Towards a UML Extension for Hypermedia Design* **

Hubert Baumeister[1], Nora Koch[1,2], and Luis Mandel[2]

[1] Institut für Informatik
Ludwig-Maximilans-Universität München
Oettingenstr. 67
D–80538 München, Germany
baumeist@informatik.uni-muenchen.de
[2] Forschungsinstitut für Angewandte Software Technologie (FAST e.V.)
Arabellastr. 17
D-81925 München, Germany
{koch,mandel}@fast.de

Abstract. The acceptance of UML as a de facto standard for the design of object-oriented systems, together with the explosive growth of the World Wide Web has raised the need for UML extensions to model hypermedia applications running on the Internet. In this paper we propose such an extension for modeling the navigation and the user interfaces of hypermedia systems. Similar to other design methods for hypermedia systems we view the design of hypermedia systems as consisting of three models: the conceptual, navigational and presentational model. The conceptual model consists of a class diagram identifying the objects of the problem domain and their relations. The navigational model describes the navigation structure of the hypermedia application by a class diagram specifying *which* navigational nodes are defined and an object diagram showing *how* these navigational nodes are visited. Finally, the presentational model describes the abstract user interface by composite objects and its dynamic behavior by state diagrams. Each model is built using the notations provided by the UML, applying the extension mechanism of the UML, i.e. stereotypes and OCL constraints, when necessary.

Keywords: Unified Modeling Language, Object-Oriented Design, Multimedia, Hypermedia, WWW

1 Introduction

The development of Web applications for Intranets and the Internet in the case of personal homepages or tiny information systems could be seen as an easy job. Such applications are normally written once and probably never modified; directly implemented without taking into account aspects like navigation design.

* This work was partially supported by the Bayerische Forschungsstiftung.
** to appear in the proceedings of UML'99, Fort Collins, USA.

However, when designing applications like Web sites of companies or large Web-based information systems, having hundreds of nodes to be visited, interacting with many different programs and running as distributed systems, a methodology together with a modeling language is needed in order to document the system, to communicate the desired structure and behavior, to visualize and control the systems architecture, showing places where simplification, factorization and reuse can be applied.

Different modeling languages can be used, but the importance of using a standard is clear: it provides a common language which facilitates the communication among project partners as well as to the external world and future readers of the system documentation. Further, tools based on this standard can be used to develop, test and validate the models. UML [BRJ99] has been accepted as the de facto industrial standard for modeling object-oriented systems. It can be extended by using the stereotype mechanism, tagged values and the Object Constraint Language (OCL) [WK99].

Different methods have been proposed for the design of hypermedia systems. From among these we have chosen the OOHDM [SRB96] as the basis for our design method presented in [KM99], because it is an object-oriented method. Further, it allows a concise specification of the navigation by introducing the concept of navigational contexts and it includes a step for modeling the user interface. However, the OOHDM notation is not UML-based.

In this paper we present a UML extension – fully compliant with the UML standard – for modeling hypermedia systems supporting our method. The classical diagrams provided by the UML, like class diagrams or statecharts are not sufficient to model all aspects of hypermedia systems, for instance, to model the navigational space and to represent this model graphically.

Hypermedia systems are complex software systems, therefore most of the methodologies for development complex software systems can be applied to their development process. In particular, the development of a hypermedia system can be divided into five tasks: requirements capture, analysis, design, implementation and test.

The goal of requirements capture is to find out the functional and non-functional requirements of the system. The functional requirements can be captured using use cases and scenarios. An example of a non-functional requirement for Web systems is the decision about the users of an application. If a broad audience is intended, one has to restrict to techniques that all Web browsers understand. Usually this implies the restriction to plain HTML. The use of Javascript, Java Applets, Active-X and plug-ins drastically reduces the reachable audience.

The result of the analysis task is the conceptual model, which is a UML class diagram consisting of classes and associations found in the problem domain. This class diagram may be supplemented by other UML diagrams, like sequence diagrams specifying the behavior of the system.

The conceptual model is the starting point for the design task. This is one aspect where the method for the development of hypermedia applications differs from the development of other software systems. Based on the conceptual model

the navigational structure of hypermedia applications is defined which consists of the navigational class model and the navigational structure model. The navigational class model specifies *which* classes and associations from the conceptual class diagram are available for navigation and the navigational structure diagram specifies *how* the navigation is performed.

The last step in the design task is the presentational design, where a rough version of the user interface is produced. This is done by first defining the user interface objects which are composed of primitive user interface objects, like text, anchor and image as well as other composite user interface objects, giving hints to the final appearance of the user interface objects on the screen. The final decisions about the layout are made in the implementation task. In a second step the behavior of these objects are defined.

The implementation task maps the objects of the presentational model to Web pages, server side scripts and client side scripts, style sheets, etc. in the case the target platform is the Web. Existing software tools for the user interface design can be used to implement the presentational model, e.g. the NeXT Interface Builder, MacroMind Director, VisualWave and JBuilder.

A test model describes the test cases with the purpose of verifying the use cases and checking the navigability of the structure, e.g. dangling links and unreachable pages.

This paper focuses on the notation and techniques used in the design phase and is structured as follows: Section 2 outlines some related work. Sections 3, 4 and 5 describe the conceptual, navigational and presentational models. Finally, we present concluding remarks and an overview of future work in Section 6.

2 Related Work

Some of the works in the hypermedia modeling field only focus on the notation, like the UML extension proposed by Conallen, or on the design process, such as RMM, OOHDM, EORM and WSDM. The latter use standard notation, like E-R notation, OMT or UML, merely for the conceptual design and define their own notation and graphical techniques for the other steps.

Conallen [Con98] defines a set of UML stereotypes for the Web, but does not present a method for the design of Web applications. It includes stereotypes for components, classes, methods and associations, such as server component, client component, server page, client page, form, frameset, link, redirect and submit. These stereotypes are appropriate to model layout and implementation aspects, but not to define the navigation structure of the Web applications.

The Relationship Management Methodology (RMM) [ISB95] addresses the design and construction of hypermedia applications by a process of seven steps. RMM is at the same time a top down and a bottom up approach. During the E-R design step, entities and relationships are identified, which will become nodes and links in the resulting hypermedia application. The second step, slice design, involves grouping entity attributes for presentation. Slices are "presentation units" which appear as pages of hypermedia applications. Separation of contents and

presentational aspects is not fulfilled in this step. RMM specifies navigation by access primitives, such as link, grouping (menus), index and guided tour. The techniques proposed for the user interface design in [BBI96] are elaboration of mock-ups and prototyping.

The Enhanced Object-Relationship Model (EORM) is defined as an iterative process concentrating on the enrichment of the object-oriented model by the representation of relations between objects (links) as objects. According to [Lan96], this has the following advantages: relations become semantically rich as they are extensible constructs, they can participate in other relations and they can be part of reusable libraries. The method is based on three frameworks of reusable libraries: for class definition, composition (link class definition) and GUIs.

The Object-Oriented Design Method (OOHDM) comprises four activities; they are conceptual modeling, navigational design, abstract interface design and implementation [Ros96]. These activities are performed in a mix of incremental, iterative and prototype-based development style. This method sees an application as a view over the conceptual model. The concept of navigational context is introduced to describe the navigational structures. It is a powerful concept that allows different groupings of navigational objects with the purpose to navigate them in different contexts. A special notation is used for the representation of the navigation structure. In earlier papers OMT was proposed [SRB96] as the notation of the conceptual schema; later paper use UML instead [SR98]. However, OOHDM diagrams are not UML compliant as they use own notation for perspectives of attributes in the class diagrams and proposes other kind of diagrams for the navigational and abstract user interface design.

The Web Site Design Method (WSDM) is a user-centered approach defining the navigation objects based on the information requirements of the users of a Web application [TL97]. WSDM consists of three main phases: user modeling, conceptual design and implementation design. In the user modeling phase the potential users of the Web site are identified and classified. Different perspectives are defined for the user classes; these are different ways user classes look at the same information. The navigational model consists of a number of navigation tracks expressing how users of a particular perspective can navigate through the available information. This method defines its own graphical notation for the objects of the navigational model. The navigational design achieves Web applications that have a very hierarchical structure.

3 Conceptual Model

The result of the analysis task is the conceptual model of the problem domain defined by classes relevant to the domain and associations between these classes. The goal is to capture the domain semantics with as little concern as possible of the navigational and presentational aspects. Activities of the conceptual design step are to find classes, to specify attributes and operations, to define hierarchical structures and to determine sub-systems. Well-known object-oriented modeling techniques, such as aggregation, composition, generalization and specialization

are used to achieve this purpose. Classes and associations are described by attributes and operations; they can be organized into UML packages. The conceptual model is the starting point for one or more navigational designs , i.e. more than one hypermedia application. It is represented by a UML class diagram.

The UML extensions presented in this paper are illustrated by the design of a Web site for a service company. The objective of this Web site is to offer information about the company itself, the employees and their relationship to projects and departments. Projects are performed for customers and may establish relationships to partners. For demonstration purposes, we restrict ourself in the example to the classes and associations shown in Fig. 1. Many other aspects may be added, e.g. information about products, publications, events, press releases and job offers.

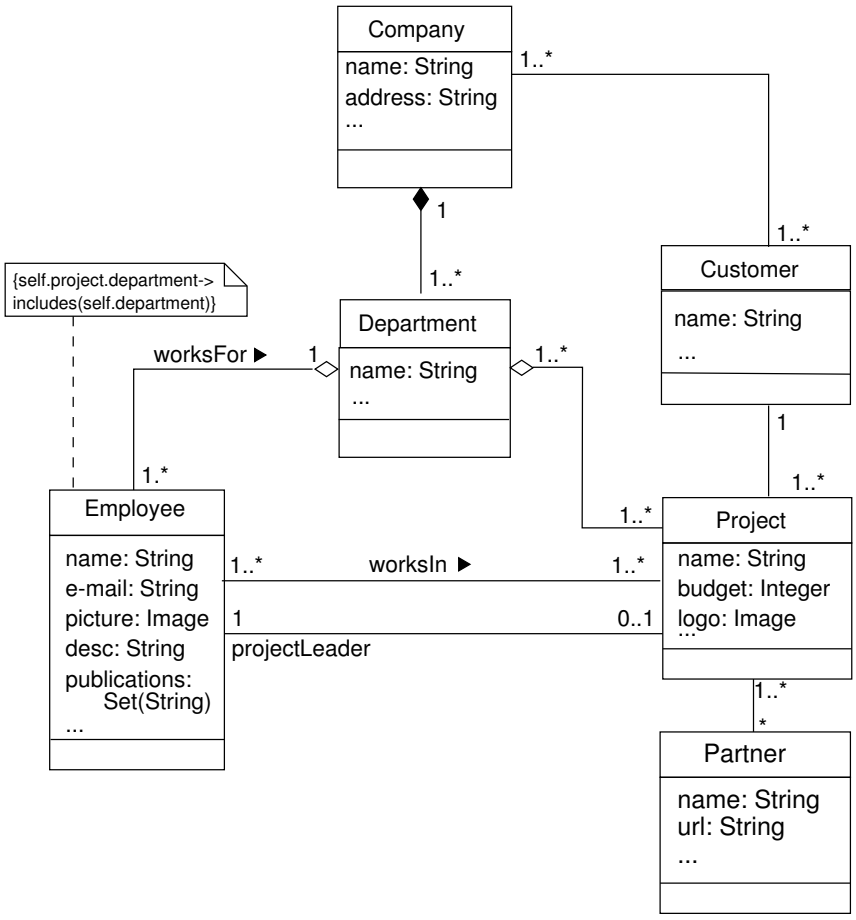

Fig. 1. Conceptual Model of a Company's Web Site.

4 Navigational Model

A hypermedia application is organized into nodes and links that establish relationships of type navigation between the nodes. Nodes that are obtained from objects of the conceptual model are called navigational objects. The navigational design is a critical step in the development of every hypermedia application. Even applications with a non-deep hierarchical structure and a small number of nodes may have a complex navigation structure. Links improve navigability on the one hand, but imply on the other hand, higher risk of loosing orientation. Building a navigational model is not only helpful for the documentation of the application structure, it also allows for a more structured navigability. The navigational design defines the structure of the hypermedia application by building two models: the navigational class model and the navigational structure model.

4.1 Navigational Class Model

The navigational class model defines a view on the conceptual model showing which classes of the conceptual model can be visited through navigation in the application. This model is built with a set of navigational classes and associations, which are obtained from the conceptual model, i.e. each navigational class and each association of the navigational class model is mapped to a class, respectively to an association in the conceptual model. In the navigational class model navigability is specified for associations, i.e. direction of the navigation along the association is indicated by an arrow attached to the end of the association's line. It is possible to attach an arrow to both ends of an association. In this case the user can navigate in both directions along the association. We distinguish for each link a navigational source object and a navigational target object; the latter one may be determined dynamically.

A navigational class is defined as a stereotyped class «navigational class» (cf. Fig. 2) with the same name as the corresponding class of the conceptual model. We use a small box as the icon for the stereotype «navigational class». Navigational objects are instances of these navigational classes connected by links (in UML terms) that are instances of the associations of the navigational model. The navigational class model is usually a sub-graph of the conceptual model where classes and associations which are not needed for navigation are eliminated or reduced to attributes of other classes. The values of these attributes can be computed from some conceptual objects. The formula to compute the derived attribute is given by an OCL expression. A derived attribute is denoted in UML by a slash (/) before its name. Navigational classes and associations with navigability are graphically represented in a UML class diagram.

The navigational class model for our example is shown in Fig. 2. The first step for the construction of this model consists of determining which classes and associations of the domain model are relevant for the company's Web site. All classes and associations of the conceptual model with the exception of Customer and the associations linking Customer to Company and Customer to Project, respectively, are included in the navigational model. The customers of a company

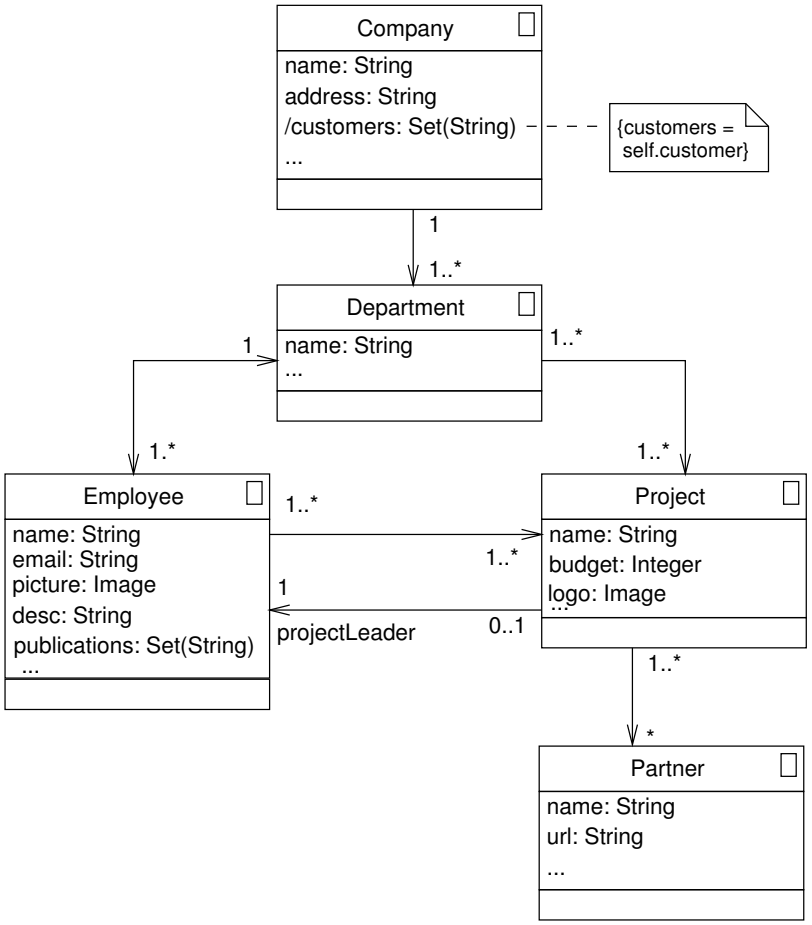

Fig. 2. Navigational Class Model of a Company's Web Site.

are only included as a derived attribute of class Company. This is to conceal the information which customer participates in which project from the visitor of the Web site.

4.2 Navigational Structure Model

The navigational structure model is based on the navigational class model. It defines the navigation structure of the application, i.e. how navigational objects are visited. Additional model elements are required to perform the navigation between navigational objects: menus, indexes, external nodes and navigational contexts. We introduce the concept navigational nodes to denote navigational objects as well as the above mentioned model elements.

Navigational Context A navigational context (context for short) consists of a sequence of navigational nodes. It includes the definition of links that connect each navigational node belonging to a navigational context to the previous and to the next navigational node within the context. In addition, links to the first and to the last, as well as circular navigation may be defined. This concept was introduced by OOHDM to permit different groupings of the navigational objects. This way a navigational object can be navigated in different contexts. For example, the information about the "Forsoft" project is shown as one of the "projects of the department R&D" and one of the "projects of the employee Baumeister".

A navigational context is depicted as an object with stereotype «navigational context» and has associated an OCL-expression defining its sequence of navigational nodes. Navigation is performed within a navigational context, but navigational context changes are possible. Continuing with the company's Web site example, if the "Forsoft" project is visited in the context of "projects of R&D", it is possible to continue navigation with other projects the same employee has been working at.

We distinguish between navigational contexts (simple contexts), grouped contexts and filtered contexts (cf. Fig. 3). An example of a simple navigational context is "all projects" (projects by name). A grouped context is a sequence of sequences of navigational nodes, such as projects by department denoting a sequence of contexts, each of them are the projects of one department. It represents a partition of a sequence of navigational objects into sub-sequences given by a common value of an attribute or a common object related by an association. As the notation for grouped context we use the stereotype «grouped context» and write Class by attribute/rolename/classname inside the box.

A filtered context («filtered context») allows a dynamic selection of a collection of elements from a navigational context that satisfies a property. This property is supplied usually by the user in a query which is part of the filtered context. An example of a filtered context is the result of the query: "employees that have been working at the Forsoft project since 1997".

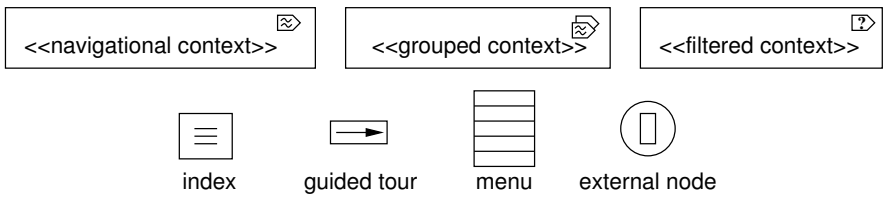

Fig. 3. Stereotypes for the Navigational Structure Model.

We group navigational contexts for a same navigational class in a UML package (cf. Fig. 4). Navigational contexts are related by special associations which allow for context changes. This is possible since the same object is part of differ-

ent sequences (navigational contexts). A stereotyped association is defined; we call it «change». It permits navigation into another context and to return to the starting point before the navigational context was changed. The possibility to change from one context to another within a package is the default semantics for a package of contexts when the changes are not explicitly drawn. In case only certain context changes are planned, a diagram of the package must show which context changes are allowed. As shown in Fig. 4, only a context change from project by department to project by employee is permitted.

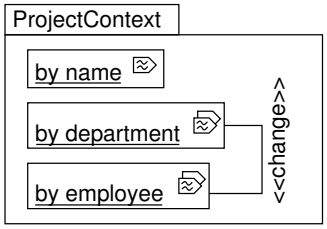

Fig. 4. Package of Navigational Contexts.

Access Primitives To define a navigational model it is necessary to specify how navigational contexts are accessed. The model elements defined for this access (called access primitives) are: guided tours, indexes and menus.

- An index allows direct access to each element within a navigational context. Usually an index comprises a list of descriptions of the nodes of the navigational context, from which the user selects one. These descriptions are the starting points of the navigation.
- A guided tour gives access to the first object of a navigational context. Objects are navigated then sequentially. Guided tours may be controlled by the user or by the system.
- A menu is an index on a navigational context of a set of navigational nodes. Every hypermedia application has at least one entry point or initial node, called the main menu (start page).

Stereotyped classes «index», «guided tour» and «menu» are defined for the access primitives index, guided tour and menu (cf. Fig. 3).

External Node An external node («external node») is a navigational node belonging to another hypermedia application, i.e. this node is not part of the application that is being modeled. Partners are modeled as external nodes in our example.

Navigational Structure Diagram To show how these navigational contexts, access primitives and external nodes collaborate we use a UML object diagram. The navigational structure model is illustrated by the company's Web site example (cf. Fig. 5). The following navigational contexts are needed to support the navigation specified in the navigational class model: department by name, employee by name and by department as well as project by name, by department and by employee.

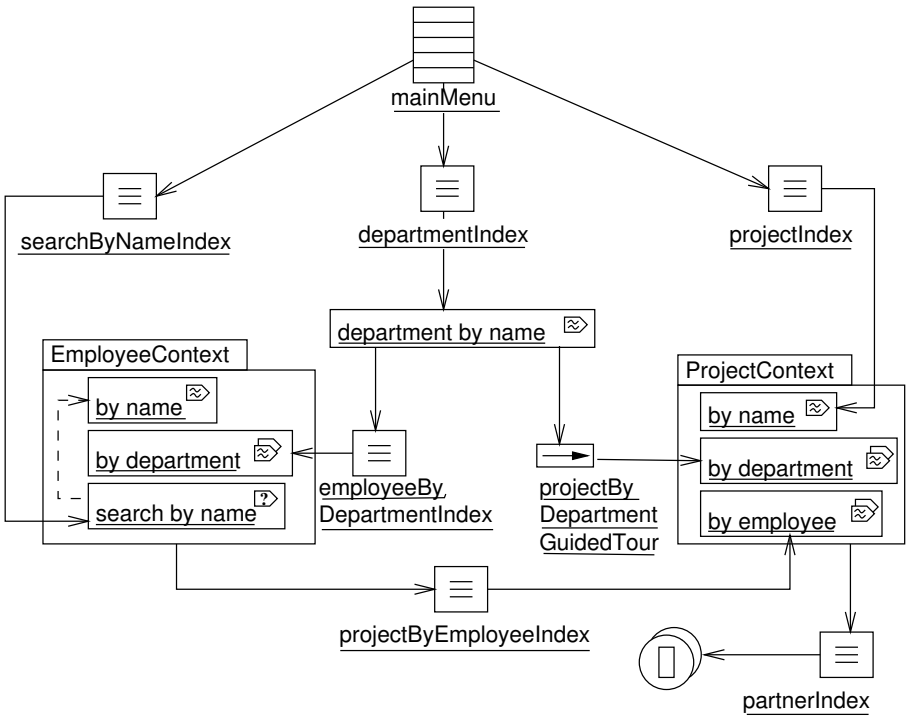

Fig. 5. Navigational Structure Model of the Company's Web Site.

The main menu is the starting point to access different navigational contexts related to the defined navigational classes. Indexes are added to permit direct access to the objects of a navigational context. That is the case of departmentIndex, employeeIndex and projectIndex. A guided tour through all the projects of a department has been incorporated to show users the activities of one department.

5 Presentational Model

The next step in the process of designing a hypermedia application is to design its user interface. While the navigational design defines *what* is the navigational

structure of an application, the task of the presentational design is to define *how* this navigational structure is presented to the user. Note that the same navigational structure may yield different presentational designs depending, among other things, on the restrictions of the intended target platform and the used technology.

Most of the methods for hypermedia design suggest the development of prototypical pages for the design of the user interface. Instead, we propose first to define a presentational model as a composition of user interface components, like text, anchors, forms, buttons, videos, etc. The presentational model is a schematic user interface that provides hints on the position, color and the relative size of the user interface components, but does do not prescribe their final appearance.

This is very much in the spirit of HTML that was designed to describe the logical structure of a Web page and not its appearance in a Web browser. To actually define the appearance of the logical elements is the responsibility of the Web browser. This implies that different browsers and even the same Web browser on different platforms may present the same Web page differently.

The presentational model consists of the static presentational model and the dynamic presentational model. The static presentational model is represented by UML composition diagrams describing how the user interface is constructed from user interface components. UML statecharts are used in the dynamic presentational model to describe the behavior of these components, for example how navigation is activated and which user interface transformations will take place.

5.1 Static Presentational Model

The static presentational model defines how navigational nodes of the navigational model are presented to the user. It consists of a collection of user interface objects that are represented by UML composite objects, showing the composition of user interface objects by other user interface objects.

A user interface object can be either a primitive user interface object like text and button, or a composition of user interface objects. A special kind of composite user interface object is a presentational object that depends on the state of a navigational object. User interface objects can have their own state, like a button which can be either in state up or down, and they react to user events. Their behavior is defined in the dynamic presentational model. In Fig. 6 we present the stereotypes for the most frequently used interface objects: anchor, text, image, audio, button, form, image, video, collection and anchored collection. The user can define additional user interface objects by composition and by subclassing a user interface class. The user interface objects have the following semantics:

- An anchor is a clickable area and is the starting point of a navigation. They are presented in the literature mostly as part of links, seldom as independent objects. A similar distinction between the concept of link and anchor can be

found in the Dexter Hypertext Reference Model [HS94]. An anchor consists of a presentation, which can be a text, an image, a video, another interactive object, etc. together with a link.
- A text is a sequence of characters together with formatting information.
- A button is a clickable area that has some action associated to it, which is defined by the dynamic presentational model. Example of actions are playVideo, displayImage and stopAudio.
- A form is used to request information from the user, for example his name or a search string. A form consists of input fields, menus, checkboxes etc.
- Images, audio and video are multimedia objects. Images can be displayed; audio and video can be started, stopped, rewinded and forwarded. To provide these functionality interactive user interface objects, such as buttons or anchors may be associated to these multimedia objects.
- An external application is included in the current application, but is not related to the navigational objects of the application. An external application can be implemented for example by applets, Active-X components or embedded objects.
- The user interface objects collection and anchored collection are introduced to provide a convenient representation of a composite consisting of a set of user interface objects and a set of anchors respectively.

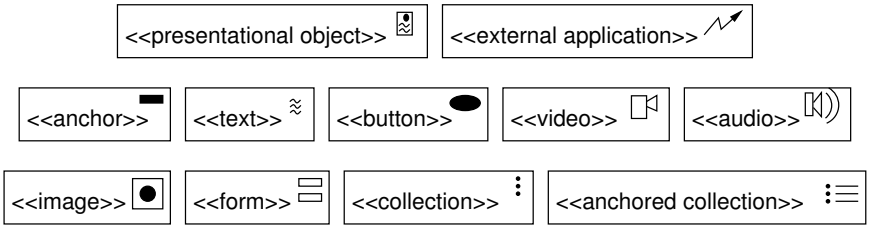

Fig. 6. Often Used User Interface Objects.

A presentational object (≪presentational object≫ cf. Fig. 6) is a user interface object that depends on the state of a navigational node and is the composition of other user interface objects including other presentational objects.

On the left side of Fig. 7 we show the presentational object for an Employee and on the right side the presentational object of the navigational context employee by name as UML composite object diagrams. The attributes of an Employee are represented by user interface objects in the presentational object employee, e.g. the picture of an employee is represented by an image. This object contains anchors linking to the projects of an employee and to the department the employee is working in.

The presentational object employee is embedded in the presentational object for the navigational context employee by name. The buttons prev and next allow to

jump to the previous and to the next element in the context, respectively, while the buttons prev by dept and next by dept permit the change to the navigational context employee by department.

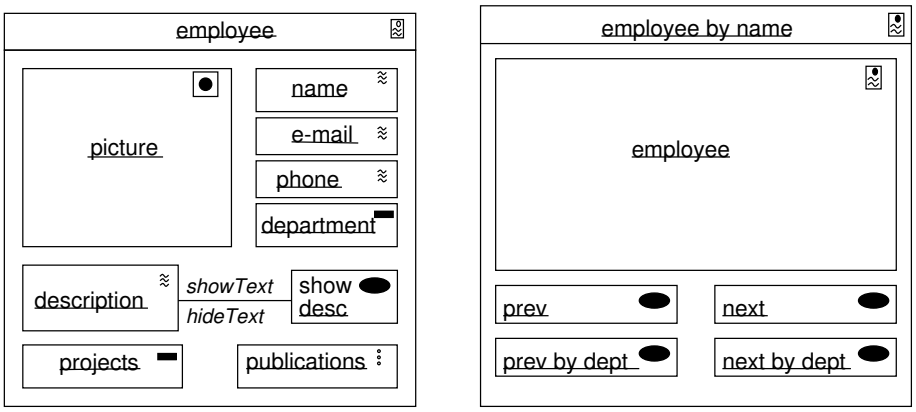

Fig. 7. Presentational Object for Navigational Class Employee and Presentational Object for Context employee by name.

5.2 Dynamic Presentational Model

The dynamic presentational model uses UML statecharts to define the reaction of the user interface objects in the static presentational model on external user events, like mouse movements, mouse clicks and keyboard presses, as well as on internal events like timeouts, activations, deactivations.

To control which user interface objects are perceptible to the user and which user interface object is active, i.e. can receive user events, we use two variables: perception and activation. The variable perception contains the list of currently perceptible user interface objects which are not part of other user interface objects. Perceptible means audible in the case of audio and visible in case of all other user interface objects. Each user interface object in the list held by perception can be thought of as being presented in its own window on the screen. Whenever an element is put into the perception the internal event show is generated and whenever an element is removed the event hide is generated.

The variable activation contains the user interface object from the list of perceptible user interface objects which is currently active, which means that it can react to user input and will receive external user interface events. Whenever a user interface object is assigned to the variable activation it will receive the internal event activate and the previous object assigned to activation, if any, will receive the internal event deactivate. For example, a video user interface object may choose to start playing when it receives activate and to stop playing when it receives deactivate.

When a user interface object receives an event it may change the variables perception and activation, generate new events and send messages to other user interface objects or navigational objects. A composite user interface object delegates the events it receives to its components. Most user interface objects have a default behavior which the designer of a hypermedia application need not specify. An exception are buttons. The dynamic presentational model has to define the actions that are performed when a button is pressed. In Fig. 8 an example is given how statecharts are used to define the behavior of the button showDescr of the presentational object employee (cf. Fig. 7) to toggle the display of the description text user interface object whenever the button is pressed.

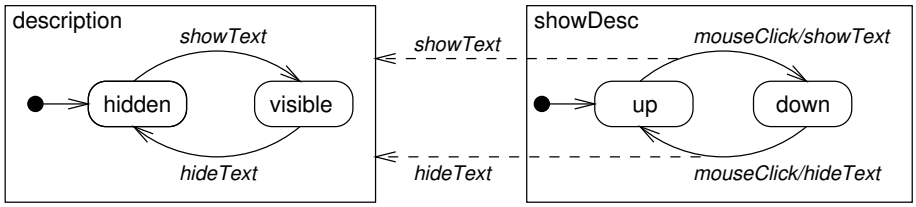

Fig. 8. UML Statechart Specifying the Behavior of the showDescr Button.

The variable perception can be changed in several ways. First, a user interface object can be added to the list of user interface objects. This implies that a new window displaying this user interface object is generated. Second, a user interface object may be removed from the list. Thus, removing the window displaying that user interface object. Third, a user interface object in the list may be replaced by another user interface object in that list. In this case, the window displaying the user interface object that was removed now displays the added user interface object. These three options are called *macro navigation*. In contrast, we call *micro navigation* the replacements of components of a user interface object in the list of perceptible user interface objects. In the context of Web design, macro navigation can be implemented by fetching a new HTML page from the server, while micro navigation can be implemented using frames, which allow to replace only part of the display.

6 Conclusions and Future Work

The UML extension of hypermedia design presented here is a model-based approach whose modeling techniques are UML diagrams and whose graphical representation only uses UML notation. Limiting to the notation proposed by the UML instead of introducing a new notation has the advantage of using a well-known standard and that UML is supported by many case tools. UML is extended to model the navigation and the presentation according to the UML extension mechanisms. These mechanisms are based on the definition of stereotypes and the use of OCL. As far as we know this is the only hypermedia design

method that is full UML compliant in every step. The central point of the design are the navigational models that aid hypermedia designer in the specification of a clear structure of hypermedia systems and improve orientation of the users in the application navigation space. The concept of navigational context and the graphical notation of the navigational class and navigational structure diagram achieve to represent complex hypermedia structures in a summarized and clear way.

In contrast to our approach, which deals with the design of a complete hypermedia application, user interface design methods focus only on the design of the user interface part of an application (cf. [Pre94]). In our approach the user interface design is captured mainly in the presentational model. However, methods and design guidelines developed in the HCI and user interface design community can be applied to the design of the presentational model, such as the guidelines for data display mentioned in [Shn98].

Our future work will concentrate its attention on refining the techniques and notations presented here. Specifically hypermedia applications that support more functionality, such as database transactions will be object of study and modeling using the proposed technique and notation. The design process described here is part of a methodology covering the whole life cycle of hypermedia systems. The description of the tasks requirement capture, analysis, implementation and test is on work. Case tools that support UML can be used for the conceptual, navigational and presentational design. We are developing an editor for the UML-based hypermedia design that not only adds the defined stereotypes as model elements, but also includes generation rules for the navigational model based on the conceptual model as well as for the presentational model based on the navigational model.

References

[BBI96] V. Balasubramanian, M. Bieber, and T. Isakowitz. Systematic hypermedia design. Technical report, CRIS Working Papers series. Stern School of Business, New York University, 1996.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language: User Guide*. Addison Wesley, 1999.

[Con98] J. Conallen. Modeling Web application design with UML, 1998. Available at `http://www.conallen.com/ModelingWebApplications.html`.

[HS94] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, February 1994.

[ISB95] T. Isakowitz, E. Stohr, and P. Balasubramanian. A methodology for the design of structured hypermedia applications. *Communications of the ACM*, 8(38), 1995.

[KM99] N. Koch and L. Mandel. Using UML to design hypermedia applications. Technical Report 9901, Institut für Informatik, Ludwig-Maximilians-Universität, München, March 1999.

[Lan96] D. Lange. An object-oriented design approach for developing hypermedia information systems. *Journal of Organizational Computing and Electronic Commerce*, 6(3):269–293, 1996.

[Pre94] Jenny Preece. *Human-Computer Interaction*. Addison-Wesley, 1994.

[Ros96] G. Rossi. *OOHDM: Object-Oriented Hypermedia Design Method (in portuguese)*. PhD thesis, PUC-Rio, Brasil, 1996.

[Shn98] Ben Shneiderman. *Designing the User Interface*. Addison-Wesley, 1998.

[SR98] D. Schwabe and G. Rossi. Developing hypermedia applications using OOHDM. In *Workshop on Hypermedia Development Process, Methods and Models, Hypertext'98*, 1998.

[SRB96] D. Schwabe, G. Rossi, and S. Barbosa. Systematic hypermedia design with OOHDM. In *Proceedings of the ACM International Conference on Hypertext (Hypertext'96)*, 1996.

[TL97] O. De Troyer and C. Leune. WSDM: a user-centered design method for Web sites. In *Proceedings of the 7th Internatinal World Wide Web Conference*, 1997.

[WK99] J. Warmer and A. Kleppe. *The Object Constraint Language*. Addison Wesley, 1999.