, I. Jacobson, and J. Rumbaugh: "The Unified Modeling Language User Guide", The Addison-Wesley Object Technology Series, 1998.

[5] P. Brusilowsky and Milosavljevic (guest eds.) "The New Review of Hypermedia and Multimedia - Special Issue on Adaptivity and User Modeling in Hypermedia Systems", Taylor Graham, Vol. 4, 1998

[6] S. Ceri, P. Fraternali, S. Paraboschi: "Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications", in Proc. VLDB'99, Edinburgh, September 1999, Morgan Kaufmann, pp. 615-626.

[7] S. Ceri, P. Fraternali, A. Bongio: "Web Modeling Language (WebML): a modeling language for designing Web sites", in Proc. Int. Conf. WWW9, Amsterdam, May 5 2000

[8] J. Conallen: "Building Web Applications with UML", Addison-Wesley, 2000.

[9] L. Delcambre, D. Maier, "Models for Superimposed Information", In Proc. Of the International Workshop on the World-Wide Web and Conceptual Modelling, WCM'99, Paris, 15 Nov. 1999.

[10] O.M.F. De Troyer, C.J. Leune, "WSDM: a user centered design method for Web site", in Proc. Of Int. Conf. WWW7

[11] P. Fraternali, P. Paolini: "A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications", Proc. EDBT 1998, pp. 421-435.

[12] F. Garzotto, P. Paolini, D. Schwabe, "HDM - A Model-Based Approach to Hypertext Application Design", TOIS 11(1) (1993), pp.1-26

[13] Garzotto F., Mainetti L., Paolini P., "Hypermedia Design, Analysis and Evaluation Issues", Communications of the ACM, Vol.38, No.8, Aug.1995, pp. 49-56.

[14] F. Garzotto, L. Mainetti, P. Paolini, "Navigation in Hypermedia Applications: Modeling and Semantics", in Journal of Organizational Computing and Electronic Commerce, 6 (3), 1996

[15] F. Garzotto, P. Paolini, D. Bolchini, and S. Valenti: "Modeling by patterns" of Web Applications, in Proc. Of the International workshop on the World-Wide Web and Conceptual Modelling, WWWCM'99, Paris, 15 Nov. 1999, pp. 293-306.

[16] T. Isakowitz, E. Stohr, P. Balasubramanian: "RMM: A Methodology for Structured Hypermedia Design", CACM (1995), 38(8), pp. 34-44.

[17] MILIA - The World's Interactive Content Marketplace - THINK.TANK SUMMIT" - panel on "Personalization and Customization for E-commerce" Cannes (France), Feb. 2000.

[18] G. Rossi, D. Schwabe and F. Lyardet, "Improving Web information systems with navigational patterns", in Proc. Of Int. Conf. WWW8, Elsevier, pp.589-600.

[19] M. Schranz, J. Weidl, K. Göschka, and S. Zechmeister, "Engineering Complex World Wide Web Services with Jessica and UML", In Proceedings of the 'Hawaii International Conference On System Sciences HICSS-33', Maui, Hawaii, USA, Jan 4-7, IEEE Computer Society 2000, p.167.

[20] D. Schwabe, G. Rossi, "An Object Oriented Approach to Web-Based Application Design", Theory and Practice of Object Systems, 4 (4), J. Wiley, 1998