Engineering Web Applications for Reuse

Daniel Schwabe *, Gustavo Rossi **, Luiselena Esmeraldo *, Fernando Lyardet**
***Departamento de Informática, PUC-Rio, Brazil**
E-mail: {schwabe, luiselena} @inf.puc-rio.br
****LIFIA Facultad de Informática, UNLP.**
La Plata, Argentina
E-mail: {fer, gustavo}@sol.info.unlp.edu.ar

Abstract

In this paper we present Web design frameworks as a conceptual approach to maximize reuse in Web applications. We first analyze the current state of the art of Web applications design, stating the need for an approach that clearly separates concerns (conceptual, navigational, interface). We briefly introduce the OOHDM approach for Web applications design. We next focus on the problem of design reuse in Web applications. After a short review the state of the art of object-oriented application frameworks we present the rationale for a slightly different approach focusing on *design* reuse instead of *code* reuse. We then present OOHDM-Frame, a syntax for defining the hot spots of generic Web applications designs. We illustrate the use of OOHDM-Frame with a case study in the domain of Conference Paper Review Systems**.** We finally discuss how to implement Web design frameworks in different Web platforms.

1 Introduction

Building complex Web applications such as E-commerce applications is a time consuming task. We need to understand the underlying domain (objects, behaviors, business rules, etc). We must carefully design the applications' navigational architecture and user interface, if we want them to be usable. We must clearly understand the user tasks in order to decide which navigation facilities we should include (indexes, guided tours, landmarks, etc.), according to the user needs. The interface should help the user browse through the sea of information by giving him cues and feedback on his actions, and by presenting the information in a clear and meaningful way.

These applications often act as integrators of distributed data or behavior repositories, and usually support different user profiles. They typically evolve over time, so *modularity and separation of concerns* are clearly a must. At the same time, by their very nature, these applications should be deployed quickly and with zero defects; we must be able to center our development enterprise into *a reuse metaphor, combining design reuse and components reuse.*

We have been designing Web applications using the Object Oriented Hypermedia Design Method (OOHDM) for some years [7]. OOHDM considers Web applications as *navigational views* over an object model and provides some basic constructs for navigation (contexts, indexes, etc) and for user interface design. Using OOHDM we can apply well-known object-oriented software engineering practices to the construction of applications involving navigation. We have been looking at ways to maximize reuse in the development process, since we have observed a certain degree of commonality among solutions in similar application domains. For example, most online stores have similar navigation structures, and they provide similar functions to their users.

In this paper we introduce Web design frameworks as a novel concept to push design reuse in Web applications. We use the OOHDM notation in the examples though the concepts underlying this paper can be used with other approaches such as WebML [1], HDM2000 [4], etc. In section 2 we discuss separation of concerns in Web applications, and briefly introduce the OOHDM design approach. In section 3 we discuss design reuse and introduce Web design frameworks by showing how they compare with traditional application frameworks. In section 4 we introduce our notation

by discussing an example in the field of Web reviewing applications. In section 5 we address the problem of framework instantiation and evolution. Finally in section 6 we discuss further work on engineering for reuse.

2 Separation of concerns in Web applications

As previously stated, for engineering a Web application we must solve diverse problems. Current trends shows that we have to deal with increasing complexity at different levels: sophisticated application domains (financial, medical, geographical software, etc); the need to provide easy browsing access to large volumes of multimedia data and finally the appearance of new appliances and devices for which we must build (easy to use) Web interfaces. This complexity in software can only by dealt with by separation of modeling concerns in a clear and modular way. Separation of concerns provides not only a solid ground for Web software evolution but also, as we explain in this paper, a basis for maximizing design and implementation reuse. We briefly introduce the OOHDM approach by presenting the three concerns addressed by the method.

2.1 Application behavior

The core of every software application (be it Web-based or not) is its domain or conceptual model. It must reflect which objects the application deals with, their relationships and behaviors. In OOHDM the conceptual model is specified using the UML notation [10].

The distinctive aspect of Web applications is their navigational architecture: they are basically hypermedia applications. We must define which objects (nodes) the user will perceive and how he will traverse the hyperspace (links, indexes, etc). Nodes and Links are defined according the user profile and the task he must perform**.**

Consider the domain of conference paper review systems (i.e., web-based applications that help manage program committees in the process of receiving and evaluating papers for a conference). A possible conceptual model for this domain is given in Figure 1. Notice that this model allows reviewers to express their degree of interest in (i.e., bid for) certain papers, as well as indicating their degree of expertise over the conference topics.

2.2 Navigational Modeling

For the purposes of supporting the evaluation process, we have defined a “view” over this schema, where Nodes and Links are Observers (See [3]) of conceptual classes as shown in Figure 2. Each Node class can be defined by combining attributes from different conceptual classes, while links indicate navigation paths and reflect conceptual relationships. For example, “Person” has incorporated “Expertise” and “Topic” as an attribute; similarly, “Paper” has also incorporated “Topic” as an attribute.

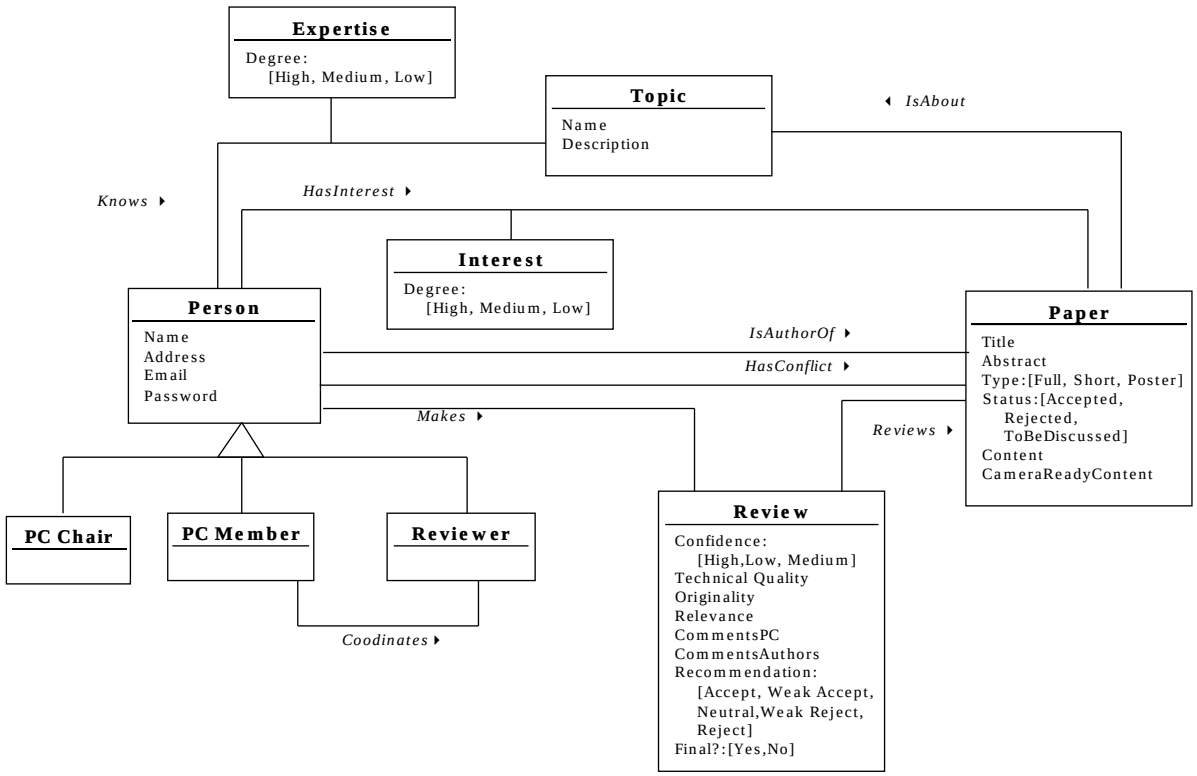

Figure 1 – Conceptual Model for the "Conference Paper Review System" domain.

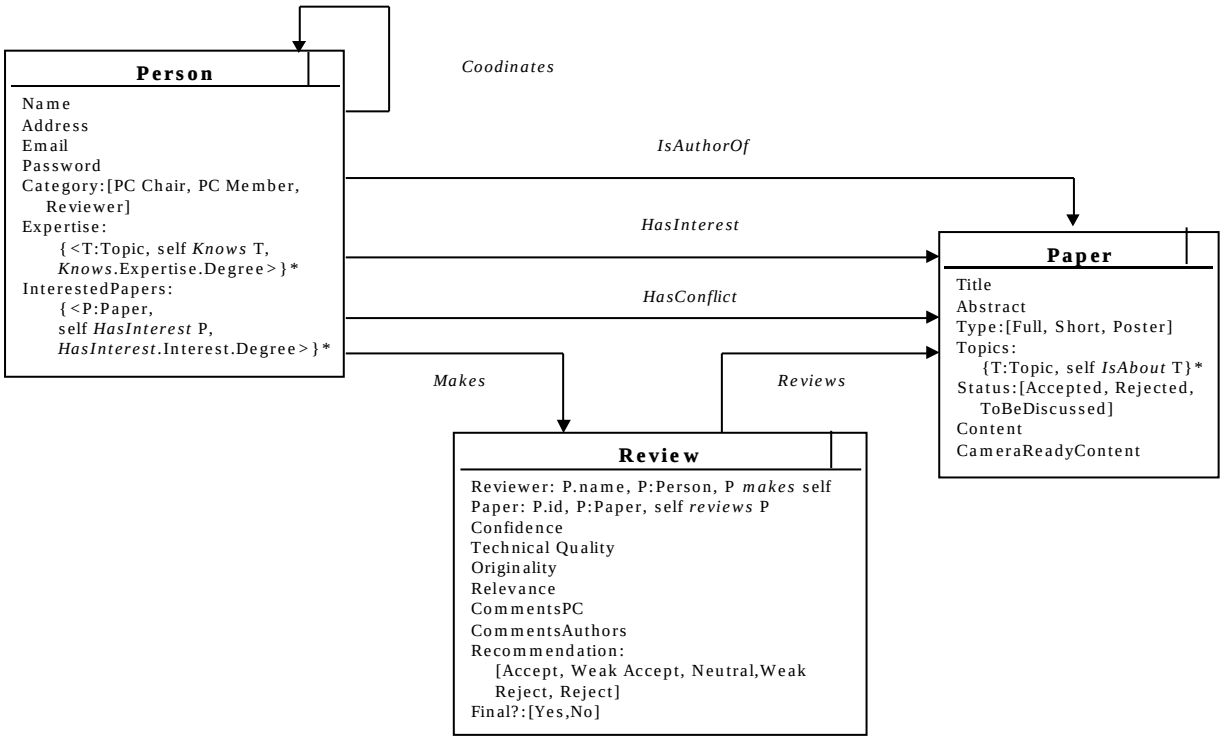

Figure 2 – Navigation Class Model for the "Conference Paper Review System" domain

In OOHDM we have extended the UML notation for defining Node and Link classes. Nodes attributes are defined using an Object Query Language that allows picking attributes from classes related with the observed one [7]. The navigational schema contains Node classes and Link relationships that indicate the basic navigation architecture. The navigational schema only contains

links derived from “semantic” relationships (e.g. the relationship IsAuthorOf between Person and Paper); we can not express “finer grained” links such as those among papers of the same author or papers reviewed by the same reviewer.

In OOHDM we define Navigational Contexts as Sets of nodes that may be accessed using indexes and traversed in some order or freely. Contexts can be defined by a query predicate such as all instances of Node class Paper with Topic=”Hypermedia”. Navigational contexts are specified using a slight addition to the UML notation by taking into account their nature as sets of nodes. The navigational context schema also shows indexes and notable entry points in the hyperspace (landmarks). An index is a navigation object that acts as an access structure containing references to other navigation objects. Since the same node may appear in more than one context, for example, “Paper by Author” and “Paper by Reviewer”, with different attributes and outgoing links (for example for reaching the next/previous node in the context), these differences are expressed using InContext classes (decorators with the semantics of [3]). In Figure 3 we show a possible navigational context schema of the example.

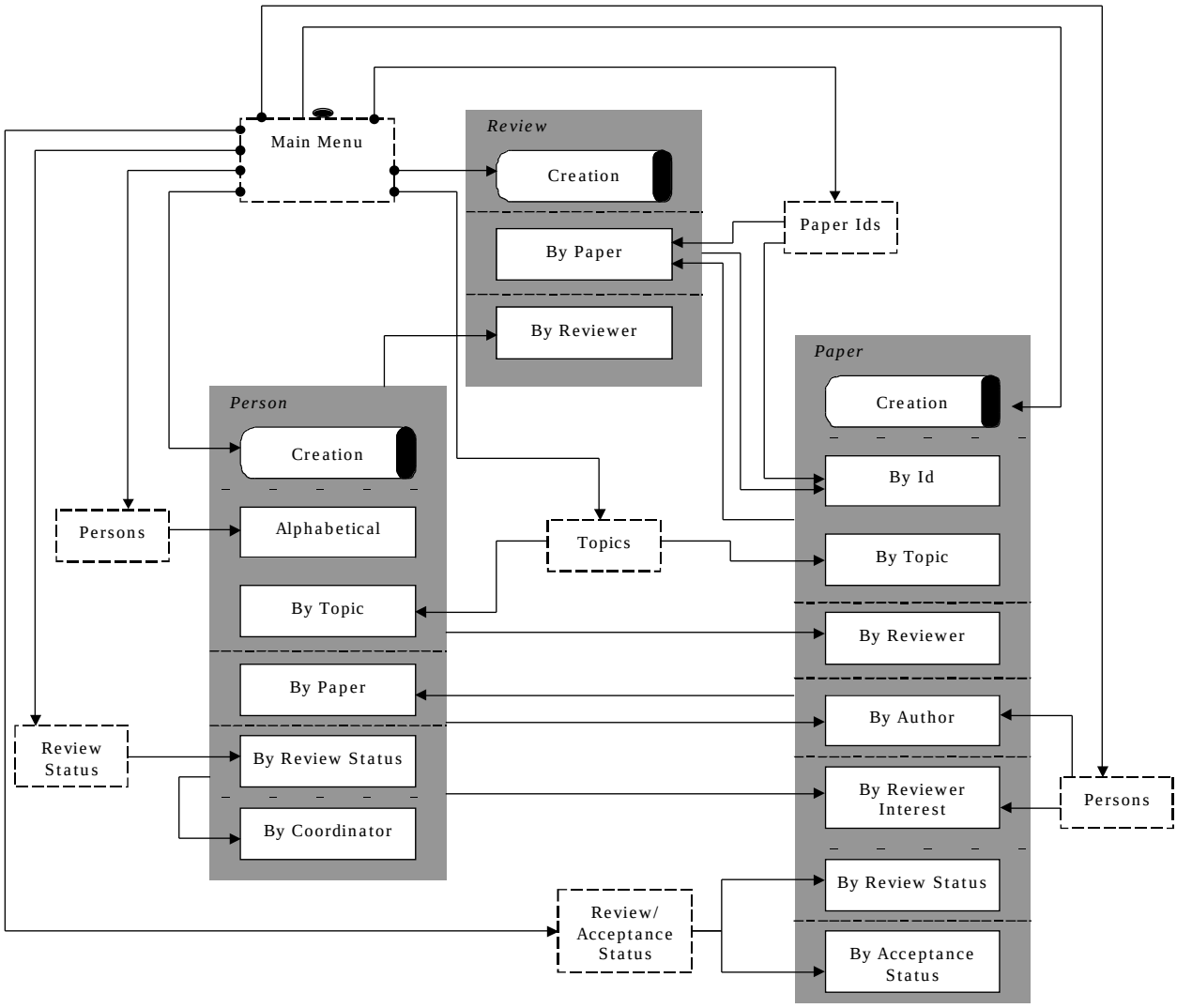

Figure 3 – A Navigational Context Schema for the “Conference Paper Review System” domain.

In this diagram, rectangular boxes with full borders denote contexts (sets of nodes); the label of each gray area indicates the class of the elements in the contexts within it. For example, “Persons Alphabetical” stands for the set of persons in alphabetical order. This information is actually contained in accompanying “Context Specification Cards”, which are not shown here for reasons of

space; the reader is referred to [7] for more details. Boxes with dashed borders indicate access structures (indices), which also have corresponding "Access Structure Specification Cards".

In the example, it is specified that, for instance, papers may be navigated in several ways, such as "By Reviewer" (i.e., all papers assigned to some given reviewer), or "By Topic" (i.e., all papers classified under a given topic), etc.

Since this application domain allows submission of new papers, creation of reviews, and registration of new reviewers, there are corresponding dynamic contexts (denoted by a black bar on the right), within which new instances of the appropriate navigation classes may be created. This is also true for the instantiation of relations (e.g., assignment of a paper to a reviewer), which may occur during navigation of the "Paper by Reviewer Interest" context.

Since operations in this domain depend on the identity of the reader, the Main Menu is a protected access structure, indicated by the small black oval at the top. This means that only authorized readers may access it and continue navigation; the Access Structure Specification card spells out the different user classes and access rights.

2.3 Interface Design

Finally, the third concern involves the specification of the user interface. For a given navigational model (i.e. a particular user profile) we may have to define different interface models according to the particular interface device he/she will use, e.g. an internet browser, a mobile phone, etc. In OOHDM we have used a formal model of interface objects that also considers the interface as a set of Observers of navigation objects; for reasons of space, we will not further discuss user interface issues in this paper.

The most outstanding contribution of the OOHDM approach is the clear identification of these three design concerns in an implementation-independent way. Even if we choose to use a non object-oriented implementation (for example combining relational stores with ASP or JSP-based page generation), we get a clear understanding of the various design aspects and their rationale, thus simplifying system evolution and maintenance.

3 Design Reuse in Web applications

By separating concerns into conceptual, navigational and interface, we get a basis for reasoning about design reuse in this kind of applications. It should be clear that in the conceptual model we could apply well-known object-oriented reuse techniques such as defining abstract classes, and building application frameworks by systematically applying well-known design patterns [Gamma95, Fayad99]. In this work however, we are interested in showing how to push framework technology further into the Web domain.

In a previous paper [6] we introduced navigation patterns as a way to record, convey and reuse design experience. Navigation patterns, as their counterparts in object-oriented design, show how to go beyond the basic Web navigation paradigm to solve recurrent problems. We have found that micro-architectural reuse is feasible, and shortens development times. For example the Landmark navigation pattern (see [6]) explains how to model relevant sub-sites (or nodes) that must be reachable from every point in the application. In our example (Figure 3) they are indicated by arrows with a black circle at the source**.**

Although the kind of reuse provided by patterns is valuable, complex Web applications need a way to maximize reuse of larger design structures. These design structures may arise at the application (conceptual level) or during navigational design. For example, the set of activities triggered when users order an item in an electronic store is usually similar in different stores. Similarly, the way in which they may navigate items in those stores is also similar. We should be

able to express these commonalties in such a way that only the specific aspects of a particular store e.g. particulars payment procedures, different ways of browsing through the store) should be designed or programmed. Similarly, although different paper review systems may vary somewhat in the individual procedures for selecting papers – e.g., some employ a two-tiered system, others don't – they all have common functionality, such as reviewer registration, paper submission, assignment of paper to referees, review submission, etc.

We next introduce Web design frameworks as a solution to this problem. We first review the state of the art in object-oriented frameworks and then highlight the differences with Web design frameworks.

3.1 Object-Oriented Frameworks.

An object-oriented application framework is a reusable design built from a set of abstract and concrete classes, and a model of object collaborations in a particular domain. An application framework acts as the skeleton of a set of applications that can be customized for a particular application. When many different applications must be constructed in the same domain, application frameworks provide "templates" for supporting their commonalties, and accommodating individual variations (differences). These "templates" usually have the form of abstract classes that must be sub-classified with concrete ones, or filled with "hook" methods that must be implemented by the application's designer [5]. One of the important defining aspects of a framework are its hot-spots, i.e., the places in the framework where the designer may introduce the variations or differences for a particular application in the same domain as the framework.

Building frameworks is a challenging and usually difficult task since we must understand the structure and behavior of different applications in the same domain, and be able to express these aspects in a generic way. However, once we have specified an application framework, the task of implementing individual applications is simplified. Evolution of applications is also more stable as we make them evolve together with the framework [2].

The basic philosophy behind frameworks (i.e. building reusable designs) can be applied to Web applications. However, we must carefully analyze and understand the variation points in Web application design models and the kind of abstractions involved in this kind of application.

Unfortunately, current modeling notations (such as the UML) do not provide good support for the specification of frameworks. Existing approaches emphasize the use of a specific programming language in which the hot spots are defined. Instead, we are interested in expressing generic Web designs using a simple notation that can be then used to generate either an application framework (in a particular language) or a running application in the intended domain. We next characterize Web design frameworks and explain our notation using the preceding example.

3.2 The architecture of Web design frameworks

We define a Web *design* framework as a generic design of possible Web application models, including conceptual, navigational and interface aspects, in a given domain. Web design frameworks are different from Web *application* frameworks because whereas the latter are programmed in a specific language, Web *design* frameworks are environment and language-independent. A Web design framework will contain a reusable application model in a particular domain that can be later instantiated into specific applications in that domain.

Our previous discussion on separation of concerns gives us a basis for characterizing the components of a Web application, which we can use to design for reuse. In Figure 4 we elaborate the OOHDM approach to show an abstract diagram of a Web framework architecture, indicating points of flexibility (i.e. those design components where we should design the framework "hot-spots")

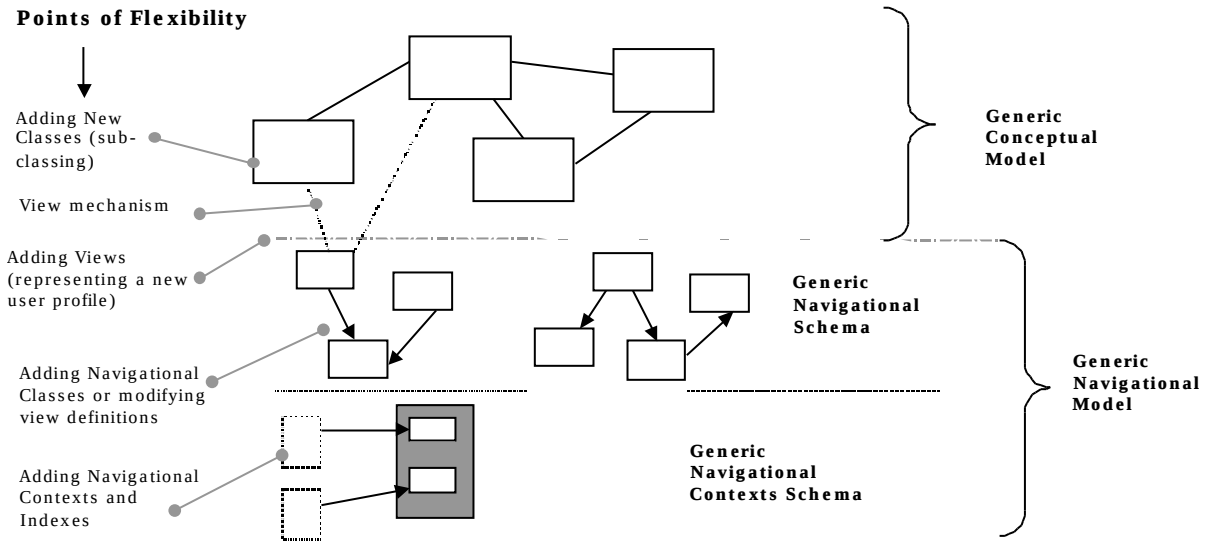

Figure 4 - Components of a Web Design Framework

A Web Design Framework generalizes the design model of a Web application for a complete family of Web applications. In Figure 4 we indicate two main Generic Models, the Conceptual and Navigational Models; the Navigational Model is described using a view mechanism.

The Navigational Model itself has two sub-models: the Generic Navigation Schema and the Generic Navigational Context Schema. In each model we can find places where we can design for reuse (expressed as Points of Flexibility). For example, the well-known mechanism of sub-classing provides a way to design flexible (conceptual and navigational) models.

We next refine this discussion by examining how we obtain genericity and reusability in the previously described architecture. We also elaborate the idea of points of flexibility. For reasons of space we will not discuss reuse at the interface level.

3.2.1 Reuse in the Conceptual Model

It is clear from Figure 4 that we have at least two places where we can specify hot spots: in the conceptual model and in the navigational model. Genericity in the conceptual model can be achieved using well-known object-oriented reuse techniques [2]. In particular, the hot spots of the conceptual model for the application domain can be defined according to [5].

In our example, it is possible to extend the conceptual model by defining sub-classes. For instance, “Paper” could be sub-classed into “Full Paper”, “Short Paper”, “Poster”, etc. Given that designing for reuse at the conceptual level employs well-known object-oriented techniques (See for example [2]), we will focus on the design of reusable navigational models.

3.2.2 Reuse in the Navigational Model

Variability in the navigational model can be achieved in different ways:

- By building completely new applications from the same conceptual model (e.g. defining a new user profile). In our example, one could build an application where PC members are allowed to delegate evaluation of papers to others, previously unknown (to the system) persons or a view for the general chair of the conference to evaluate how review is proceeding.
- By defining a generic navigational schema (that allows adding new Node or Link classes and refining the definition of attributes). In our example, one could redefine the “Recommendation” attribute of a review, for instance, defining them as numeric values as opposed to enumerated as specified in By achieving genericity in the Navigational Context Schema (i.e. defining abstract access structures and navigational contexts). In our example, one could allow several

alternative ways to group papers (into navigation contexts) – by recommendation type, by review status, by referee preference, by author, etc. Each application will typically offer only a few of these.

A Web design framework is thus the combination of a generic conceptual schema and generic navigational and context schemas. It can be readily seen that with this combination we get a high degree of genericity in which we can change the underlying application model, add or refine profiles or tasks, and define different navigation topologies for specific applications. A particular application is then obtained first by exercising the hot spots in each schema, obtaining a concrete design, and then by mapping this design into an implementation environment. We next discuss OOHDM-Frame, our notation for specifying Web design frameworks.

4 Specifying Web Design Frameworks

A Web design framework is a generic design for a family of Web applications. We have added some simple primitives to OOHDM in order to allow the specification of flexible design structures. Our purpose here is not to present the notation in full detail (the reader can refer to [9]) but mainly to show how we can achieve genericity in Web design models. As we will show, there are many different ways to achieve genericity; not all of them need to be used together necessarily. We may need to build completely different applications in the same domain (for example different kinds of electronic stores); we may instead need to accommodate different user profiles (views) in the abstract model of one application and then specialize it for each profile (new user profiles of the same store). For the sake of simplicity we discuss design mechanisms in an independent way; combining them is straightforward, although it should be noted that design complexity might grow.

4.1 Genericity in the Conceptual Model

Genericity in the conceptual model is achieved by abstracting the classes and behavior of different applications in the family. As commented in Section 3.2.1, this objective can be achieved using object-oriented design techniques. OOHDM-Frame uses the UML notation to specify generic conceptual models. The only extension to that notation is that dashed elements are optional, meaning that they may be omitted in any particular framework instantiation.

We next refine the discussion in Section 3.2.2 by explaining how to specify reusable navigation models

4.2 Specifying different user's profiles

This kind of reuse lies in the middle between the conceptual and the navigational model; in fact each Web application is considered a view on the conceptual model. Given a particular conceptual model (generic in a domain or specific for an application) we can reuse it for different user profiles, as exemplified for "Person" in the navigation schema in Figure 2.

4.3 Specifying generic Navigational Schema

The Navigational Schema expresses which nodes the user will navigate, which attributes those objects will contain and which links connect those objects. Therefore, Node and Link classes represent the basic navigational architecture of a Web application. When designing a Web framework we may specify abstract Node classes (i.e. with a reduced number of attributes) and Link classes. These classes should be specialized for a particular application in the domain in much the same way as in the conceptual model.

A less trivial variation point in this process can be found in the viewing mechanism, which allows the definition of attributes inside a Node. For example, the "Review" class in the navigation schema in Figure 2 has imported attributes "Reviewer", which is the name of the person that created

the “Review”, and “Paper”, which is the id of the reviewed paper. Similarly, “Reviewer” has an expertise list of pairs <topic, degree>, derived from the conceptual relation class “Expertise” associated with the “Knows” relation between “Person” and “Topic”.

4.4 Expressing abstract Navigational Contexts

Different applications in the same domain may contain different navigational contexts. In some conferences, for example, reviewers may be allowed to “bid” for papers of interest to be assigned to them for reviewing. In this situation, it might be useful to have a context gathering all papers of interest for each reviewer, to help the PC Chair to make the final assignments. For other conferences where this is not the case, such a context would not be necessary.

In order to express such generic contexts, the OOHDM primitives are extended with the notion of a Generic Context, which stands for a possible set of contexts, defined parametrically by stating constraints over the properties that define possible instances. Such Generic Contexts are denoted by double-dashed rectangles, as can be seen in Figure 5. For example, the “Paper by Property” generic context in that diagram stands for a (possibly singleton) set of contexts based on “Paper” properties — e.g., “Recommendation=Accepted”, “Status=ToBeDiscussed”, etc.... Generic contexts are further specified through Specification cards, which detail the constraints and other aspects governing instantiation (see Figure 6).

The same reasoning applies to link-derived contexts, i.e., contexts whose defining property is based on a 1-n link. For example, “Paper by Derived Link with Person” is a generic context that may be instantiated into any context based on any of the relations between “Paper” and “Person” — “has interest”, “has conflict”, “reviews”. In other words, possible concrete contexts for this generic context would be “Papers that Person X has interest in”, “Papers that Person X is reviewing”, etc....

Similarly to Generic Contexts, generic Access Structures allow specifying abstract indexes that can be later instantiated into concrete ones. For example, “Paper Property” in Figure 5 is a generic index that may be instantiated into several indexes, according to the particular contexts instantiated for the “Paper by Property” generic context.

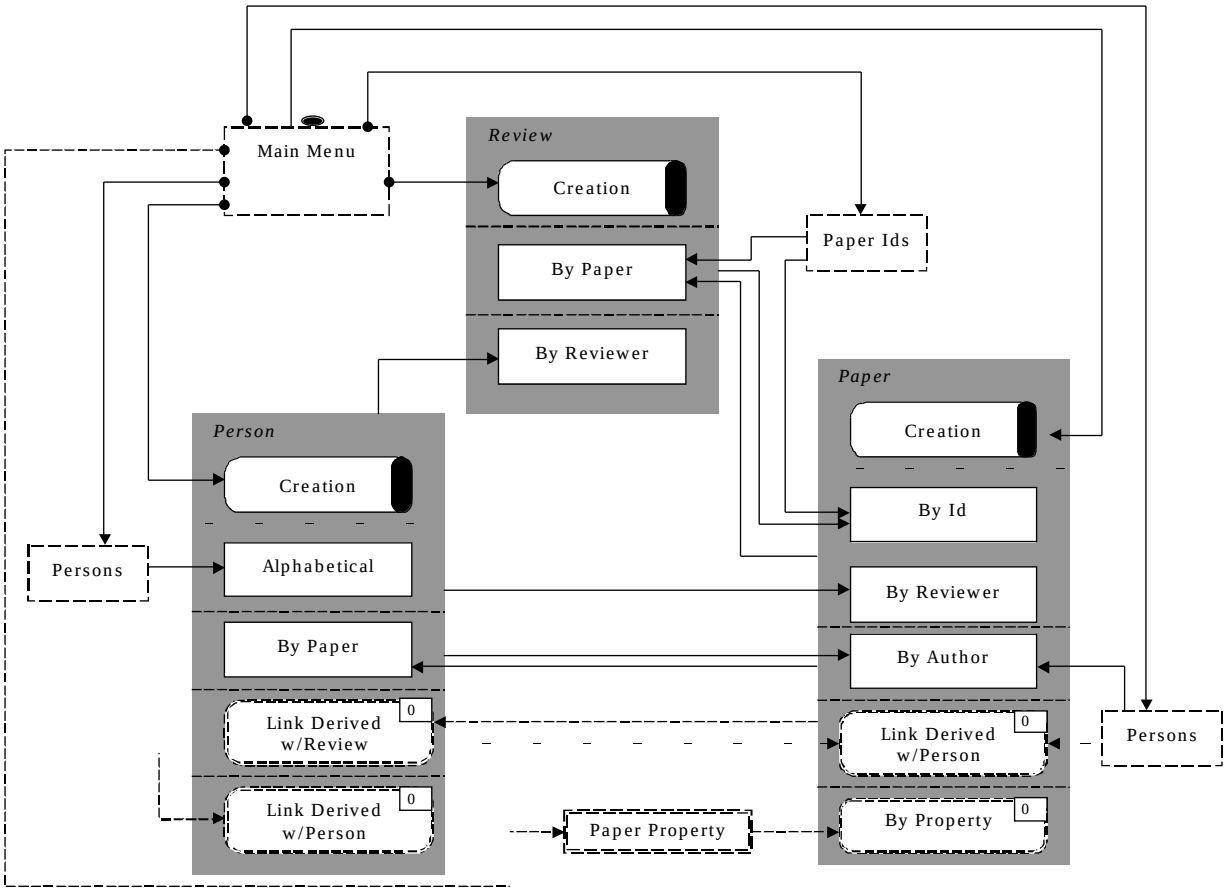

Figure 5 – Generic Context Schema for the "Conference Paper Review System" domain. Double dashed border indicates generic elements (contexts or access structures).

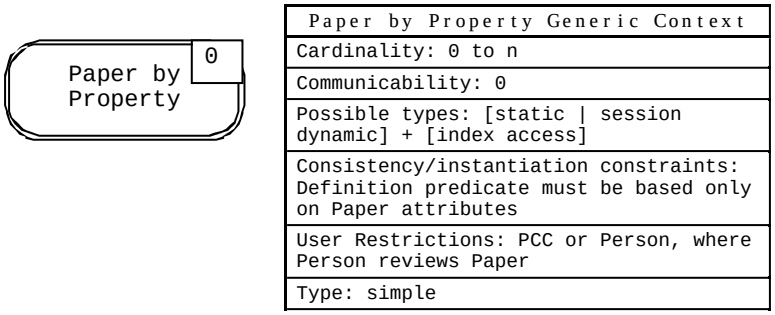

Figure 6- A generic context specification card.

When concrete contexts and access structures appear in the generic context schema, it means that all applications derived from this framework must include them. In a sense, they constitute the common part among all applications built with the framework for the given domain. In our example, the context schema in Figure 5 specifies that all paper review systems must allow:

- The creation of new reviews,
- The submission (creation) of papers,
- The registration (creation) of reviewers,
- Navigation among reviewers in alphabetical order,
- Navigation among all reviewers of a given paper,

- Navigation among all papers by their Ids and
- Navigation among all papers assigned to a given reviewer.
- Review may be navigated by paper Id or by reviewer.

Given the generic context schema in Figure 5, one possible instantiation is the diagram shown in Figure 3. In this case, for instance, the “Paper by Property” generic context has been instantiated into two concrete contexts, “Paper by Acceptance Status” and “Paper by Review Status”. The “Paper by Derived Link with Person” has been instantiated into a single context, “Paper by Reviewer Preference”.

5 Instantiating a Web Design Framework...

Once we complete the design of our application we must implement it in the Web environment. There are many different alternatives to produce running Web applications from a given framework. In this section we briefly discuss two of them: instantiating the design framework into an OOHDM model, and then implementing the resulting model in the Web, or implementing the design framework using a Web application framework [2].

5.1 Into a Web application model

The first strategy for obtaining a specific Web application in the framework domain is to instantiate the abstract model into a valid OOHDM model and then implementing this model using standard Web tools. The process for deriving a concrete OOHDM model from an OOHDM-Frame specification is straightforward as it only involves defining concrete classes (conceptual and navigational) and contexts from the generic diagrams.

We have shown in [7] how to map an OOHDM model into a Web application. We have been using OOHDM-Web [8] for this purpose. In OOHDM-Web, a complete OOHDM design is represented using special purpose data structures, which are basically nested lists of attribute-value pairs. These data structures contain class definitions, navigation context definitions, access structure definitions and interface definitions. These definitions include the description of database entries that store instance data.

Context definitions comprise the query definition that selects the elements that belong to the context; the same is true for access structure definitions. Interface definitions are mixed HTML templates, one for each class in each context where it appears. The mixed HTML templates are pure HTML formatting instructions interspersed with function calls to a library of pre-defined functions, which are part of the OOHDM-Web environment. These functions allow retrieval of object attributes, or reference to other objects in specified contexts. Reference functions are defined in such a way that, when activated by the user, cause the exhibition of the destination object in the appropriate context, using the template defined for that context.

We have generalized this approach for dealing with OOHDM-Frame models by allowing substitution of generic definitions for the concrete ones whenever appropriate. The resulting representation describes the generic design of the framework in question. The instantiation process will substitute the generic definitions in the framework by the definitions (using the OOHDM-Web representation) of their corresponding instantiated elements. For example, a generic class-derived context can be substituted by several class-derived contexts in the instantiated framework; this is achieved by actually replacing, in the data structure that describes de framework, the generic context description by the descriptions of the actual contexts, using the OOHDM-Web format.

At the end of this process, when all hot-spots have been plugged into the corresponding concrete application elements, the resulting data structure is a valid OOHDM-Web representation of the final instantiated application, ready to be used. Our current implementation does not

automatically support all constraint verifications, which must be done manually by the designer while instantiating the framework.

5.2 Into an application framework

Web design frameworks for a particular application domain can be also mapped to an object-oriented framework and, in this sense, they can be used as the design documentation of application frameworks.

We have implemented an object-oriented architecture that allows designers to implement Web application frameworks for specific domains. This architecture contains classes that support the core OOHDM primitives (nodes, links, indexes and contexts); these abstract classes can be plugged into domain specific classes (Paper, Person, etc.) to extend the behavior of these application classes with navigation functionality. Using this architecture, a designer should implement the generic conceptual model using an object-oriented programming language (for example Java), and for each particular application either sub-class or instantiate the domain classes and connect them with the navigation classes and objects that were derived from the OOHDM-Frame generic navigational schema.

This architecture decouples the application and navigational model from the components that provide dynamic content generation on the Web (ranging from CGI/ISAPI to ASP/JSP) and persistence. In this way a Web application framework can be designed to be independent of particular commercial technologies, and can thus evolve seamlessly (see Figure 7).

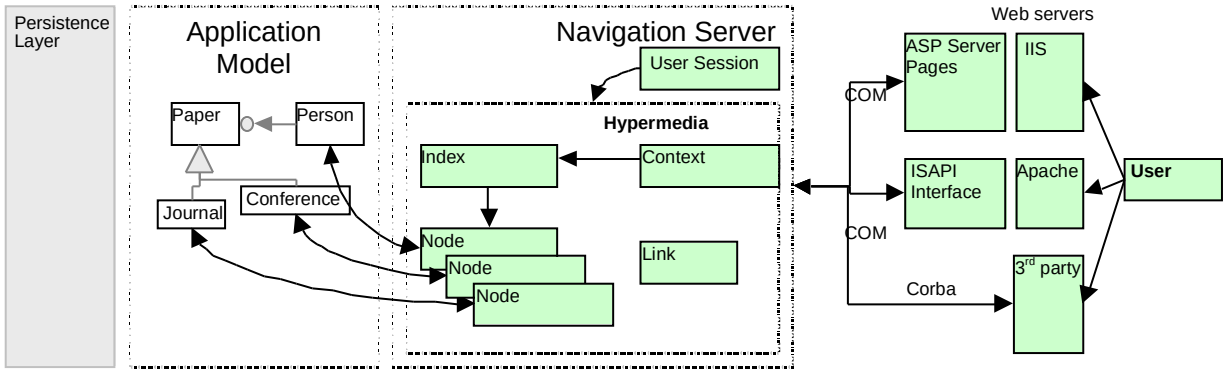

Figure 7 - An architecture for building Web application frameworks

The architecture in Figure 7 has two main components: the Application Model and the Navigation Server. The former will contain all application behavior (as expressed in the OOHDM-Frame generic conceptual model) expressed using an object-oriented language. Meanwhile, classes in the NavigationServer are focused on providing the ability to access nodes in different contexts, managing the navigation spaces and linking among nodes. The entire HTML rendering task is performed on the Web server side by either a custom third-party CGI/ISAPI module or through dynamic HTML. The other components manage persistence and dynamic page generation, typically page servers like ASP or JSP.

6 Discussion and Further Work

In this paper we have discussed how to engineer Web applications focusing on reuse. We emphasized that separation of concerns is paramount in this kind of applications. We introduced Web Design frameworks as a novel technology to further design reuse in Web applications. A design framework contains the specification of both the behavior and navigational structure of a family of Web applications in a particular domain.

We have also introduced OOHDM-Frame, a concise syntax that allows expressing generic OOHDM models that comprise a framework specification. We showed that Web design frameworks can be mapped into an OOHDM model and then into a Web application by using the OOHDM-Web environment or other Web implementation tools. We showed that design frameworks could also be mapped in a straightforward way into object-oriented application frameworks by showing a specific architecture.

One of the most important architectural components in Web Design Frameworks is Generic Navigational Contexts. Contexts are recurrent patterns in Web applications since they usually deal with sets of similar objects. We are now incorporating other navigation patterns into OOHDM-Frame to enhance its expressive power.

We strongly believe that development; delivery and maintenance times in the Web domain require reuse-centric approaches. The systematic reuse of semi-complete design structures, as described by Web design frameworks is a key approach for maximizing reuse in Web application development

7 References

1. S. Ceri, P. Fraternali, A. Bongio: "Web Modeling Language (WebML): a modeling language for designing Web sites". Proceedings of the 9th International Conference on the WWW (WWW9), Amsterdam, May 2000.

2. M. Fayad, D. Schmidt and R. Johnson (editors): "Building Application Frameworks", Wiley 1999.

3. E. Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns. Elements of reusable object-oriented software". Addison Wesley, 1995.

4. F. Garzotto, P. Paolini, D. Bolchini, S. Valenti:"Modeling-by-Patterns of Web Applications"; Proc. International Workshop on the World Wide Web and Conceptual Modeling, WWW CM'99, Paris, November1999

5. W. Pree: "Design Patterns for object-oriented software", Addison Wesley, 1994.

6. G. Rossi, F. Lyardet and D. Schwabe: "Patterns for designing navigable spaces", In Pattern Languages of Programs 4, Addison Wesley, 1999.

7. D. Schwabe, G. Rossi: "An object-oriented approach to web-based application design". Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October 1998.

8. D. Schwabe, R. Pontes, I. Moura: "OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW", ACM SigWEB Newsletter, Vol. 8, #2, June 1999.

9. D. Schwabe, G. Rossi, L. Esmeraldo, F. Lyardet: "Web Design Frameworks: An approach to improve reuse in Web applications. Proceedings of the WWW9 Web Engineering Workshop, Springer Verlag LNCS, forthcoming.

10. UML Document Set. Version 1.013 January, 1997, Rational, 1997. (Available at http://www.rational.com/uml/references/index.html)