

Using UML to Design Hypermedia Applications*

Nora Koch^{1,2}, Luis Mandel²

¹Ludwig-Maximilians-Universität München
Institut für Informatik
Oettingenstr. 67
D-80538 München
Tel. +49 89 2178 2151
Fax +49 89 2178 2152
kochn@informatik.uni-muenchen.de

²Forschungsinstitut für Angewandte Softwaretechnologie (FAST e.V.)
Arabellastr. 17
D-81925 München
Tel. +49 89 92004739
Fax +49 89 92004718
koch@fast.de, mandel@fast.de

Abstract

This paper presents a UML-based design of hypermedia systems. It comprises the conceptual, navigational and presentational modeling of hypermedia applications. These design steps are similar to those proposed in other methods for hypermedia and Web design. As most of these methods it is based on the separation of contents, structure and layout. The novelty of this approach consists in the modeling techniques and notation used that are entirely based on the Unified Modeling Language (UML). Thus, a UML extension is defined for the navigational and presentational design using the general extensions mechanisms provided by the standard Unified Modeling Language: Stereotypes and the Object Constraint Language (OCL).

Keywords

UML, hypermedia systems, design method, Web applications, UML extension, structured navigation.

1 Introduction

Hypermedia applications provide users with information and navigation facilities to access to information based systems. In addition hypermedia systems may support in some cases functionality to manipulate this information. The development of hypermedia applications requires abilities to choose multimedia contents, to define an adequate structure, to design the user interface and to select the appropriate implementation technique. Ad hoc building processes focus on the contents and the presentation paying little attention to the navigational structure. Quality of a hypermedia application depends not only on the richness of contents and on high-quality graphic design, but also needs a well-structured navigation.

* Technical Report 9901, Ludwig-Maximilians-Universität München, March 1999.
This work was partially supported by the Bayerische Forschungsstiftung.

The design process described in this paper is a model-based approach for building hypermedia applications. It treats contents, structure and presentation on the same terms. This process makes a clear distinction between the conceptual design of the domain, the design of hypermedia navigational structure and the visual representation of the information, in similar way as it is proposed by other hypermedia design methods, such as OOHDM [ScRoBa96], EORM [Lange96] and WSDM [DeTroLeu97].

The main objective of the conceptual design is to capture the domain semantics and present it as an object-oriented structural model. The navigational design defines the structure of the hypermedia application describing how navigation can take place. The static and dynamic aspects of the user interface are modelled during the presentational design depicting the layout in a schematic way. The design of hypermedia applications is an incremental, iterative and prototyped-based process. We consider that the development of a prototype plays an important role in the design of hypermedia applications. But to detail implementations aspects of a prototype is not within the scope of this paper.

Each of the above mentioned methods uses existent object-oriented models for the conceptual design (OMT, UML) but introduces an own notation and diagrammatic techniques for navigational and/or user interface design. The models we propose, are based on the standard Unified Modeling Language (UML), which are built with well-known UML model elements and UML extensions. These extensions are defined following the UML extending mechanisms [Booch99, UML97]. A UML compliant method has two important advantages: First, documentation about syntax and semantics of the UML modeling elements is already available. Second and the most important one, case tools supporting UML can be used in the hypermedia development process.

The scope of this paper is to show how *UML notation* and *UML modeling techniques* can be used in the hypermedia design phase. We outline the objectives of each phase focusing on these two aspects: modeling technique and notation. UML stereotypes are specified for navigational and presentational model elements, according to the mechanisms for standard extensions defined in the UML.

This paper is structured as follows: Section two outlines some related work. Section three gives an overview of the phases of the hypermedia design method. Sections four, five and six describe the conceptual, navigational and presentational modeling approach of the design process respectively. In Section seven we present some concluding remarks and an overview of future work.

2 Related Work

In recent years many methods have been developed for the design of hypermedia applications. All of them describe a sequence of steps particularly for the design process; the approaches are either object-oriented or based on the entity-relationship model.

The **Relationship Management Methodology** (RMM) addresses the design and construction of hypermedia application through a process of seven steps [IsStBa95]. RMM is at the same time a top-down and a bottom-up approach. These steps are: E-R design, slice design, navigational design, user interface design, protocol conversion design, run-time behaviour, and construction and testing. An application design is represented with an RMDM

(Relationship Management Data Model) based on the entity-relationship model and on the Hypermedia Design Model (HDM of [GaMaPa96]).

During the E-R design step entities and relationships are identified. They 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. The navigational design is the step for the identification of the paths that enable hypertext navigation. RMDM specifies navigation through access primitives: link, grouping (menus), index and guided tour. The conversion protocol design converts components of RMD and E-R into physical objects in the target hypermedia application. The step user interface design involves the design of the screen layouts of every model element. These elements are: access primitives, link anchors, indexes and general navigational aid. The techniques proposed for the user interface design are elaboration of mock-ups and prototyping [BaBieIsa96].

The **Enhanced Object-Relationship Model (EORM)** is defined as an iterative process concentrating in the enrichment of the object-oriented model by the representation of relations between objects (links) as objects. According to [Lange96], this representation 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 proposes as well the construction of a prototype of the user interface at an early stage during design.

This method is based on three frameworks: class, composition and GUI. The class framework consists of a reusable library of class definitions. To identify classes for an application EORM follows standard object-oriented techniques. The result is represented with OMT notation. The composition framework consists of a reusable library of link class definitions. The last step is the design of the GUI application using elements of the GUI framework. Thus, this method proposes as main steps: to determine the windows of the domain and which presentation has to appear in each window, to obtain presentations from attributes and operations of classes and determine how functionality is assigned to window menus.

The **Object-Oriented Design Method (OOHDM)** comprises four activities; they are conceptual modeling, navigational design, abstract interface design and implementation [Rossi96, ScRoBa96]. These activities are performed in a mix of incremental, iterative and prototype-based development style. Object-oriented models are built in each step improving the models design in previous iterations.

The conceptual model of the application is represented with a class diagram. This method sees an application as a view over the conceptual model. The concept of navigational class and navigational context are introduced to describe navigational structures. Navigational context is a powerful concept that allows different groupings of navigational objects with the purpose to navigate them in different contexts. The access to these navigational elements is modelled with access structures, such as indexes and guided tours. Different types of indexes and navigational contexts are defined in this step and a special notation is used for the graphical representation of the navigational schema. Some difficulties arise using this method as a detailed description of the semantics of the model elements is missed. The abstract interface model is the result of the specification of the interface objects the user will perceive. Static and dynamic aspects of the user interface are modelled with ADVs and StateCharts

[CaCoLu93, Harel87]. ADV (Abstract Data View) notation is used for the graphical representation.

OOHDM does not prescribe any particular notation for the diagrammatic representation of the conceptual schema; in earlier papers OMT was proposed [ScRoBa96], in a latter one they replaced it by a UML [ScRo98] notation. However, OOHDM it is not UML compliant as it uses an own notation for the so called perspectives for attributes in the class diagrams and proposes other kind of diagrams (navigational context schema, configuration diagram, ADV charts) for the navigational and abstract user interface design.

The **Web Site Design Method** (WSDM) is a user-centred approach defining the navigation objects based on the information requirements of the users of a Web application [DeTroLeu97]. 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. Starting point is the description of the domain taken into account the user activities. Different perspectives are defined for the user classes; these are different ways user classes look at the same information. Conceptual design consists of two steps: object modeling and navigational design. Object modeling is done in three steps: business object modeling, user object modeling and perspective object modeling.

The navigational model consists of a number of navigation tracks, one for each perspective expressing how users of a particular perspective can navigate through the available information. WSDM describes it in terms of components and links. It distinguishes three types of components: navigation, information and external. Each navigation track consist of three layers: context, navigation and information layers. The context layer is the top level of the navigation track starting with a navigation component. The information layer is the bottom level of the navigation track. The navigation layer connects the context layer and the information layer. To access the information intermediate components and links are created, such as indexes. This method uses its own defined graphical notation for the navigational model objects.

This kind of navigational design achieves Web applications that have a very hierarchical structure. The implementation design step achieves to create a consistent and efficient look and feel for the conceptual model. Some recommendations are given in this step, such as the use of index pages, information divided into right-sized chunks, use of context and information cues and use of navigational cues.

Another method that was recently developed is the **Scenario-based Object-oriented Hypermedia Design Methodology** [LeLeYo98]. It consists of six phases: domain analysis, object modeling, view design, navigation design, implementation design and construction. This methodology has many similarities with RMM, OOHDM and EORM. It differs in the use of scenarios, which are described through scenario activity charts. These scenarios are defined in the domain analysis phase and are used for the object modeling. View design consists in determine object-oriented views generated from single object classes or from associations between object classes. The navigation design uses scenarios to determine access structure nodes. They are defined as a menu-like mechanism that enables users to access to other parts of hypermedia documents. This access structures together with the object-oriented views are called navigation units. The identification of navigation links complete the navigational design. During implementation design phase the user interface, pages and a

logical database schema are modelled. This method presents a clear sequence of steps that benefits of the scenarios obtained as results of the analysis phase. It defines its own graphical notation.

A set of UML stereotypes for Web modeling are presented by **Conallen** [Cona98]. It defines components, class, method and associations stereotypes, such as server component, client component, server page, client page, form, frameset, target, server method, client method, links, redirects, builds, targeted links, submit. These stereotypes are appropriate to model layout and implementation aspects, but not to define the navigation structure of the Web application. An interesting Web architecture is presented in this work, but it does not define a method for the design of Web applications.

3 Designing Hypermedia Systems

The goal of our method is to present a UML-based approach to design hypermedia applications. It is a systematic approach based on the construction of models. These models are built with UML standard model elements or model elements defined according to the standard UML extension mechanisms. The design comprises of three steps:

- *Conceptual design*
- *Navigational design*
- *Presentational design*

The *conceptual design* produces a conceptual model of the problem domain defined through classes and associations between classes relevant to the domain. The other two steps take into account the special characteristics of the hypermedia paradigm: the navigational functionality and the multimedia user interface.

The basis of the *navigational design* is the conceptual model and the outcome is a navigational model, which can be seen as a view over the conceptual model. The navigational model is defined in a two step process. In the first step it is specified, *which* objects can be potentially reached through navigation and in the second one *how* these objects are reached. Therefore, additional objects are utilised to access navigational objects and a grouping concept is required to achieve an optimal navigation.

The task of the *presentational design* is to define the presentation of the navigational objects identified in the previous step. The outcome of the presentational design is the static and dynamic presentational model. While static presentational model associates to each navigational object at least one presentational object, the dynamic presentational model describes the behaviour of these presentational objects.

With these three steps a clear separation of contents, navigation and presentation is provided. Rossi and Schwabe stress that a consequence of this separation is a more modular and reusable design and a framework to reason about the design process encapsulating design experience specific to each step [ScRoBa96].

The design process is illustrated with a Web system called *Film Assistant*. The goal of this application is to offer information about films, persons working in the film business, such as actors, actresses, directors, producers, cinematographers, costume designers, editors, composers, etc. as well as about festivals and awards. Users are informed about news related

to the film business. Film recommendations are provided to the users in response to criteria they selected or entered in a form.

Films are described through a large list of attributes including title, year, language, rating, genre, country, crew, distributor, length, book, keywords, etc. A film genre may have the following values: action, thriller, comedy, documentary, etc. A film festival consists of film presentations, nomination of films prizes in different categories (best film, best actor, best production, best music, etc.) and the determination of the winners of the awards. Thus, films and persons may be honoured with awards of type nomination or winner. News are associated to films and persons; they include articles and comments. A Web page of the resulting Web system is shown in Fig. 1.

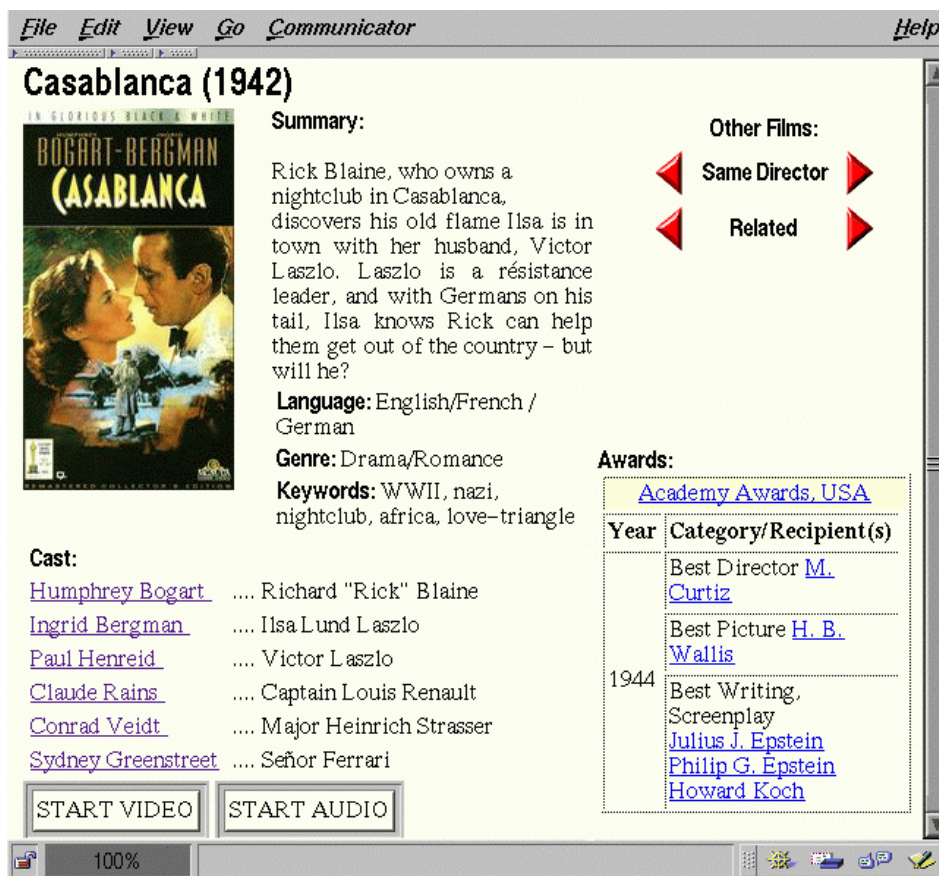


Figure 1: The Film Assistant

4 Conceptual Design

The result of the analysis step is the *conceptual model* of the problem domain. It includes all the concepts that are relevant for the application and for the different users or user groups that have been identified. Therefore, pre-condition for an appropriate conceptual design is a careful analysis, which determines use cases and scenarios [Jacobs92] for the user typical navigation activities.

The main objective of the conceptual design is to capture the domain semantics with as little concern as possible of the navigational and presentational aspects. Decisions as whether each

concept corresponds to one hypermedia page, a hypermedia document or if the page is generated on-the-fly are postponed to the implementation phase.

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 composition, generalisation and specialisation are used to achieve this purpose. Classes and associations defined in this step are used during navigational design to derive nodes of the hypermedia structure. Relationships will be used to derive links. Classes and associations can be organised in groups. We use UML packages for these groups. Classes are described through attributes and operations and represented graphically with UML notation [UML97, Booch99] as depicted in Fig. 2.

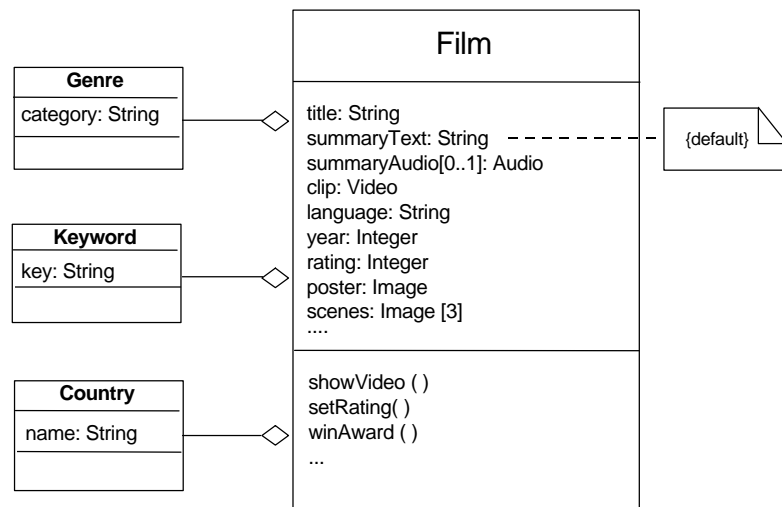


Figure 2: Class Film and aggregated classes Genre, Keyword and Country

The example of Fig. 2 shows also how attributes that have more than one type can be specified using UML notation. This is the case of summary: attributes summaryText and summaryAudio are then defined. The default type for summary is String.

The result of the conceptual design step is summarised in a conceptual model that consists of classes and associations between classes modeling the problem domain. It is represented by a UML class diagram [UML97, Booch99]. UML object diagrams may be used for objects that require a special treatment.

In the *Film Assistant* the following classes have been identified to model the domain: Film, Festival, Person, Award, News, Genre, Country and Keyword. Persons is a class that groups actors, actresses, directors, producers, cinematographers, costume designers, editors, composers, etc. Genre are different film categories, such as drama, comedy, science fiction and thriller. Different types of relationships between classes are identified, such as the class Film related through an association "isHonouredWith" to the class Award and through "bornIn" to the class named Country. Class Award "is part of" class Festival (composition) and the relationship between Film and Person is of type composition. The Conceptual Model of the *Film Assistant* is depicted in Fig. 3.

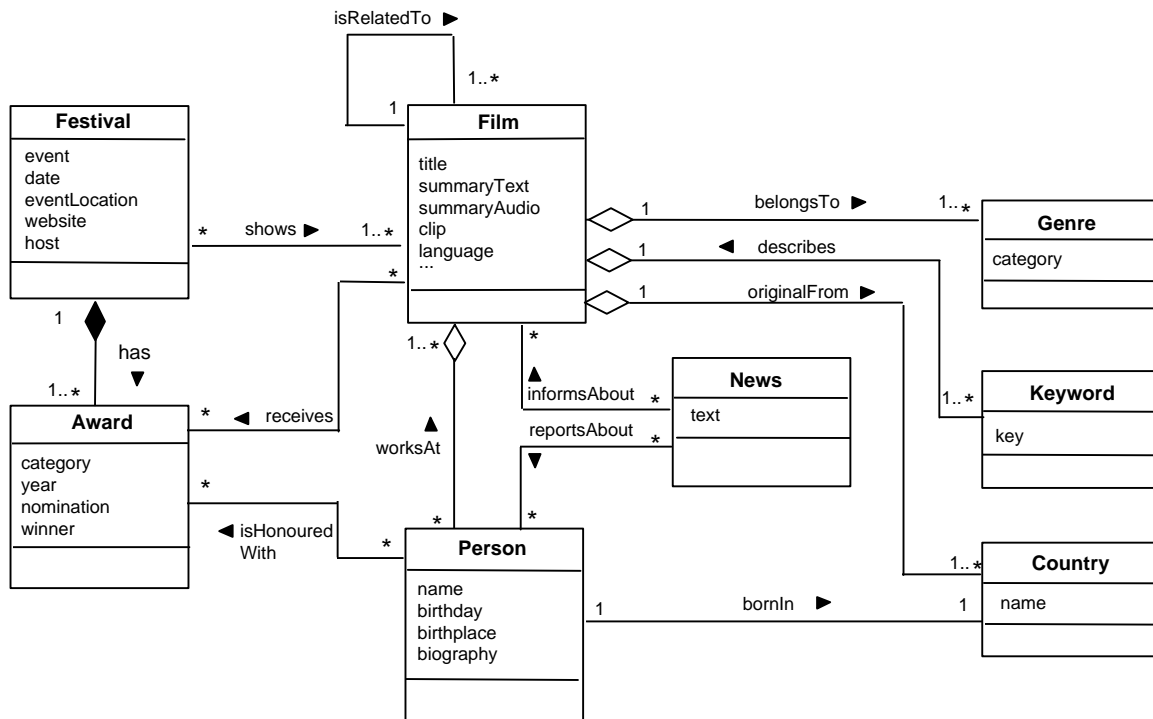


Figure 3: Conceptual Model of the Film Assistant

This conceptual model may be used as the starting point to design more than one navigational design, i.e. more than one hypermedia application.

5 Navigational Design

Navigational design is a critical step in the design of every hypermedia application. Even simple applications with a non-deep hierarchical structure will become complex very soon by the addition of new links. Additional links improve navigability on the one hand but imply on the other hand higher risk to lose orientation. Building a navigational model not only is helpful for the documentation of the application structure, it also allows for a more structured increase of navigability.

The *navigational design* defines the structure of the hypermedia application. Two models are built in this steps: the *navigational class model* and the *navigational structure model*.

An hypermedia application is organised into nodes and links that establish relationships of type navigation between the nodes. *Navigational nodes* that are obtained from objects of the conceptual model are called *navigational objects*.

5.1 Navigational Class Model

The navigational class model defines a view [ScRoBa96] 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 between these navigational classes. Classes and relationships 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 shown through the arrow attached to the end of the association's line. 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*» (Fig. 4) with same name as the corresponding class of the conceptual model. *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* can be seen as a sub-graph of the conceptual model where some classes which are not relevant for the navigation are eliminated and/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 [WaKI99]. A derived attribute is denoted in UML by a slash(/) before its name. Navigational classes then differ from conceptual classes in attributes and methods that are added or eliminated.

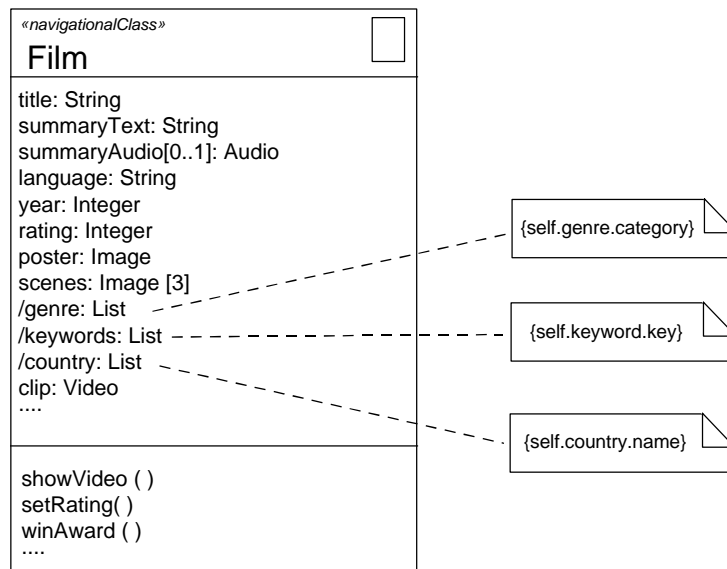


Figure 4: Stereotyped Navigational Class Film

Navigational classes and associations with navigability are graphically represented in a UML class diagram. The resulting *navigational class model* for the *Film Assistant* example is shown in Fig. 5.

First step for the construction of this model consists of determining which classes of the domain model are relevant as nodes for the hypermedia application *Film Assistant*. All classes of the conceptual model with exception of Genre, Keyword and Country are included in the navigational model. These three classes are not relevant for the navigation as they will not have pages associated to them. Therefore, Genre, Keyword and Country are incorporated as derived attributes in the navigational class Film.

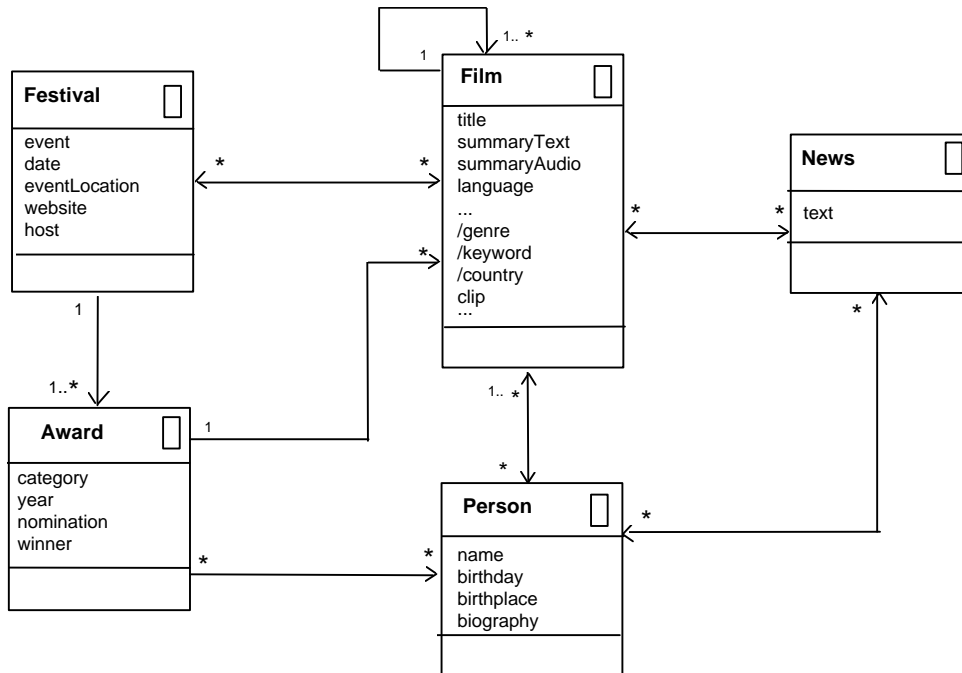


Figure 5: Navigational Class Model of Film Assistant

5.2 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 of *navigational node* to refer to navigational objects as well as the above mentioned model elements.

5.2.1 Navigational context

A *navigational context* (context for short) consists of a sequence of navigational nodes. This concept was introduced by OOHDM [Rossi96, ScRoBa96] 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 film "Casablanca" is shown as one of the "films of the year 1942" and one of the "films with Humphrey Bogart".

The set of navigational nodes belonging to a navigational context are usually connected through an aspect, such as being objects of a given class or being related to another class through an association. Navigational contexts allow the organisation of the navigational space in sequences that can be traversed following a predefined order (Fig. 6). They include the definition of links that connect each navigational node belonging to a navigational context to the previous and to the next navigational node. In addition, links to the first and to the last as well as circular navigation may be defined. Navigational contexts may be nested.

Navigation is performed within a navigational context, but navigational context changes are possible. Continuing with film examples, if "Casablanca" is visited in the context of "films with Humphrey Bogart", it is possible to continue with other "films of the same genre" or "films of the same year".

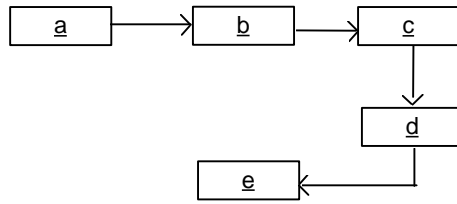


Figure 6: Navigational Context

OOHDM defines six different contexts: simple class derived, class derived group, simple link derived, link derived group, arbitrary and dynamic [ScRoBa96]. We only distinguish between *navigational context* (simple context), *grouped context* and *filtered context*. An example of a simple navigational context is „all festivals“ (festivals by event). A *grouped context* is a sequence of sequences of navigational nodes, such as „films by actor“ (denotes a sequence of actors, each of them has played roles in a sequence of films).

A *filtered context* allows a dynamic selection of a collection of elements from a navigational context satisfying a property. This property is supplied usually by the user in a query. An example of a filtered context is the result of the query: "all films nominated for the category best original music in 1998".

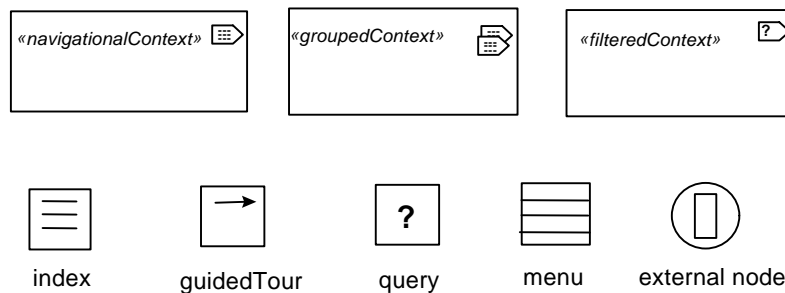


Figure 7: Stereotypes for the Navigational Structure Model

We define stereotyped classes *«navigationalContext»*, *«groupedContext»* and *«filteredContext»* (Fig. 7), which have associated an OCL-expression defining the sequence of navigational nodes.

Navigational contexts for a same navigational class are grouped in a UML package. Navigational contexts are related through special associations which allow for context changes. This is possible since the same object is part of different sequences (navigational contexts). A stereotyped association is defined; we call it *«change»*. It permits navigation in another context and to return to the starting point before the navigational context was changed. For example in the *Film Assistant* it is possible to change from any context of class Film to the context News by film, which will show all news for the given film.

Fig. 8 shows the package of navigational contexts for the Film navigational objects and the dependency relationship between this package and the navigational objects, which are depicted with UML multi-object notation. The possibility to change from one context to another within a package is the default semantics for a package of contexts. In such a case changes need not explicitly be drawn. When only some changes are allowed, a diagram of the package must show the permitted context changes.

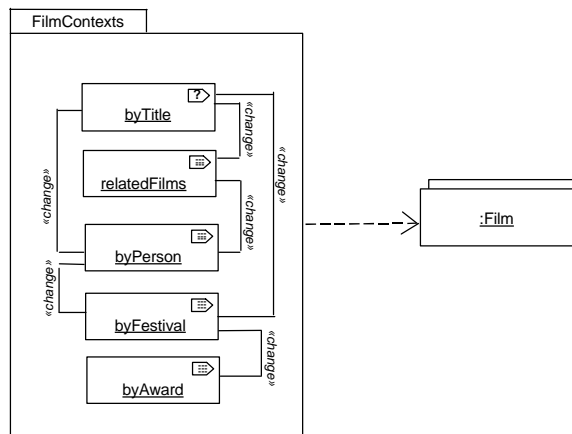


Figure 8: Package of Navigational Contexts.

5.2.2 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*, *queries* 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. Sometimes navigation to the neighbours (next, previous) is suppressed.
- A *guided tour* gives access to the first object of a navigational context. Objects are navigated then sequentially (Fig. 9). Guided tours may be controlled by the user or by the system.
- A *query* is an input form; when evaluated it produces a filtered context.
- 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, so called main menu (homepage).

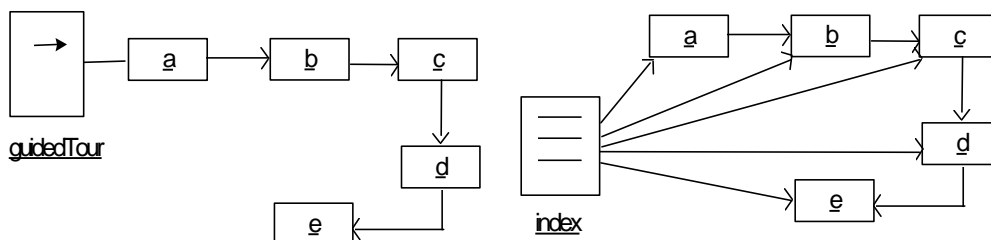


Figure 9: Guided Tour and Index for a Navigational Context

Stereotyped classes «*index*», «*guidedTour*», «*query*», and «*menu*» are defined for index, guided tour, query and menu. A graphical notation for the access primitives is depicted in Fig. 7 and their semantic is given in Fig. 9.

5.2.3 External Node

An *external node* is a navigational node belonging to another hypermedia application, i.e. this node is not part of the application that is being modelled.

No external node is included in our example. If we had included film production companies, these navigational objects would have been external nodes in case these companies have their own Web site.

5.2.4 Navigational Structure Diagram

To show how these navigational contexts, access primitives and external nodes collaborate we use a UML object diagram. The activities that are performed to pass from the navigational class model to the navigational structure model are the following:

- *defining navigational contexts.* For each navigational class at least one navigational context has to be defined. A simple navigational context is required to access to the context from a menu through an index. A filtered context is appropriate when a query is used to reach the context. For each association with navigability and multiplicity "zero or more" or "one or more" a context "by the source class" has to be included. These navigational contexts are grouped into a UML package for graphical representation's purposes only (Fig. 8).
- *determining context changes.* At the first glance all navigational context changes are allowed. Context changes are added to the model in function of the typical navigation activities of the user that were determined with a use case or a scenario based analysis.
- *adding access primitives.* One type of access to navigational contexts has to be chosen in each case. Context changes usually require the definition of additional indexes.

5.2.5 Example

We use the *Film Assistant* example to illustrate the navigational structure model. Given the navigational classes and associations defined in the navigational class modeling step and according to the first activity mentioned above, the following are some of the navigational context needed: film by title, related films, film by person, person by film, awards by category, news by date, news by film.

Each navigational context consisting of more than one navigational node requires at least one access primitive. A 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 index of Films, index of Persons, index of Festivals and index of News. A guided tour through recommended films is incorporated to assist users trying to select a Film they have not seen yet based on some user preferences. Queries to find films and persons according to users input or choices in a pull down menu are also possible.

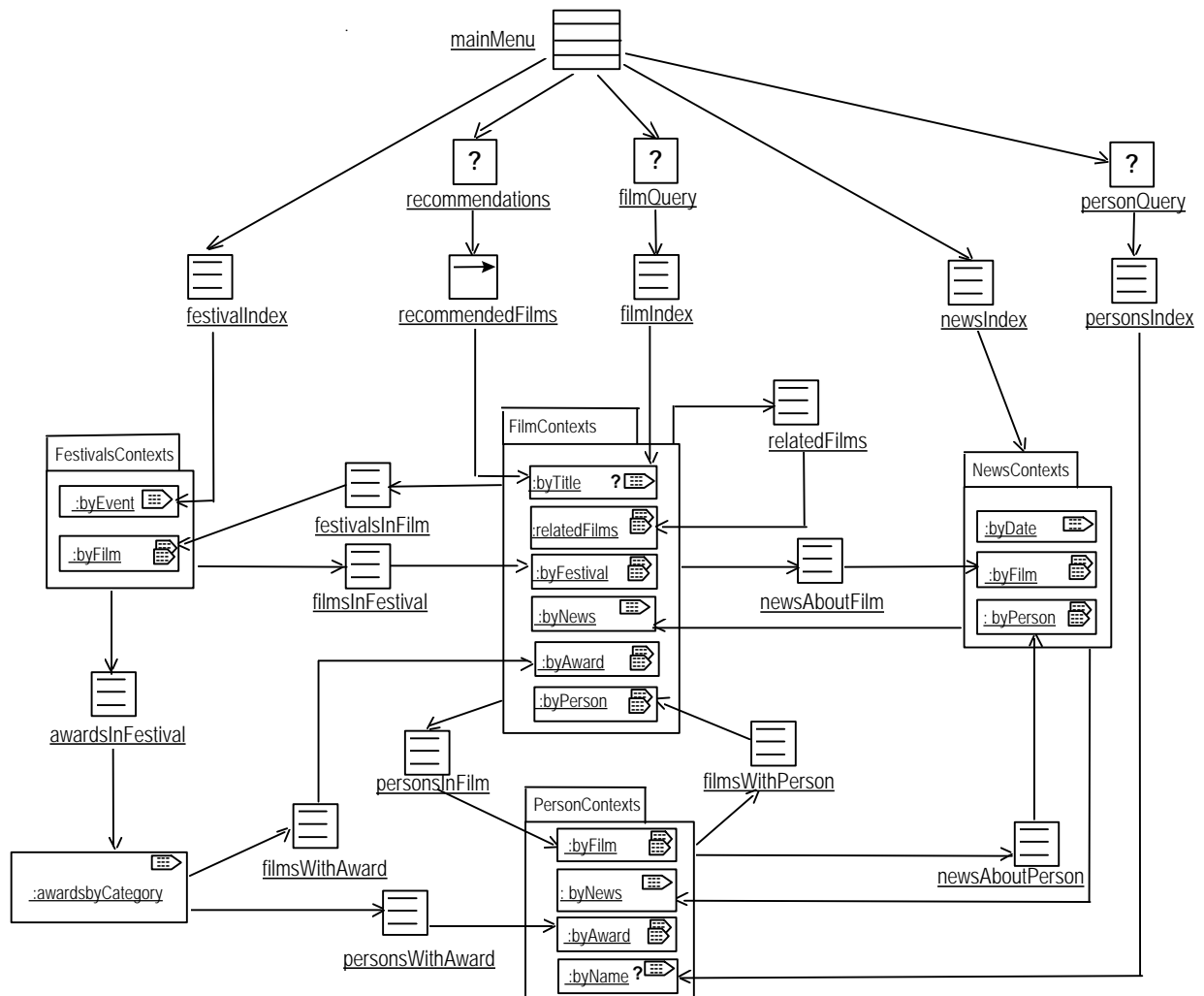


Figure 10: Navigational Structure Model of the Film Assistant

Fig. 10 shows UML object diagram for the navigational structure model of the *Film Assistant*. Navigational models aid hypermedia designer in the specification of a clear structure of hypermedia systems and improve orientation of the users in the application hyperspace. The concept *navigational context* and the graphical notation of the navigational class and navigational structure diagram achieve to represent complex hypermedia structures in a summarised and clear way.

6 Presentational Design

The presentational design supports the modeling of an *abstract user interface*, showing how the navigational structure is presented to the user. To achieve this purpose a static and a dynamic presentational model is built. Presentational design means defining the way how navigational nodes will appear, selecting user interface objects to activate navigation, and determining which interface transformations will take place. The same navigational structure may yield different presentations depending on the restrictions of the target platform and the technology used.

Most of the methods for hypermedia design suggest the development of prototypical pages for this step with exception of OOHDM and SOHDM. The first one utilises the ADV notation [CaCoLu93]. The second one represents user interfaces with a graphical notation

based on a set of own predefined components such as Button, List and Image [LeLeYo98]. Instead, we propose first to define a presentational model as a composition of user interface objects.

We choose UML model elements and UML diagrams as technique in the same way we have done it for the navigational model. The presentational model is a rough design of the user interface; decisions about details such as size, colour or font of user interface elements are taken when developing the prototype or in the implementation phase. However, the layout of user interface elements in the static presentational model may provide hints, for example, on the position and the size of the user interface elements relative to each other.

For each navigational node at least one presentational object has to be defined. If the presentation of an object depends on the navigational context within the navigational object is visited, one presentational object for each context has to be specified.

The presentational model consists of the static presentational model and the dynamic presentational model. The first one is represented by UML composition diagrams that describe how the user interface are built. The second one, is represented by UML state diagrams describing the behaviour of the components, i.e. how navigation is activated and which user interface transformations take place.

6.1 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 and relationships between these user interface objects. A user interface object can be either a primitive user interface object like text, image and button, or a composition of user interface objects. A presentational object is a special kind of composite user interface object based on a navigational object.

For the most frequently used user interface objects we define stereotypes according to the extension mechanism provided by UML. These user interface objects are: *anchor*, *text*, *image*, *audio*, *video*, *form*, *button*, *collection* and *anchored collection* (see Fig. 11).

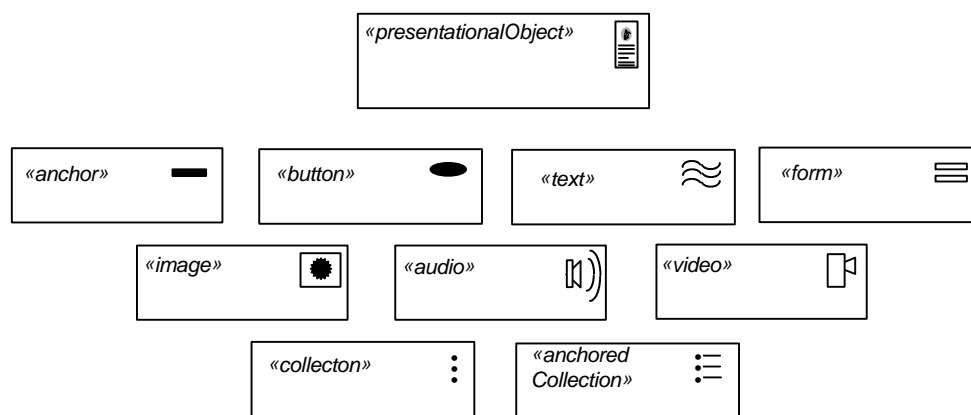


Figure 11: Stereotyped for Presentational Object and User Interface Objects

The user interface objects have the following semantics:

- An *anchor* is a clickable area, which is the starting point of a navigation establishing this way the relationship to other nodes. There are mostly presented in the literature as part of links, seldom as independent objects. An anchor consists of a presentation together with a link. The presentation may be either a text (even a single character), an image, a video, a group of mixed media, any interactive object or a whole document.
- A *text* is a sequence of characters together with formatting information.
- A *button* is a clickable area, which has an action associated to it. Example of actions are playVideo, displayImage, stopAudio and runApplet. Note that anchors may be implemented as buttons, that means that buttons can also be trigger of navigation.
- A *form* is used to request information from the user. They supply information in one or more input fields or select options from a browser or checkbox. The semantics of this model element includes the display of the content, the waiting for the user activity, the evaluation of the input and the trigger of the defined event.
- *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 user interface objects that allow for interaction, such as buttons or anchors may be associated to these multimedia objects.
- *Collections* and *anchored collections* are model elements introduced to provide a convenient representation of composites that are frequently used. It avoids the textual description by comprehension or by extension. A collection consists of a set of user interface objects. An anchored collection comprises a set of anchors. It is not specified whether the collection will laid out horizontally or vertically; objects still may be arranged as a table.

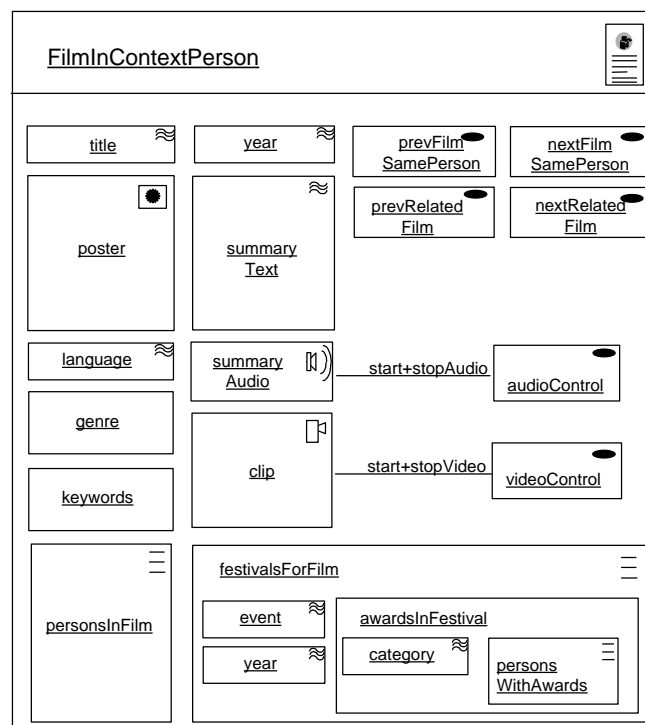


Figure 12: UML Composition for the Film Presentational Object

A static presentational model for the *Film Assistant* consists of a set of UML compositions of presentational objects (see Fig. 12). The buttons `audioControl` and `videoControl` permit the start and stop of the `summaryAudio` and the `clip` respectively.

The following activities are performed to build the static presentational model based on the navigational structure model:

- At least one presentational object is defined for each navigational node.
- A presentational object is shown as composition of:
 - objects obtained from the attributes of the corresponding navigational node,
 - interactive objects to control audio and video,
 - anchors or buttons to allow for navigation in the contexts defined for the navigational class origin of this presentational class, and
 - indexes or guided tours to access to contexts defined for other navigational classes.

6.2 Dynamic Presentational Model

The dynamic presentational model describes the behaviour of the presentational objects, i.e. the changes on the user interface when the user interacts with it or when the system reacts to internal events such as timeouts.

Each presentational object may be active or inactive; on the other hand it may be perceptible for the user or not be perceptible. Perceptible means audible in the case of audio and visible in case of all other user interface objects. Only one user interface object can be active at the same time. We use two variables, which contain the active user interface object and the list of the currently perceptible objects, respectively. Whenever an element is added to the *perception* variable the internal event *show* is generated and whenever it is removed the event *hide* is generated.

The contents of these variables are changed when an action is triggered by a transition in the dynamic presentational model. We can distinguish between macro navigation and micro navigation according to the total or partial replacement of the user interface object. In the context of Web design, for example, 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.

UML state diagrams have been chosen to depict the dynamic presentation model. The behaviour of a presentational object is defined through states and transitions between states. A state is specified by a name, entry and exit actions, internal transitions, and/or substates. A transition has an action associated to it, i.e. it is triggered by an event. In the case of user interfaces most of the events are generated by the user, such as mouse focus, mouse clicks, or keyboard inputs. Complex behaviours can be modelled easily in UML with substates. Two different types of substates are possible: sequential and concurrent [Rumb99, Harel87].

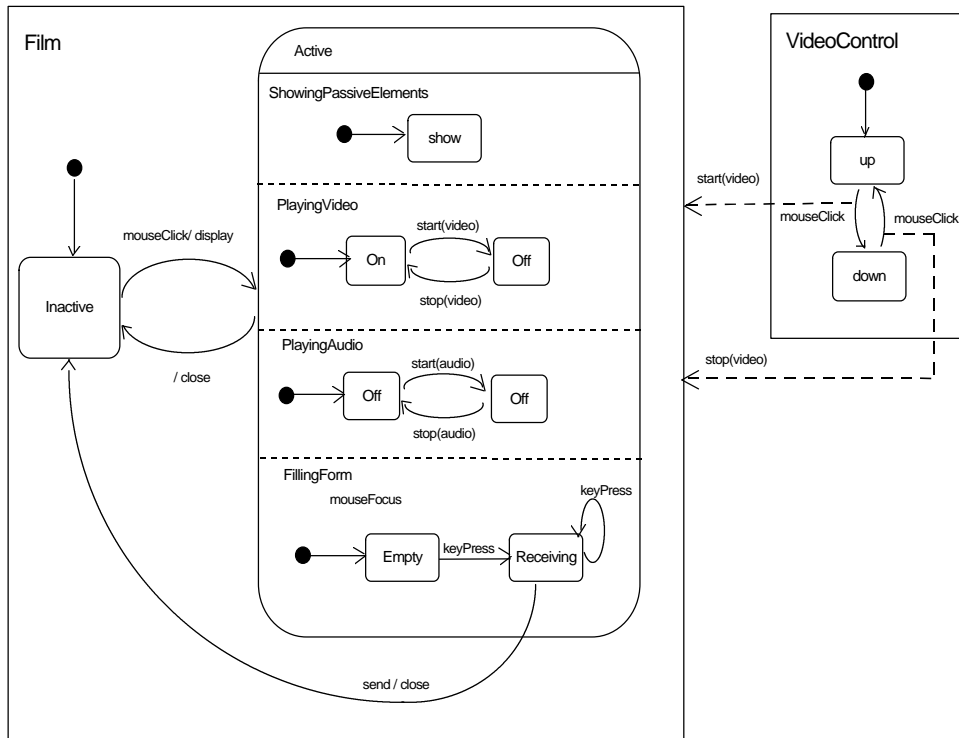


Figure 13: UML Statechart for the Presentational Objects Film and Video Control

In our example we used concurrent substates to denote that the Active state of a Film presentational object has the following substates: showingPassiveElements, playingVideo, playingAudio and fillingForm. The object is in any sequential state from each one of the concurrent substates. The first of these substates has only the state show. It can be excluded from the state diagram as it will permanently remain in this state. We do not include in Fig. 13 the state diagram for the object AudioControl as it is similar to VideoControl state diagram.

The design of these UML state diagrams is expensive and usually so many details are not necessary for user interface objects with well known behaviour. Therefore, we use them only for complex composite user interface objects.

7 Conclusions and future work

Designing hypermedia systems requires thoughtful planning. It is an iterative design process, thus going back to previous steps, correcting and refining the model. But, hypermedia applications that are designed systematically before implementation require less cycles of improvement. The sequence of the steps can be interchanged. As part of the design it might be appropriate to develop a prototype, but we think, that the implementation of the prototype will not replace neither the conceptual nor the navigational and presentational modeling.

In this paper we presented a UML-based methodology for the design of hypermedia systems. It is a model based approach, which modeling techniques are UML diagrams and which graphical representation uses only UML notation. As far as we know this is the only hypermedia design method that is full UML compliant in every step.

Limiting to the notation proposed by the UML instead of introducing a new notation has the enormous advantage of being a well-known standard and to be supported by many case tools. UML has been extended to model the navigation and the presentation according to the UML

extensions mechanisms. This mechanism is based on the definition of stereotypes and the use of OCL. In this paper, design of hypermedia applications has been explained in detail. Appropriate techniques that support the activities of each step have been presented as well as a detailed example.

We hope that the UML extension here described will convince developers of hypermedia applications to take full advantage of the benefits of a designed-based development instead of implementation ad hoc. Implementation of model-based hypermedia applications will introduce flexibility in hypermedia applications, reduce the "lost in hyperspace" problem of classic hypermedia systems and allow for an easy maintenance.

Our future work will concentrate its attention on refining the techniques and notations here presented. Specially hypermedia applications that support more functionality, such as database transactions will be object of study and will be modelled using the proposed technique and notation. The design phase here described is part of a methodology covering the whole life cycle of hypermedia systems. The description of the requirements capture, analysis, implementation, maintenance and quality control is on work.

References

- [BaBieIsa96] V. Balasubramanian, M. Bieber and T. Isakowitz. *Systematic Hypermedia Design*. CRIS Working Papers series. Stern School of Business, NYU, 1996.
- [Booch99] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide*. Addison Wesley, 1999.
- [Boyle97] T. Boyle. *Design for Multimedia Learning*. Prentice Hall, 1997.
- [CaBra97] L. Calvi, and P. Bra, *Improving the Usability of Hypertext Courseware through Adaptive Linking*. FLEXIBLE Hypertext Workshop, 1997.
- [CaCoLu93] L. Carneiro, D. Cowan and C. Lucena. *Introducing ADVcharts: a Graphical Specification of Abstract Data Views*. In Proceedings of CASCON'93, 1993.
- [Cona98] J. Conallen. <http://www.conallen.com/ModelingWebApplications.html>
- [DeTroLeu97] O. De Troyer and C. Leune. *WSDM: a User-Centered Design Method for Web Sites*. Proceedings of the 7th International World Wide Web Conference, 1997.
- [GaMaPa96] F. Garzotto, L. Mainetti and P. Paolini. *Navigation in Hypermedia Applications: Modeling and Semantics*. Journal of Organizational Computing and Electronic Commerce, Vol. 6, No. 3, 1996.
- [Jacobs92] I. Jacobson. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press, 1992.
- [Harel87] D. Harel. *Statecharts: a Visual Formalism for Complex Systems*. Science of Computer Programming, 8(3), 1987.
- [IsStBa95] T. Isakowitz, E. Stohr and P. Balasubramanian. *A Methodology for the Design of Structured Hypermedia Applications*. Communications of the ACM, 38(8),34-44, 1995.
- [Koch98] N. Koch. *Towards a Methodology for Adaptive Hypermedia Systems*, Proceedings of the 6th Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen,

ABIS-98, U. Timm and M. Rössel (EDs.), 1998.

- [KocTur97] N. Koch and A. Turk. *Towards a Methodical Development of Electronic catalogues*, International Journal of Electronic Markets, University of St. Gallen, Vol. 7(3), 28-31, 1997.
- [Lange96] D. Lange. *An Object-Oriented Design Approach for Developing Hypermedia Information Systems*. Journal of Organizational Computing and Electronic Commerce, Vol. 6(3), 269-293, 1996.
- [LeLeYo98] H. Lee, C. Lee and C. Yoo. *A Scenario-Based Object-Oriented Methodology for Developing Hypermedia Information Systems*. 31st Annual International Conference on Systems Sciences, IEEE Computing, 1998.
- [MaCe98] L. Mandel and M. V. Cengarle. *On the Expressive Power of OCL*,. Ludwig-Maximilians-University München, Technical Report 9806, 1998.
- [Rossi96] G. Rossi. Thesis, 1996. (in Portuguese)
- [Rumb99] G. Booch, J. Rumbaugh and I. Jacobson. *Unified Modeling Language Reference Manual*. Addison Wesley, 1999.
- [ScRo98] D. Schwabe and G. Rossi. *Developing Hypermedia Applications using OOHDM*, Workshop on Hypermedia Development Process, Methods and Models, Hypertext '98, 1998.
- [ScRoBa96] D. Schwabe, G. Rossi and S. Barbosa. *Systematic Hypermedia Design with OOHDM*. Proceedings of the ACM International Conference on Hypertext (Hypertext '96), 1996.
- [UML97] UML Document Set, Version 1.1 September 1997, Rational.
<http://www.rational.com/uml/references>
- [WaKl99] J. Warmer and A. Kleppe. *The Object Constraint Language*. Addison Wesley, 1999.
- [Wilkin95] N. Wilkinson. *Using CRC Cards: An Informal Approach to Object-Oriented Development*. Prentice Hall, 1995