Systematic Hypermedia Application Design with OOHDM

Daniel Schwabe, Gustavo Rossi *, Simone D.J. Barbosa

Departamento de Informática
Pontifícia Universidade Católica
R. Marquês de São Vicente, 225
Rio de Janeiro, RJ 22453-900, Brazil
e-mail: [schwabe,rossi,sim] @ inf.puc-rio.br

(*) also LIFIA, F. Cs. Exactas-UNLP, and CONICET, Argentina

Abstract

In this paper we analyze the process of hypermedia applications design and implementation, focusing in particular on two critical aspects of these applications: the navigational and interface structure.

We discuss the way in which we build the navigation and abstract interface models using the Object-Oriented Hypermedia Design Method (OOHDM); we show which concerns must be taken into account for each task by giving examples from a real project we are developing, the Portinari Project. We show which implementation concerns must be considered when defining interface behavior, discussing both a Toolbook and a HTML implementation of the example application.

Keywords: Hypermedia Design, Methodology, Modeling, Object Orientation, Navigation, Interfaces

1. Introduction

In the past three years there has been growing interest in hypermedia design approaches [Izakowitz 95, Garzotto 93, Lange 94]. There are many different problems the hypermedia designer has to deal with, since the combination of navigation through an intricate information space with the unstructured and dynamic nature of multimedia data poses new and complex problems that must be solved in a systematic and modular way (see for example [Hardman93]). Each design activity in a design methodology should address different concerns at the proper stage and at the proper level of abstraction [Nanard 95], and design decisions should be recorded and traced backward and forward in the development process.

In this paper we argue that using the Object-Oriented Hypermedia Design Methodology (OOHDM) [Schwabe95a, Schwabe95b, Rossi95], we can solve these problems while maintaining the "exploratory" nature of the hypermedia paradigm. The structure of this paper is as follows: we first present an example of an application that has been developed using OOHDM, and implemented in both HTML and Toolbook. We then present the OOHDM design process, stressing which design concerns are taken into account in each activity and which modeling and abstraction mechanisms are used during each step in the development cycle, using the example for illustrating the various concepts. Next, we make a few remarks about the main points of OOHDM. Finally we compare our method with others in the hypermedia field and indicate some further work in hypermedia design.

2. An Example Application

Since space limitation prevents us from going into detail of all the formalisms, we will make a more "informal" presentation of the main concepts of OOHDM. To help that, we present a brief example taken from the HTML implementation (see <http://www.lids.puc-rio.br/~pp>) of a hypermedia presentation of the material collected by the Portinari Project [Lanzelotte 93]. This collection includes all the works produced by one of the foremost Brazilian painters, Candido Portinari, as well as documents (books, photos, videos, letters, newspaper and magazine articles, recorded interviews, etc...) documenting his life and of his contemporaries, which include many of the most important intellectuals, artists and political figures of his generation. The application presents the material of the Portinari Project to the general public, for access in a kiosk or through the Word Wide Web.

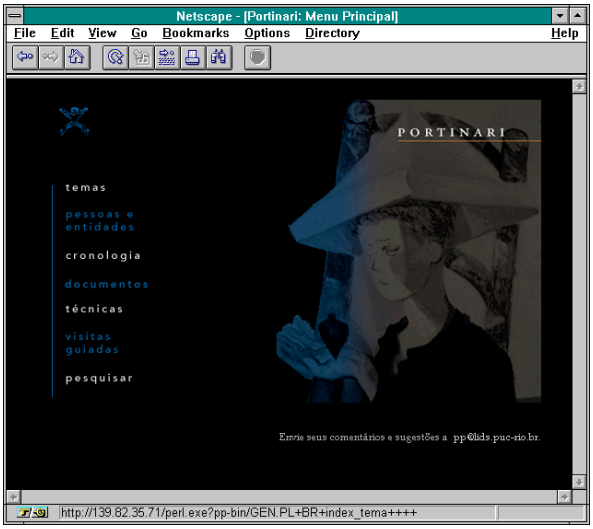
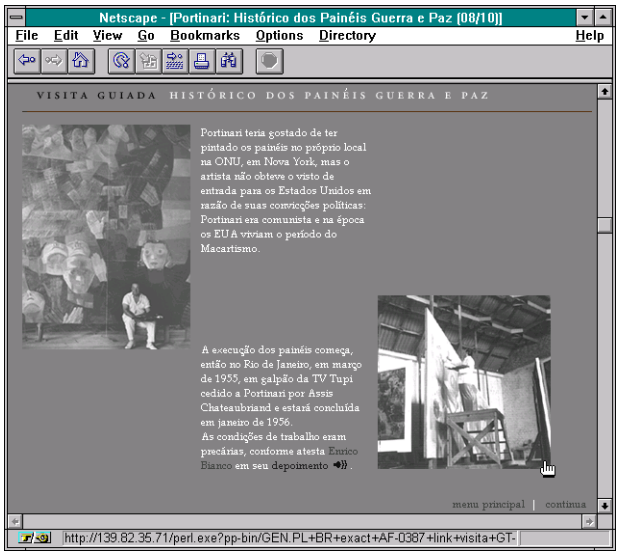


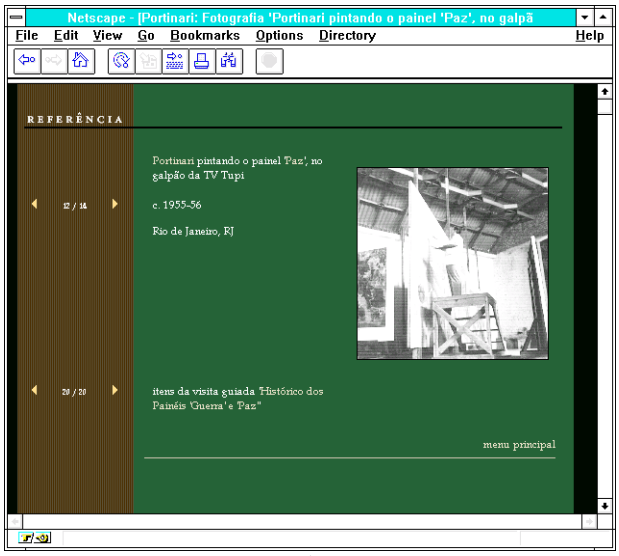
Figure 1. The Main Menu in the Portinari Project Application (WWW version)

Figure 1 shows the main menu, where the user has a choice of viewing a hierarchical theme index (clicking on "tema"); an alphabetical index of persons and entities (clicking on "pessoas e entidades"); a timeline (clicking on "cronologia"); a hierarchical subject index (clicking on "documentos); a hierarchical index of techniques (clicking on "técnicas"); a guided tour (clicking on "visita guiada"); and a search function (clicking on "pesquisar").

Let us suppose the user has chosen to follow a guided tour, which in this example tells the story, as a linear sequence of screens, of how the panels "War" and "Peace" were commissioned, painted and installed at the UN headquarters in NY. Figure 2a shows one screen of the guided tour.



(a)



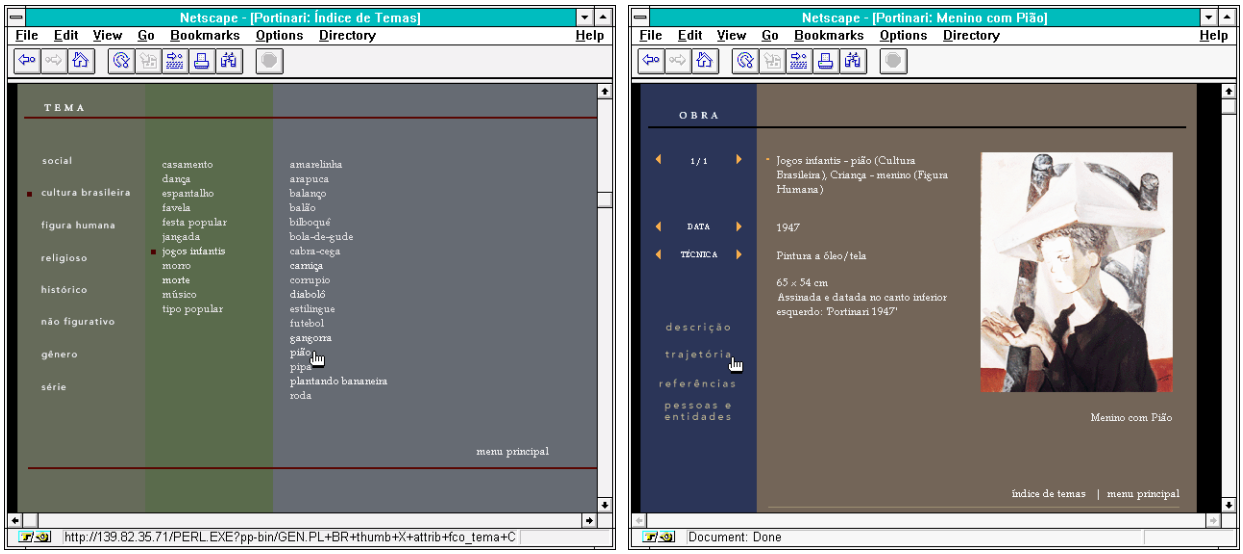
(b)

Figure 2.
(a) A screen of the guided tour about the "War" and "Peace" panels at the UN headquarters
(b) A screen showing the information about a photograph in the collection

Each screen may contain references to documents or artworks that are part of the collection, such as an interview (which may be heard by clicking on the speaker icon). For instance, clicking on the picture, it is possible to find out more information about this document (a photograph), as shown in figure 2b. In this screen, the arrows next to the date allow the user to navigate through other available photographs in chronological order (and it is indicated that this is the 12th out of 14 photographs available). Since the reader arrived at this photograph via the guided tour, the bottom left row indicates that it is possible to navigate directly to other items cited in the guided tour (by clicking on the arrows), or to navigate back to the guided tour (by clicking on the guided tour name to the right of the arrows). This information would not be present if the reader had arrived through other paths.

Changing subject, let us assume the reader has chosen the "tema" (theme) button in the main menu. He will be presented with a series of hierarchically nested indexes of subjects, which may end in a screen such as shown in figure

3a. The bullets next to each category marks the choice at each successive level. Assuming the reader has chosen the theme “cultura brasileira/jogos infantis/pião“ (Brazilian culture/children’s games/top), the reader will see the screen in figure 3b.



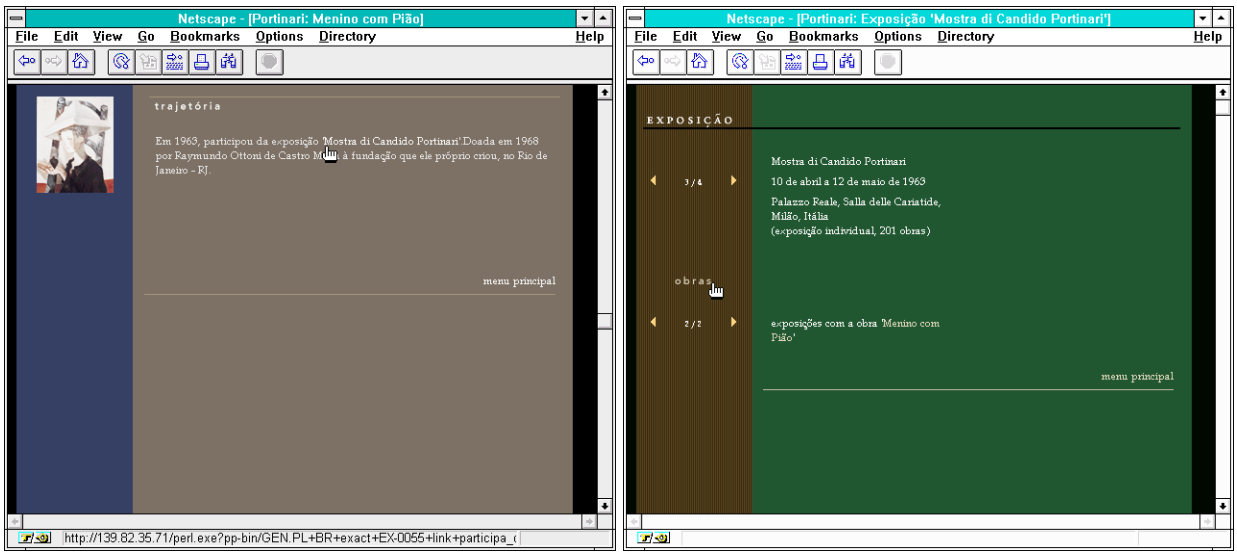
(a) (b)

Figure 3.
(a) A hierarchical theme index
(b) A screen showing information about an artwork documented by the Portinari Project

This screen shows information about an artwork (a painting in this case). On the top left, the first row corresponds to the “theme” navigation context, allowing the user to navigate through artworks that belong to this particular theme – in this case, the theme is “Jogos Infantis - pião”, and it is the only artwork with this theme in this implementation. Notice also that this artwork is also classified under another theme, “Criança/menino (Figura Humana)” (Child/boy (Human Figure)). The second row corresponds to the “date” navigation context, allowing the user to browse through the instances of class “artwork” in chronological order. The third row corresponds to the “technique” navigation context. For all three contexts, clicking on any of the arrows follows a “previous” or “next” link in the corresponding context.

The descriptive text below these lines shows general comments about the artwork, including references to other artworks when, for example, the current artwork is a study for another one. Clicking on the image of the artwork will show a large scale, full screen version of the image.

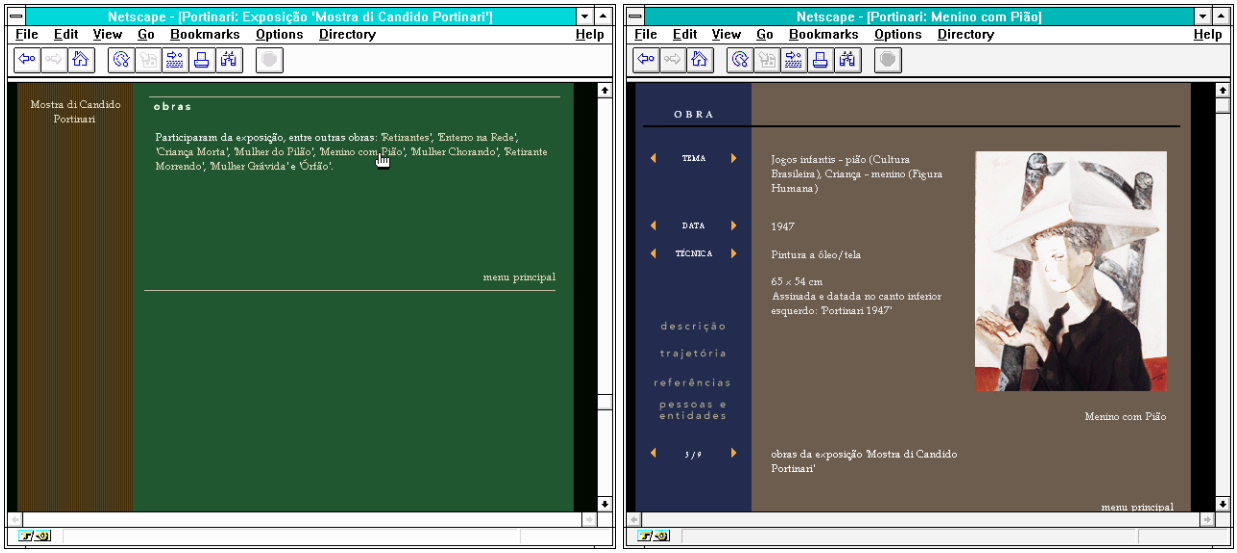
The anchor “descrição” (description) presents a textual description of the artwork; the anchor “trajetória” (trajectory) presents a textual description of its history (who bought it, if and where it was shown as part of an exhibit or auction, etc...); the anchor “referências” (references) presents a list of documents in the collection that make reference to this artwork; and anchor “pessoas e entidades” (persons and entities) shows a list of persons or entities related to this artwork. Let us assume the reader has clicked on “trajetória”, and sees the information shown in picture 4a., where it is mentioned that this artwork was shown in the “Mostra di Candido Portinari” exhibit. Clicking on the exhibit’s name, the user will see a screen with information about it, shown in figure 4b.



(a) (b)

Figure 4
(a) Additional information about an artwork, including exhibit in which it appeared
(b) A screen showing information about an exhibit documented by the Portinari Project

This screen follows the same conventions as the previous ones. The first row at the top left allows the user to navigate, by clicking on the arrows, to other exhibits in chronological order, and it is indicated that this is the third out of four exhibits available. The bottom row, on the left, shows that the reader has arrived at this exhibit navigating from an artwork, which has also been shown in other exhibits. Clicking on the button "obras" (artworks), the user can see the other artworks that participated in this exhibit, as shown in figure 5a.



(a) (b)

Figure 5
(a) The list of artworks in an exhibit
(b) Information about an artwork, including contextual information about exhibits in which it appeared

If the user now chooses the same artwork, he will see the screen in figure 5b (compare with figure 3b). The bottom row now shows information about the artworks that participated in the event "Mostra di Candido Portinari", which is an exhibit. There is a context made up of all the artworks that appeared in that exhibit (this is the fifth of nine), and the user may navigate forward or backward in this context by clicking on the arrows. If the user clicks on the name of the context ("obras da exposição Mostra di Candido Portinari"), an index (in this case, a screen with thumbnail images) of all the artworks in that context is shown, and the user may select one to continue navigation. It should be noted that this "selective appearance" of contextual information was a design decision for this application.

2. OOHDM's Design Process

The Object-Oriented Hypermedia Design Method is a model-based approach for building hypermedia applications. It comprises four different activities namely conceptual design, navigational design, abstract interface design and implementation. They are performed in a mix of incremental, iterative and prototyped-based development style, as discussed in [Nanard 95]. During each activity, except for the last one (implementation), a set of object-oriented models describing particular design concerns are built or enriched from previous iterations. In Figure 6, taken from [Schwabe 95b], we summarize OOHDM activities, and modeling approaches.

Steps	Products	Mechanisms	Design Concerns
Conceptual Design	Classes, sub-systems, relationships, attribute perspectives	Classification, composition, generalization and specialization	Modeling the semantics of the application domain
Navigational Design	Nodes, links, access structures, navigational contexts, navigational transformations	Mapping between conceptual and navigation objects	Takes into account user profile and task. Emphasis on cognitive aspects.
Abstract Interface Design	Abstract interface objects, responses to external events, interface transformations	Mapping between navigation and perceptible objects	Modeling perceptible objects, implementing chosen metaphors. Describe interface for navigational objects
Implementation	Running application	Those provided by the target environment	Performance, completeness

Figure 6. Summary of the OOHDM Method (taken from [Schwabe 95b])

2.1 Conceptual Design

During Conceptual Design a model of the application domain is built using well known object-oriented modeling principles (OMT, [Rumbaugh 91]), augmented with some primitives such as attribute perspectives and sub-systems. Conceptual classes may be built using aggregation and generalization/specialization hierarchies. The main concern during this step is to capture the domain semantics as "neutrally" as possible, with very little concern for the types of users and tasks. The product of this step is a class and instance schema built out of Sub-Systems, Classes and Relationships. The instance schema describes exceptional objects and it is intended to avoid class explosion when possible.

We have chosen an object-oriented modeling approach because the semantic of aggregation and generalization/specialization is well defined in object-oriented data models [Banerjee87], there are powerful approaches for describing object views, which we use for defining the navigational model, and finally because object-orientation has arguably become the standard choice for building complex interactive applications [Jacobson 92]. Section 3.2 presents other reasons based on the behavioral semantics of objects.

Figure 7 shows (a simplified version of) the conceptual schema for the example application, where, for the sake of conciseness, we have not shown all attributes of all classes. A brief inspection of this schema shows that it is quite general, and has information that can be used by many different types of users, for many different purposes.

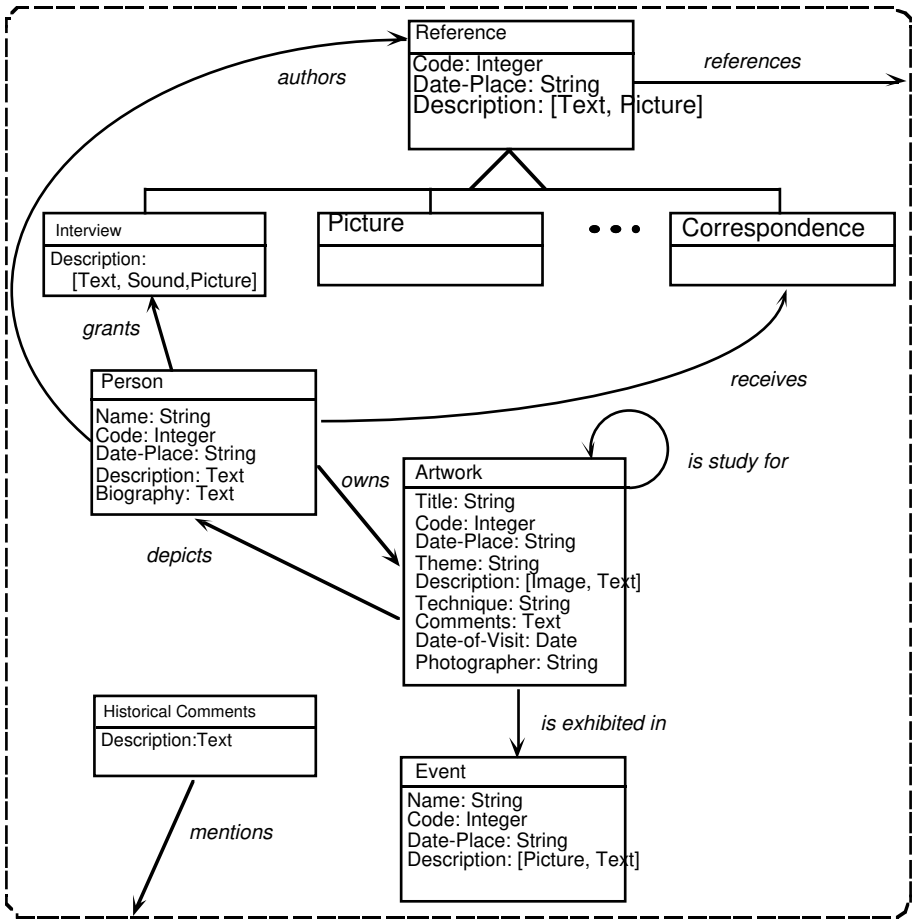


Figure 7. Then conceptual schema for the "Portinari Project"

In this example, class "Artwork" has attribute "Description" which has two perspectives, "Text" and "Picture", corresponding to textual (which may be used for full text search) and pictorial representations of the artwork, exemplifying an extension OOHDM makes to OMT. The arrow between "Reference" and the dashed box is a shorthand to indicate that "Reference" is related to all other classes within the dashed box via the "references" relation; the same holds for "Historical Comments".

2.2 Navigational Design

In order to have an application that can be used by a set of intended users, trying to accomplish a certain set of tasks, it is necessary, in general, to reorganize the information represented in the conceptual model. In OOHDM, this is achieved by defining a navigational model that is a view of the conceptual model. This reflects the point of view that one of the key distinguishing features of hypermedia applications is the notion of navigation, which must be designed taking into account the types of intended users, and the set of tasks they are to perform using the application. Different navigational models may be built for the same conceptual schema, entailing possibly several applications, each catering to different set of users and tasks.

Navigation design is expressed in two schemas, the navigational class schema, and the navigation context schema. The navigable objects of a hypermedia application are defined by a navigational class schema, whose classes reflect the chosen view over the application domain. In OOHDM, similarly to HDM [Garzotto 93] and RMD [Isakowitz 95], there is a set of pre-defined types of navigational classes: nodes, links, and access structures. The semantics of nodes and links are the usual in hypermedia applications, and access structures such as indexes and guided tours represent possible ways of accessing nodes.

Nodes are defined as object-oriented views of conceptual classes defined during conceptual design, using a query language similar to the one in [Kim94], allowing a node to be defined by combining attributes of different related classes in the conceptual schema. They contain single typed attributes and link anchors, and may be atomic or composites as in [Gronbaek94]. In the same way, links reflect relationships intended to be explored by the final user and are also defined as views on relationships in the conceptual schema.

Figure 8 shows three navigational class definitions for the example. Although there is some similarity with the conceptual classes, several differences are worth noting. For example, node class "Artwork" does not contain several attributes of the "Artwork" conceptual class, as they represent information deemed inappropriate or of no interest to

the intended audience – e.g., photographer's name, date of visit (by photographer), name of researcher, etc.... Note also that attributes with multiple perspectives become different attributes of the node class, for example "Description"/"Text" becomes "Description" and "Description"/"Image" becomes "Picture".

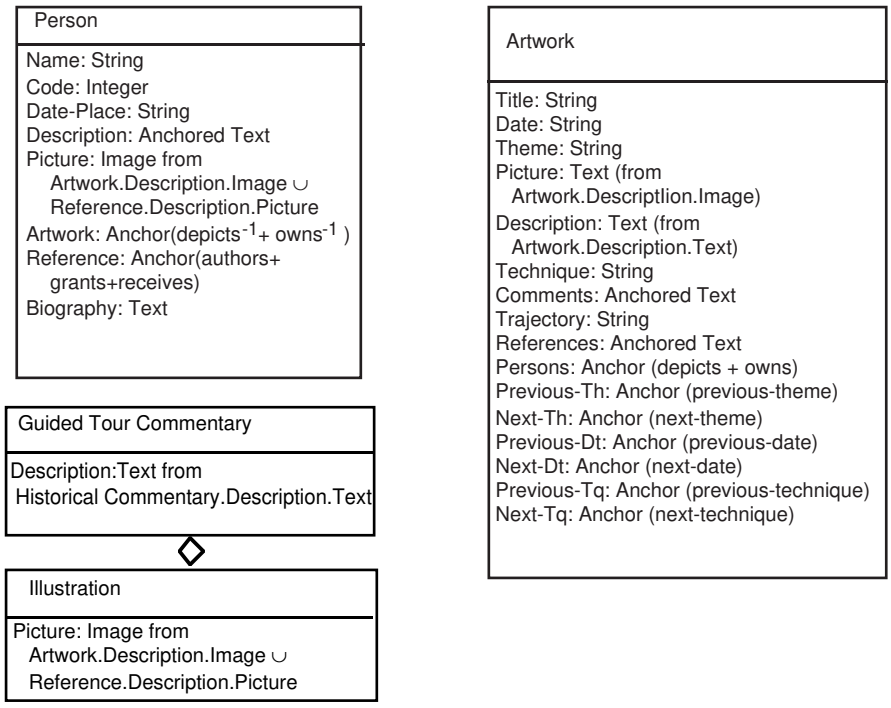


Figure 8. Navigational Class Schema for the Portinari Project application

Navigational class "Person" also shows an example of "cutting and pasting" of attributes, as it essentially enriches conceptual class "Person" with the "Description"/"Image" perspective taken from the "Description"/"Image" attribute of a "Photograph" or "Artwork" which that object ("Person") is related to, via, respectively, the "is referenced" or "depicted in" relations. Similarly, navigational class "Guided Tour Commentary" is built by combining the "Description"/"Text" attribute of conceptual class "Historical Commentary" with an aggregation of Illustration, whose "Picture" attribute is derived from the "Description"/"Image" attribute of "Photograph" or "Artwork". It should also be noted that node classes include "Anchor" attributes for all the outgoing links, which in turn are views of relations in the conceptual schema. The anchors corresponding to "previous" and "next" in some of the examples shown in section one will be further explained after we discuss navigational contexts.

In well-designed hypermedia applications the author must take into account the way in which the user explores the hypermedia in order to avoid redundant information and to prevent the user from getting "lost in hyperspace". For example, one may access artworks in chronological order. When examining one artwork, the user may decide that he is interested in looking at other artworks with a similar theme. At that point, it should be made clear that the user now has the options of continuing to examine the artworks by theme or in chronological order, and the user should have the means to indicate his choice of navigation. A different situation arises when presenting the same material in different contexts, as exemplified in figures 3b and 5b.

In OOHDM the main structuring primitive of a navigational schema is the notion of navigational context (see [Schwabe 95b, Barbosa 95] for a more extensive discussion). A navigational context is a set of nodes, links, context classes and other (nested) navigational contexts. They are induced (in different ways depending on the type of class) from navigation classes – nodes, links, indexes, and guided tours. They may be defined intentionally or extensionally, by either defining a property that all nodes and links in the context hold or by enumerating its members. For example, a 1-to-n link induces a navigational context that allows traversing sequentially all the link targets (e.g., all references to an artwork). In the same way an index (e.g., all paintings about childhood) or a Guided Tour (some selected paintings) define a navigational context.

Context classes complement the definition of a navigational class, (a node), indicating which information is shown and which anchors are available when accessing the object in a particular context. Navigational contexts play a similar role as collections [Garzotto94] and have been inspired in the concept of nested contexts [Casanova93].

Figure 9 shows the navigation context schema for the Portinari Project application, and figure 10 shows the definition of one of the contexts that appear in the navigation schema.

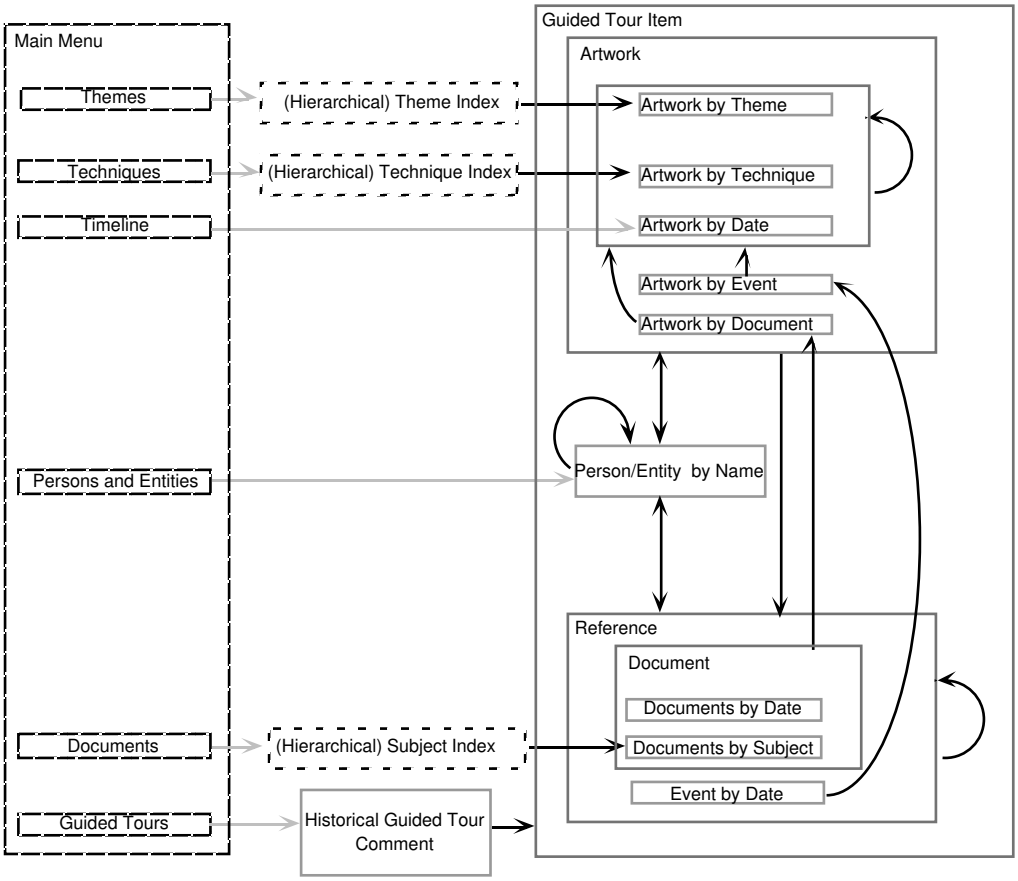


Figure 9 - Navigational Schema for the Portinari Project application

In the diagram in figure 9, the actual navigation contexts are boxes with light gray border (e.g., "Artwork by Theme"); boxes with darker gray borders are used to group contexts for referencing purposes, where arrows pointing to solid boxes indicate links to any enclosed item; the hierarchical indexes drawn with light dashed lines were not drawn in detail, as they would be very complex (consider the nesting required to represent the index in figure 3.a). The boxes on the left ("Main Menu" items) are just place-holders to indicate the nesting of the elements pointed by the gray arrows.

NC Artwork by <event>
includes: $\forall$ a $\in$ Artwork, e $\in$ Event | (e,a) $\in$ exhibited $\wedge$ e.Title= <event>
entry points: e1 [first node]
path: circular, ordered ascending on a.date, a $\in$ Artwork
behavior: step by step

Figure 10. Definition o f Navigational context "Artwork by Event"

The navigational context defined in figure 10 specifies the artworks that were shown in a given exhibit. Its elements are instances of navigational class "Artwork", extended by the context class "Artwork in Event", whose definition is shown in figure 11 below. Notice that this context class essentially adds the context navigation anchors that allow the user to browse through the elements of this context (see figure 5b, bottom row on the left).

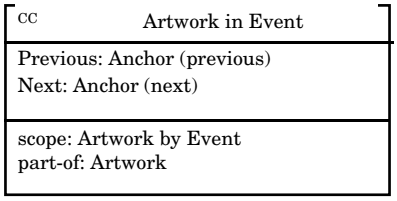


Figure 11. Context class “Artwork in Event” extends class “Artwork” within the “Artwork by Event” context

The dynamic navigational structure is completely specified by defining the navigational transformations that occur while traversing links. In some cases, for instance, we may want specify that the source node remains active, and the target node becomes active as well; or that all destination nodes of a one-to-many link become active simultaneously. In OOHDM we use an object-oriented state-transition model derived from Statecharts [Harel87], that we call Navigation Charts, supporting structural and behavioral nesting and allowing expressing dynamic navigational behavior. The Navigation Chart definition is complemented with a set of predicates expressed in Temporal Logic that allow querying the specification against desired properties such as safety, livenesss, etc.

2.3 Abstract Interface Design

Once the navigational structure has been defined, it must be made perceptible to the user through the application interface, which is done in this step by defining an abstract interface model. This means defining which interface objects the user will perceive, and in particular the way in which different navigational objects will look like, which interface objects will activate navigation, the way in which multimedia interface objects will be synchronized and which interface transformations will take place.

A clean separation between both concerns, navigational and abstract interface design, allows building different interfaces for the same navigational model, leading to a higher degree of independence from user-interface technology. In addition, this separation allows a better understanding of the overall application structure by clearly indicating which transformations in the interface are also navigational transformations and which are simply local interface transformations that do not affect the state of the navigation.

In spite of the previously mentioned interest in hypermedia design methods, and even though interface design is a critical aspect of the creation of large hypermedia applications [Schuler94], human-computer interfaces are usually described using implementation and environment dependent tools, and design decisions at the interface level are seldom documented.

In OOHDM we use the Abstract Data View design approach for describing the user interface of a hypermedia application [Cowan95]. Abstract Data Views are formal, object-oriented models of interface objects and they are specified by showing:

- The way in which they are structured using aggregation and generalization/specialization as abstraction mechanisms. ADV's express the static lay-out structure that implements the interface metaphor [Hannemann93, Schuler94]. Perception properties are also specified as attributes or parts of an ADV. ADVs allow defining the interface appearance of navigational objects and other useful interface objects (such as menu bars, buttons and menus).
- The way in which they are statically related with navigation objects. We use Configuration Diagrams [Coleman92] as a diagrammatic tool for expressing these relationships.
- How they behave when reacting to external events; in particular how they trigger navigation and which interface transformations occur when the user interacts with the application. We use ADV-Charts [Carneiro94], another derivative of Statecharts, that adds both structural and behavioral nesting and a Petri-Net like notation for expressing synchronization issues typical of multimedia data.

The modeling constructs we use during navigational and abstract interface design are very similar – in fact we use classes and objects with a formal connection model – and thus we obtain a seamless transition between both activities, allowing incremental construction of the navigational and abstract interface models. At the same time, outstanding design decisions are recorded using a notation that is powerful and concise. Navigation- and ADV-charts may be easily related and combining information in interface and navigation classes results in a strong traceability model. The reader may find a complete description of our approach in [Rossi95a]

We give brief example using the navigation class “Artwork” (from the schema in figure 8) and its associated interface objects, by presenting the behavioral specification of the corresponding Abstract Data View in just enough detail to show that we can build a formal but nevertheless simple and readable specification.

Figure 12 shows a simplified Configuration Diagram specifying some abstract interactions among interface and navigational objects (called “owners” of the interface object). Dashed lines are required services while full ones

are provided services. Attributes “theme”, “date” and “technique” act as static interface objects, i.e. they do not react to external events. Nested Abstract Data Views “Picture”, “Description”, “Context Interface”, “Show Description” and “Show References” exhibit a user-perceivable behavior. For example when the user clicks on “Show Description” the ADV “Description” is displayed. “Context Interface” is in turn composed of nested ADV implementing anchors for context navigation as discussed before.

This dynamic description is specified using another related formalism, ADV-Charts that extend StateCharts to user interface specification (see Rossi95b). We can specify the behavior of each nested ADV using a different statechart or use a concise specification as shown in figure 13.

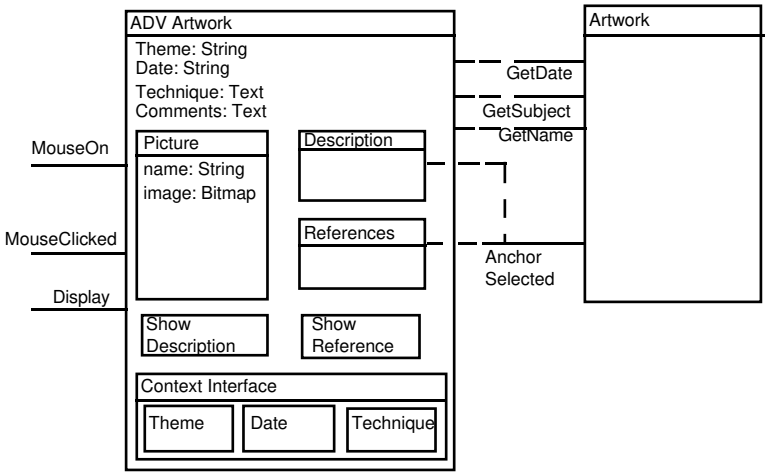


Figure 12. Configuration Diagram for ADV “artwork”

ADV Painting can react to external events MouseOn, MouseClicked and Display, while node “Artwork” provides “GetDate”, “Subject” and “Name” and “AnchorSelected”. When both ADV and node attributes have the same name and type we can omit specifying services like “GetDate” because it can be unambiguously understood by implementors. Note that “Show Description” and “Show References” do not trigger navigation, they just implement local interface behavior.

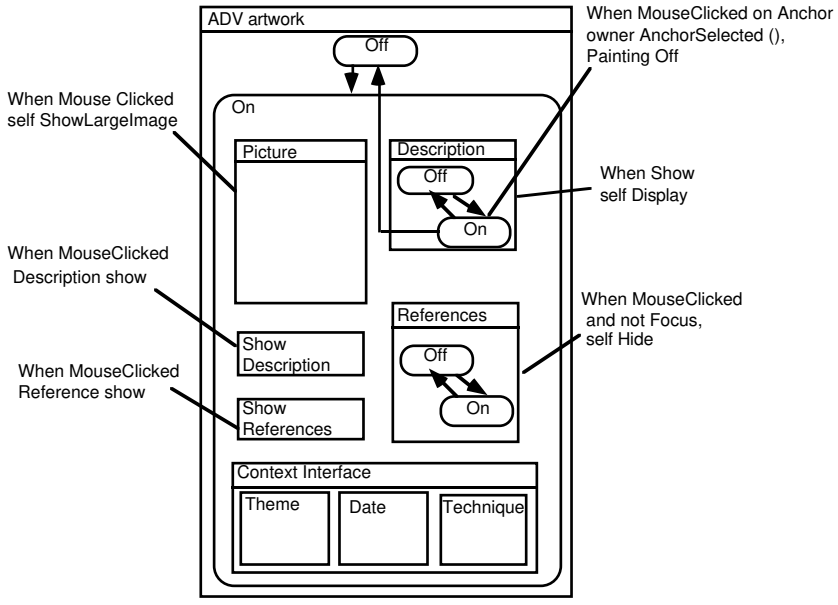


Figure 13. ADV Chart for ADV “Artwork”

In the figure above some interface objects (such “References”) may have embedded anchors triggering navigation; in this case the owner (node “artwork”) is sent the message “anchorSelected” with corresponding anchor as argument and the interface objects are removed from the perception context (“artwork” in state “Off”).

2.4. Implementation

To obtain a running implementation, the designer has to map the navigational and abstract interface models into concrete objects available in the chosen implementation environment. The model generated after performing previously defined activities can be implemented in a straightforward way using many of the currently available hypermedia platforms such as Hypercard, Toolbook, Director, HTML, etc.

The object-oriented style of the abstract interface specification allows simplifying the interface in implementations in different platforms. For example, figure 14 shows the Toolbook version of the same node shown in figure 5b. In this implementation, the anchors "descrição", "trajetória", "referências" and "pessoas e entidades", when activated, produce different behavior than the HTML version. In the Toolbook-based one the expression: self Display is implemented by a scrollable "pop-up" field which shows the corresponding information ("trajetória" in the figure), while in the HTML implementation this effect is achieved by a local-scroll effect using a local anchor (see figure 4a). Similarly the expression "self ShowLargeImage" (in "Picture") is implemented in different ways in both implementations. The condition "not Focus" in "Description", easily implemented in Toolbook, is changed in the HTML implementation by providing a button for scrolling back, although it could also be implemented using the "back" button of the browser.

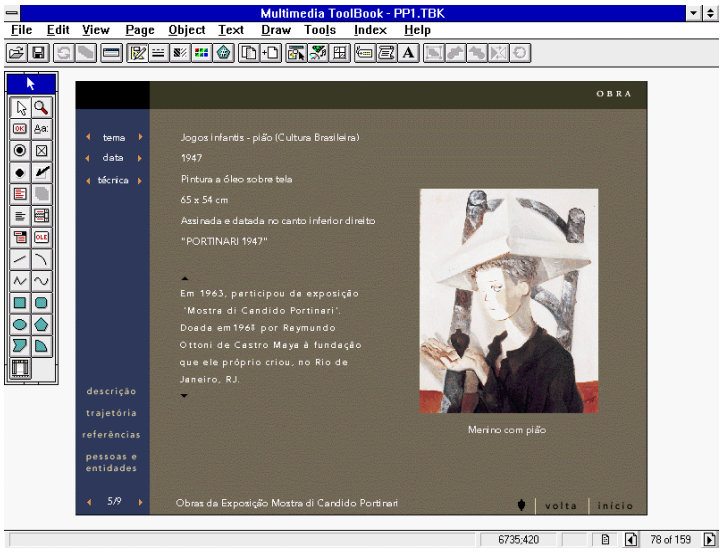


Figure 15. Toolbook version of the screen for "Artwork"

There are other differences in each implementation of the example. For instance, the guided tour in the Toolbook version has additional transition effects, as well as soundtrack. The observing reader will have noticed that the Toolbook version has a "volta" (back) button in the bottom right set of controls, which is not present in the HTML version. The reason is that the semantics of the "back" function in the web browser is exactly the same as the "previous" link in navigation contexts, whereas the same function in Toolbook has a different semantic, and therefore it has to be programmed explicitly.

Another interesting kind of implementation for OOHDM-based projects is the one provided by object-oriented frameworks [Johnson 88]. We have designed an implemented a framework that allows adding hypermedia functionality to object-oriented applications. In this case the conceptual model is finally implemented using an object-oriented language and navigational classes are instantiated from the framework (see [Rossi95b, Carvalho95]).

3. Discussion

3.1 Relations between the models

As the reader may have noticed there are some relationships among the conceptual and navigational models; in fact in other hypermedia design methods (like HDM or RDM) both activities employ the same kind of modeling primitives. We find conceptual modeling a different activity when either we are planning to build more than one application in the same domain (for different users profiles or tasks) or when we want to stress the difference among conceptual classes and attributes and more opportunistic and navigation oriented objects. In simpler applications, however, the conceptual and navigational models will be very similar. Derived entities in HDM play the same role, as they may represent combinations among conceptual objects (entities in HDM) similar to our navigational classes.

Another interesting issue arises when comparing the navigational and interface models. It is clear that each navigational transformation (e.g. leaving a node and reaching another) causes an interface transformation (e.g. a window is closed and another is opened). However, not every interface transformation is caused by a navigational operation – for example a "local" scroll effect showing more information of a node is not caused by a link traversal,

even if a new window (or interface element) appears. Understanding this difference helps in obtaining better design models that can be more easily implemented in multiple platforms. By having the two models, OOHDM facilitates this separation of concerns, helping the designer focus on the important issues at each step of the design process.

Finally, although in theory one could completely separate the abstract interface specification from the implementation, in practice we have observed that many times some features of the implementation environment, if already chosen, may condition "low-level" aspects of the interface, as for instance the fact that it is not possible to have "pop-up" fields (or more generally, layering) in HTML browsers.

3.2. Using Object Orientation

The advantages of using structural object-oriented modeling constructs in hypermedia have been discussed elsewhere [Nanard 91]. These kind of advanced data modeling approaches provide high level abstraction and composition mechanisms (e.g. classification, aggregation and inheritance hierarchies) with well-defined semantics. As object-oriented methods are now mature we can take profit from both the process models [McGregor94] and the heuristics defined in those methods [Odell94]. We also benefit from the work on views in object oriented databases [Kim 94], which allows us to define navigational models as views of conceptual models, with well defined semantics.

Our view, however, is broader than those previously mentioned approaches. First we want to specify systems in which we find dynamic multimedia behavior; besides we want our method to be useful to design a broad spectrum of applications that incorporate navigation, such as CASE environments, decision support systems, etc; and lastly, we want to use a uniform formalism during the whole life-cycle. Structural object-oriented approaches are not enough for coping with these requirements, so we consider all constructs in our modeling approach (i.e. conceptual classes, nodes, links, interface objects, etc.) to be full-fledged objects that can react to external messages by triggering their own behavior. Considering objects not only as structured, encapsulated records but also as dynamic entities allows us to:

- extend the conceptual and navigational models with classes defining other design artifacts (like editors and browsers) with well-defined communication patterns with hypermedia components. This is important in applications combining hypermedia with other kind of computations [Bieber95] or allowing the user not only to navigate across the hyperbase but also to edit and modify it [Rossi95b]. When building "hybrid" applications, i.e., those adding hypermedia functionality to non-hypermedia applications such as Software Engineering Environments [Bigelow88] or Geographic Information Systems [Scholl92], the OO approach allows smooth integration, as the author may design other kinds of classes (not necessarily nodes, links or access structures) that communicate with navigational ones with the usual semantics of message passing in object-oriented design.
- conform with existing models and tools for dealing with the human-interface of multimedia applications, defining the interface model as a reactive model that interacts with the navigational model to achieve the desired navigational and interface transformations [Rossi95a].
- reason about common navigation and interaction design patterns [Alexander77, Gamma94] in order to build new hypermedia applications by reusing design ideas and to have a common vocabulary among designers [Rossi95b].

4. Related Work

OOHDM is a direct descendant of HDM. It differs from HDM first in its object-oriented nature, and in that it includes special purpose modeling primitives for both navigational and interface design. In RMM [Izakowitz95] the authors also emphasizes the need for an iterative and incremental development life cycle; they also consider navigational and interface design to be important activities. OOHDM differs from RMM in that it includes the concept of navigational contexts and in that it explicitly models the user-interface interaction and the effect of each user-generated event both in the interface and in the navigational state of the application.

Navigational contexts are similar to "collections" as reported in [Garzotto94]. Although not cast in an object-oriented framework, collections play a similar role as navigational contexts. There is no close equivalent to navigational objects as mappings of conceptual objects and collections do not allow object extension as is possible with context objects. The possible navigation semantics for collections are a subset of the navigational semantics allowed by contexts.

Other authors have proposed the use of formal models for specifying hypertext semantics. In [Zheng92] for example, StateCharts are used for modeling different browsing semantics in hypertext. Our work is similar in that it uses ADVcharts (a derivative of StateCharts) for expressing the dynamics of the application. However it is different because it clearly separates navigational behavior (a concern of navigational design) from interface specification; moreover as ADVs may be composed to form higher level ADVs, nesting in ADVcharts not only indicate composition by behavior as in StateCharts, but also structural composition.

5. Conclusions

We have presented the Object Oriented Hypermedia Design Method, discussing each of its four basic steps. A full discussion of all models used would each take a full paper by itself, so have tried to illustrate the main points by using an example taken from an actual implementation, running both on the WWW and in Toolbook.

We are now pursuing investigation in several topics, among which are

- ways to express design patterns at all levels;
- tools to at least semi-automate the translation of OOHDM models into runtime environments such as the WWW, Toolbook and Director;
- graphical "wysiwyg" tools to simplify abstract interface specification;
- incorporating intentionally specified links;
- ways to specify navigation contexts at high levels of abstraction, using, for example, notions from linguistics such as Rhetorical Structure Theory.

6. References

[Alexander77] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King and S. Angel, "A Pattern Language". Oxford University Press, New York 1977.

[Banerjee87b] J. Banerjee et al, "Data model issues for object oriented applications", ACM TOIS 5, 1987.

[Barbosa 95] S. D. J. Barbosa,. "Modeling and Specification of Navigation in Hypermedia Applications", MSc Thesis, Dept. of Informatics, PUC-Rio, Feb. 1995 (in Portuguese).

[Bieber95] M. Bieber; C. Kacmar, "Designing Hypertext Support for Computational Applications", Comm ACM, August 1995, pp 99-107.

[Bigelow88] J. Bigelow, "CASE and Hypertext", IEEE Software, March 1988.

[Carneiro94] L.M.F. Carneiro; M.H. Coffin; D. D. Cowan; C.J.P. Lucena, "ADVcharts, a Visual Formalism for Highly Interactive Systems". In M.D. Harrison and C. Johnson editors, "Software Engineering in Human-Computer Interaction". Cambridge University Press, 1994.

[Carvalho95] S. Carvalho, G. Rossi; A. Garrido, "Design Patterns in an Object-Oriented Framework for Hypermedia", Proceedings of the Conference of the Chilean Computer Science Society, forthcoming.

[Casanova93] Casanova, M.A.; Tucherman, L.; Lima, M.J.; Netto, J. L. R; Rodriguez, N.; Soares, L.F.G., "The Nested Context Model for Hyperdocuments", Proceedings of Hypertext'91, San Antonio, 1993.

[Coleman92] D. Coleman; F. Hayes; S. Bear, "Introducing Objectcharts or How to use Statecharts in Object-Oriented Design", IEEE Transactions on Software Engineering, 18(1), 9-18, January 1992.

[Cowan93] D. D. Cowan; R. Ierusalimschy; C.J.P. Lucena; T.M. Stepien, "Abstract Data Views", Structured Programming, 14(1):1-13, January 1993.

[Cowan95] D. D. Cowan; C. J. P.Lucena, "Abstract Data Views, An Interface Specification Concept to Enhance Design for Reuse", IEEE Transactions on Software Engineering, Vol.21, No.3, March 1995.

[Gamma94] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns, Elements of Reusable Object-Oriented Software", Addison Wesley, 1994.

[Garzotto93] F. Garzotto, D. Schwabe, P. Paolini, "HDM- A Model Based Approach to Hypermedia Application Design", ACM Transaction on Information Systems, Vol. 11, #1, Jan. 1993, pp. 1-26.

[Garzotto94] F. Garzotto, L. Mainetti and P. Paolini, "Adding Multimedia Collections do the Dexter Model", Proceedings ECHT'94, Edinburgh, Sept. 1994.

[Gronbaek94] K. Gronbaek, "Composites in a Dexter-Based Hypermedia Framework", Proceedings of the ACM European Conference on Hypermedia Technology, Edinburgh, 1994.

[Hannemann93] J. Hannemann, M. Thuring, "What matters in developing interfaces for hyperdocument presentation?", Workshop in Methodological Issues on the Design of Hypertext-based User Interfaces, Darmstadt, Germany, July 1993.

[Hardman 93] L. Hardman; D. Bulterman; G. Van Rossum, "Links in Hypermedia, the Requirements for Context", Proceedings Hypertext'93, ACM, pp 183-191.

[Harel87] D. Harel; A. Pnueli; J.P. Schmidt; R. Sherman, “On the formal semantics of statecharts”. Proc. 2nd. IEEE Symposium on Logic in Computer Science, Ithaca, N.Y., June 1987.

[Isakowitz 95] T. Isakowitz; E. Stohr; P. Balasubramaniam, “RMM, A methodology for structured hypermedia design", Comm ACM, August 1995, pp 34-48.

[Jacobson 92] I. Jacobson; M. Christerson; P. Jonsson; G. Overgaard, “Object-Oriented Software Engineering- A Use Case Driven Approach”, Addison Wesley, 1992.

[Johnson 88] R. E. Johnson; B. Foote. “Designing Reusable Classes”, Journal of Object-Oriented Programming, June/July 1988, Volume 1, Number 2, pages 22-35.

[Kim 94] W. Kim, “Advanced Database systems", ACM Press, 1994.

[Lange 94] D. Lange, “An Object-Oriented design method for hypermedia information systems", Proceedings of the 27th. Annual Hawaii International Conference on System Science, January 1994.

[Lanzelotte 93] Lanzelotte, R.S.G.; Marques, M.P.; Penna, M.C.S.G.; Portinari, J.C.; Ruiz, F.D.; Schwabe, D., “The Portinari Project, Science and Art team up together to help cultural projects”, Proc. of the Second International Conference on Hypermedia and Interactivity in Museums (ICHIM’93), Cambridge, UK, Sep. 1993.

[McGregor 94] J. Mc. Gregor; T. Korson, "Integrated Object-Oriented Testing and Development Process”, Comm ACM, September 1994.

[Nanard91] J. Nanard; M. Nanard. “Using Structured Types to Incorporate Knowledge in Hypertext”, Third ACM Conferences on Hypertext Proceedings, Hypertext'91 ed. ACM Press. pp.. 329.

[Nanard 95] J. Nanard; M. Nanard, “Hypertext Design Environments and the Hypertext Design Process”, Comm. of the ACM, Vol. 38, #8, Aug. 1995.

[Odell 94] J. Odell, “Six different kinds of compositions”, Journal of Object Oriented Programming, Vol. 5 #8, 1994.

[Rossi95a] G. Rossi; D. Schwabe; C.J.P. de Lucena; D.D. Cowan, “An Object-Oriented Model for Designing the Human-Computer Interface of Hypermedia Applications”, Proceedings of the International Workshop on Hypermedia Design (IWHD'95), Springer Verlag Workshops in Computing Series, forthcoming. (available at <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/ 95_07_rossi.ps.gz>).

[Rossi95b] G.Rossi; A. Garrido; S. Carvalho, “Object-Oriented Patterns for Hypermedia Applications”, Proceedings of Patterns Languages of Programs (PLOP'95), forthcoming.

[Rumbaugh91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W.Lorensen: “Object Oriented Modeling and Design”, Prentice Hall Inc. 1991.

[Scholl92] M. Scholl; A. Voisard, “Geographic Applications: An experience with O2” in F. Bancilhon (ed), Building an Object-Oriented Database System, Morgan Kaufman, 1992.

[Schuler94] W. Schuler, “A Design Space for Hypermedia Interface”, Workshop on Methodologies for Designing and developing Hypermedia Applications , Edinburgh, September 1994.

[Schwabe94a] D. Schwabe; G. Rossi, “From Domain Models to Hypermedia Applications. An Object-Oriented Approach”, International Workshop on Methodologies for Designing and developing Hypermedia Applications , Edinburgh, September 1994.

[Schwabe94b] D. Schwabe; S. D. J. Barbosa, “Navigation Modeling of Hypermedia Applications”, Technical Report MCC 42/94, Departamento de Informática, PUC-Rio, 1994 (available at <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/ 94_42_barbosa.ps.gz>)

[Schwabe95a] D. Schwabe and G. Rossi, “Building Hypermedia Applications as Navigational Views of Information Models”, Proceedings of the Hawaii International Conference on System Sciences, Hawaii, Jan. 1995. (available at <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/ 94_41_schwabe.ps.gz>)

[Schwabe 95b] D. Schwabe and G. Rossi:, “The Object Oriented Hypermedia Design Model”, Comm. of the ACM, Vol. 38, #8, Aug. 1995. (available at <http://irss.njit.edu:5080/cgi-bin/bin/option.csh?sidebars/schwabe.html>).

[Sowmya94] A. Sowmya and S. Ramesh: “Extending Statecharts with Temporal Logic”. SCS&E Report 9401, School of Computer Science and Engineering. The University of New South Wales, 1994.

[Zheng92] Y. Zheng, M-C. Pong: “Using Statecharts to Model Hypertext”, Proceedings of the ACM European Conference on Hypertext, Milano, December 1992.