Navigation Modelling in Hypermedia Applications

Daniel Schwabe and 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,sim]@inf.puc-rio.br

ABSTRACT

A distinguishing feature of hypermedia applications is the possibility of navigational access by the user. Although this type of access can be an advantage, there many problems associated with it, for example user disorientation ("lost in hyperspace").

This paper presents the concept of navigational context, which is used to design the navigation structure of a hypermedia application. Navigational contexts are defined in the framework of the Object Oriented Hypermedia Design Methodology, which strives to systematize the process of hypermedia application design, from the conceptual modeling phase all the way to implementation.

Throughout the paper, examples of the use of navigational contexts are shown using Microsoft's Art Gallery CDROM application.

1. INTRODUCTION

In [Garzotto 93], it was stated that "the degree of success in a hypertext application is directly related to the author's ability to capture and organize the structure of a complex subject matter in such a way as to render it clear and accessible to a wide audience". It presented HDM (Hypertext Design Model), which supports an approach based on an authoring-in-the-large model for specifying hypermedia applications. In [Schwabe94, Schwabe95] an object-oriented approach to the construction of large hypermedia applications - OOHDM, Object-Oriented Hypermedia Design Method is outlined; this work describes the specification of the navigational design of a hyperdocument in OOHDM by means of navigational contexts.

A navigational context is defined as a set of nodes, links and contextual information. It provides a consistent way to specify the navigational aspects of the application, the next step after specifying the information structure using OOHDM.

The concept also explores the idea that, when presented in context, the pieces of information in a hyperdocument acquire a clearer, more precise meaning, thus making it easier for the user to understand their intended meaning. Navigational contexts may also minimize the problem of user disorientation and may reduce the cognitive load imposed by navigating through complex hyperdocuments.

2. OVERVIEW OF OOHDM

The Object-Oriented Hypermedia Design Method is a four step process, in which a model is built or enriched at each step, based on the previous one, and the last step is the actual application implementation. The style of development is meant to be a mix of iterative, incremental and prototype-based.

2.1 Step 1: Conceptual Design

In this step a model of the application domain is built using well known object-oriented modeling principles [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.

Figure 1 shows an example of a Conceptual Schema for Microsoft's Art Gallery CDROM, showing the conceptual classes, their attributes and relationships. Each class is represented by a "Class Card", drawn as a box, with the attributes shown inside. We will use this application throughout this paper, to illustrate the various concepts being defined.

A composite class is indicated by the ◊ symbol (for example, "Painting"), with cardinality restrictions indicated next to the incoming lines of sub-classes (for example, "n" repetitions of class "Comment".

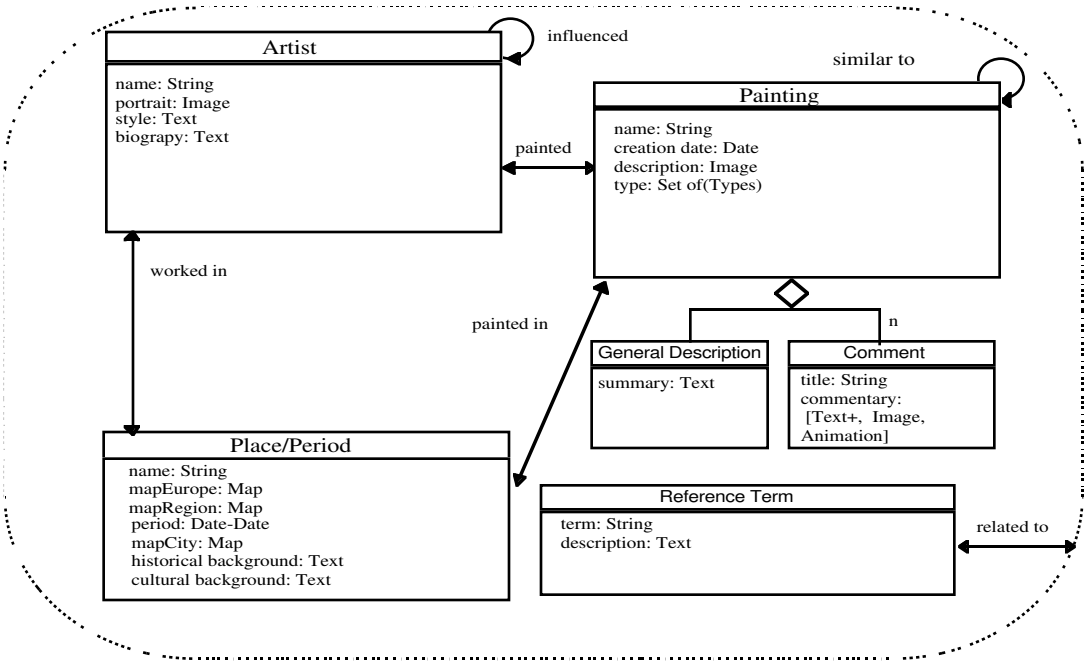


Figure 1. The conceptual schema of Microsoft's Art Gallery

The class "Artist" consists of personal data, information about the painter's style, general characteristics of his works, etc. The class "Painting" is made of general information, notes about its composition and details, and so on. The class "Place/Period" consists of the geographical information about a city or region in a certain period of time. The class "Reference Term" represents the glossary terms of the application.

Another extension to the classical object model is the use of perspectives for attribute values, as exemplified in figure 1 by the attribute "commentary" of sub-class "Comment" of class "Painting". This attribute has possible values types Text, Image and Animation, with the "+" next to Text indicating that it is the default type, i.e., all instances have at least a value of this type.

Finally, to simplify the diagram, we have indicated that "Reference Term" may be related (via "is related to") to all other classes by linking it to a dashed box, with the semantics that the reference applies to all classes enclosed in the box.

2.2 Step 2: Navigational Design

In OOHDM, an application is seen as a navigational view over the conceptual domain. This reflects the point of view that one of the key distinguishing features of hypermedia applications is the notion of navigation. It is in this step that the designer takes 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 thus expressing different views (applications) on the same domain. The navigational structure of a hypermedia application is defined by schema specifying navigational classes that reflect the chosen view over the application domain. The main structuring primitive of a navigational schema is the notion of navigational contexts, which are induced (in different ways depending on the type of class) from navigation classes such as Nodes, Links, Indices, and Guided Tours.

This step will be described in more detail in the remainder of the paper.

2.3 Step 3: Abstract Interface Design

Once the navigational structure has been defined, it must be made perceptible to the user through the application's 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, and also allowing conformance with varying user needs or preferences. This step is discussed in detail in [Rossi 95].

2.4. Step 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 steps 1-3 can be implemented in a straightforward way using many of the currently available hypermedia platforms such as Hypercard, Toolbook, MacWeb, KMS, Guide, Microcosm, etc... The use of a uniform set of modeling constructs (objects and classes), our approach allows a smooth transition from domain modeling to navigational and interface design.

The implementation step does not require that an object-oriented environment be used, although it might make the job easier. In the event that the runtime environment is not object-oriented, many techniques can be used to map an object-oriented specification into a non object-oriented runtime environment.

3. NAVIGATIONAL CONTEXTS

3.1 Informal Description

The navigational structure of an application is defined in terms of navigational contexts. A navigational context is defined as a data structure made out of nodes and links, where the nodes contain domain information (from the conceptual model) and the links represent relationships (also from the conceptual model). To define a navigational context the designer tries to identify "knowledge chunks" in the domain that "make sense" together, and therefore should guarantee the coherence of the presented material. Navigational contexts should provide a reference context, making the user comfortable enough to explore the material without feeling disoriented or lost in the complex

network of nodes and links. Subsequent sections will formalize these concepts, and show more precisely how navigational contexts are formed.

The concept of navigational contexts was inspired in the Nested Context Model (NCM) [Casanova93]. The NCM employs nested contexts to allow "the implementation of abstractions of aggregation, hierarchy and view". It is also related to the notion of "collections" as described in [Garzotto 94].

3.2 Navigational Classes and Objects

The basic building block to construct navigational contexts is the notion of Node. A Node is a navigational object that is meant to collect information (from the conceptual schema) that is deemed to be relevant to a particular class of users seeking to perform a particular set of tasks.

To draw an analogy, consider the way merchandise is organized in a supermarket – typically, it is divided by product category, laid out in different sections. In spite of this, oftentimes one finds, for example, sugar and coffee machine paper filters on the shelves next to the coffee, which violates the basic product category layout. The designers of the supermarket are well aware that the user is not very interested in "product categories" per se, but rather wants to purchase the ingredients for making coffee as expeditely as possible – and this layout makes it much easier for the user! The analogy here is that a conceptual model (the product category layout) is remapped into a functional view (put all ingredients for making coffee – the user task – close to each other).

A Node is a navigational class whose attributes are mapped from the attributes of conceptual classes according to some user profile and intended tasks. Figure 2 shows an example of a node class, "Painting" in the Art Gallery application, together with one sample instance "The Battle of San Romano". This class is derived from the corresponding class in the conceptual schema shown in figure 1; the basic difference is the addition of the "author" attribute, which is mapped from the conceptual class "Artist".

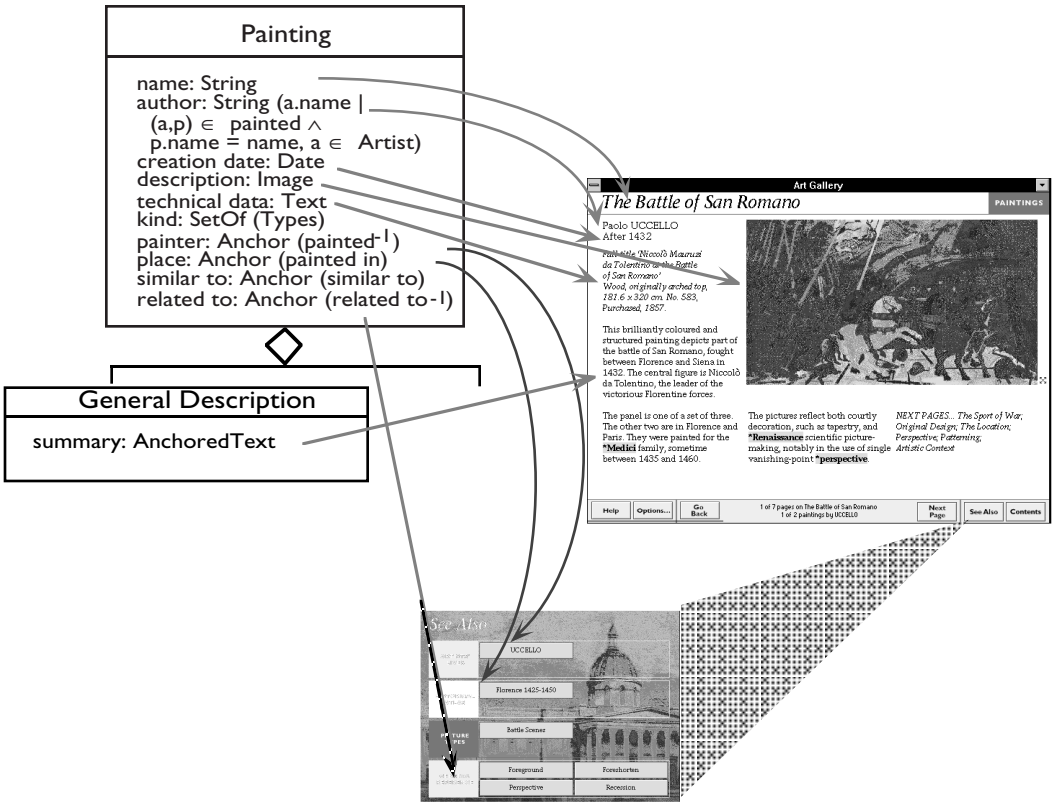


Figure 2. An example of node class definition – Painting.– and a sample instance, "The Battle of San Romano"

The actual node class "Painting" is a composition of two sub-classes, "General Description" and "Comments", much in the same way as in figure 1, but not fully shown here for the sake of simplicity. In particular, note that not all attributes are necessarily visible at the same time – for example, all the anchors shown in the bottom part appear in a separate screen when the user clicks on the "See Also" button. This button is an example of a "local scroll", which is specified in the Abstract Interface Design step. Attribute "kind" illustrates the use of an attribute for computational purposes – it is used to select a given painting when navigating by type of painting; it never appears directly at the interface.

In the general case, a node will have its attributes derived from attributes of several conceptual classes. Nevertheless, it is not infrequent that a conceptual class is mapped (almost) directly into a navigational class, as in the example above. Node attributes cannot have more than one perspective; when mapping conceptual attributes with more than one perspective (see "commentary" in figure 1), several different node attributes – one for each perspective – must be defined.

A distinguishing feature of nodes is the inclusion of attributes with the type "Anchor". This type is considered as a primitive class whose parameter is the link type it is associated to. The semantics of navigation are partially defined when defining the behavior of this class; we will come back to this issue later.

Another navigational class is the link, which is mapped from the relationships in the conceptual model. A link class definition contains its source node class, its destination node class and other attributes, including cardinality. When the relationship that is used to define the link is a "part-of" relationship between members of a composition (or aggregate), it is oftentimes called a structural link, as in the original HDM. When a link is defined between two node classes derived from the same conceptual object, but with different types for some attribute, one has the equivalent of perspective links in HDM.

The set of node and link class definitions is called navigational class schema, which is analogous to the conceptual class schema. Figure 3 shows the navigational class schema for Art Gallery. In this schema, conceptual class "Place/Period" has been mapped into a composite class, because in this application a place/period is always viewed as part of the place, in spite of the conceptual schema, where each place/period is specified independently from each other. For example, if the user is reading about Michelangelo and wishes to see the cities and periods where he worked, he will be presented the options "Florence 1475-1500", "Rest of France 1500-1600", "Rome 1500-1525", and "Rome 1525-1550". When the user chooses to navigate to "Florence 1475-1500" and keeps choosing "Next Page", she should expect to reach the other cities/periods related to Michelangelo. Instead, she will browse through all the periods in which Florence was a city of some relevance. This is an indication that each period of Florence is part of the place "Florence", from the navigational point of view.

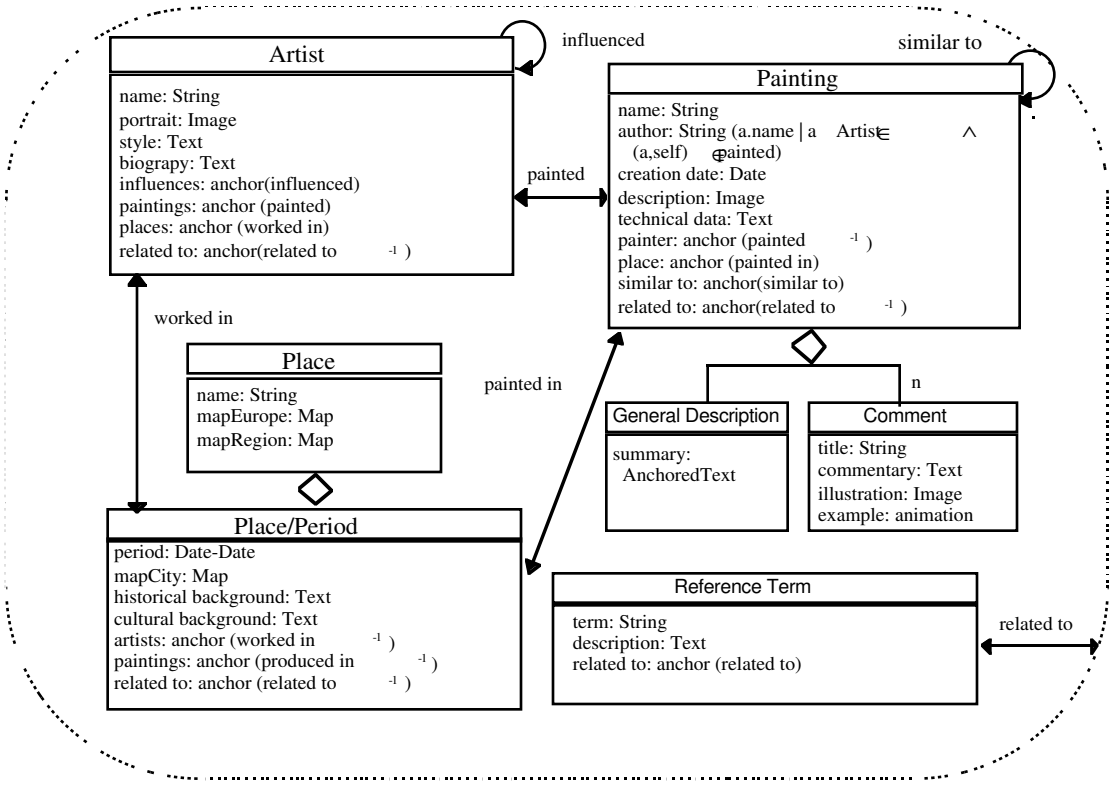


Figure 3. The navigational class schema for Art Gallery.

Note that class "Place" is used solely to factor out common attributes ("name", "mapEurope" and "mapRegion).Note also the several attributes of class "Comment" (part of "Painting") which were generated from the various perspectives in the corresponding conceptual class.

3.3 Navigational Contexts

Navigational contexts are defined as sets of nodes, links, context classes (to be defined momentarily) and other navigational contexts. They may have attributes, used to represent meta-information (i.e., information about the context itself); one or more entry points; a path connecting its elements; a behavior specification; a specification of which elements are visible in nested contexts, and which elements it is making visible in enclosing contexts (if any); and they may be related to other navigational contexts. Figure 4 shows a Navigational Context Card, analogous to a Class Card, used to specify a navigational context. In some cases, we may represent the "related to" field graphically instead of textually. We will explain the meaning of each of these parts of the specification as we show examples in the subsequent subsections.

Name of the Navigational Context	
Attributes (including metainformation)	
Includes (Objects, Classes, NCs)	
Related to	NC/Class/Obj Relationship
Entry points	
Path	
Behavior	
Imports/Exports	
Comments	

Figure 4. A Navigational Context Card

The path specification describes how the elements of the context are connected, when navigating inside the context; the behavior specification describes the control flow during navigation. Both specifications will be discussed in more detail in section 3.7, even if the examples in the following subsections already give an idea of their semantics.

Navigational contexts may be nested, which can be used to model hierarchical access structures that exhibit a "branching" path structure (see [Zellweger 89]. When a context contains another one, its elements are, in principle, accessible to any enclosed context, unless otherwise state in the "imports/exports" part of the navigational context card.

There are several types of navigational contexts that are induced by the various elements in the navigational class schema, which are briefly described next.

3.3 Navigational Contexts

3.3.1 Arbitrary

This is the most general type of context. Its elements are defined in an arbitrary manner by the designer, choosing them according to some criteria. Typical examples of such contexts are the "Guided Tours" found in many applications, where the author examines some of the nodes in the hyperbase one at a time, sequentially. Figure 5 shows an example from Art Gallery, including a screen from the application.

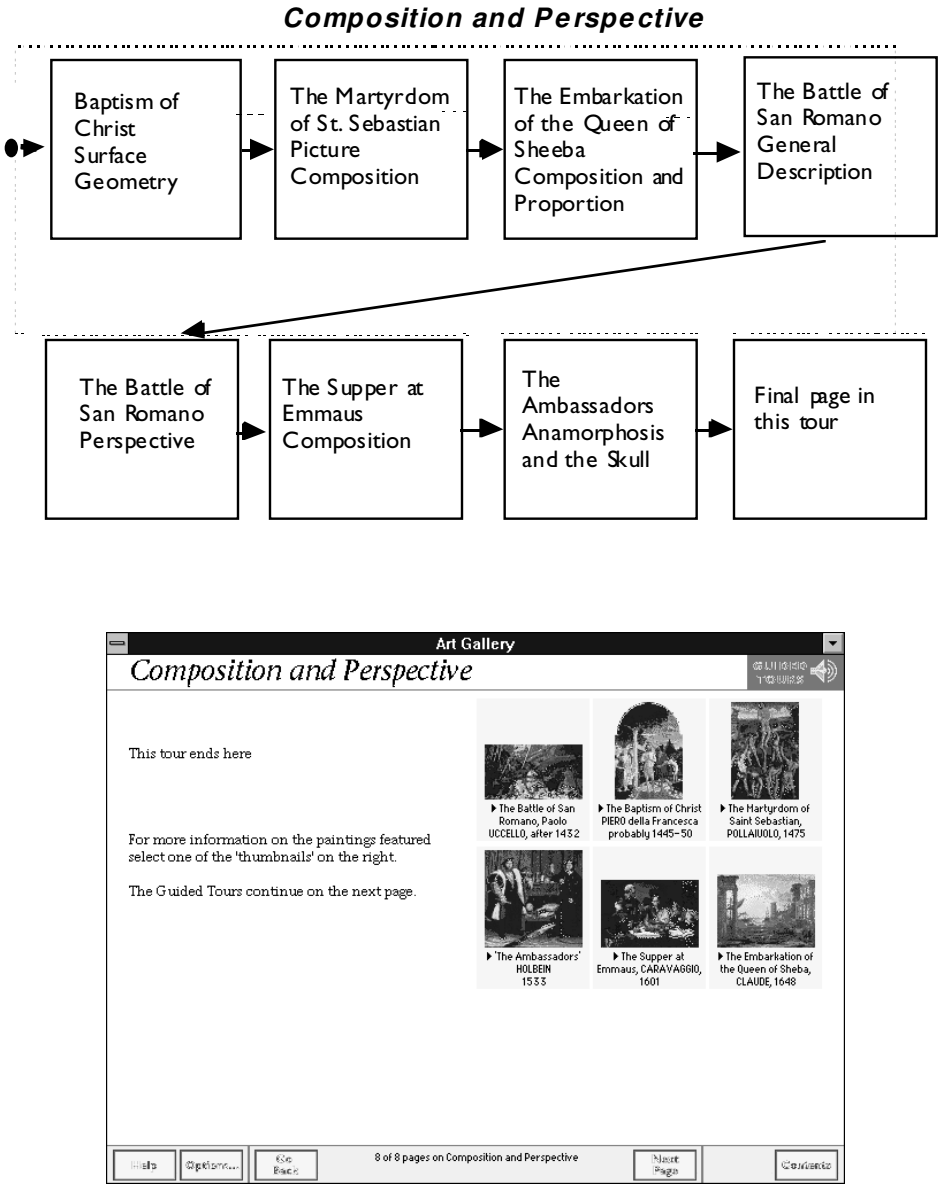


Figure 5. The Composition and Perspective Navigational Context in the Art Gallery application. The screen shown corresponds to the last object in the tour, pointing back to the other elements in the tour.

Figure 6 shows the corresponding navigational context card specifying the navigational context of figure 5. The last element in the tour is an index structure, pointing back to the other elements of the tour; we will elaborate indexes in section 3.3.5, showing how they constitute navigational contexts in themselves.

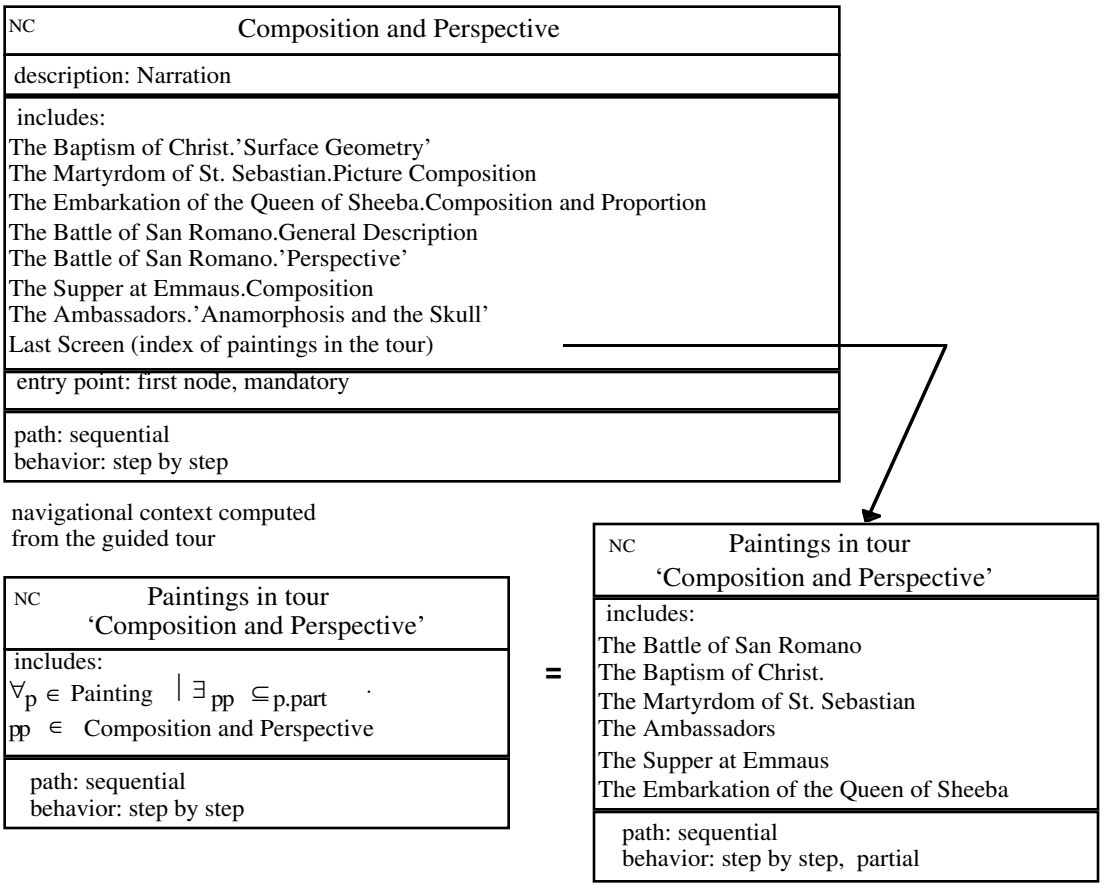


Figure 6. Navigational context cards of the "Composition and Perspective" context (guided tour).

In this example, there is only one entry point; the elements are connected by a sequential path (shown in figure 5), and the user may navigate in it by stepping in order from one element to the next.

Another important point to notice in this example is that the elements of the guided tour are parts of composite object "Painting", and therefore only the links associated with these parts (and not those associated with the whole object – the root of the composite – they are part of) are visible inside the guided tour. This is a consequence of the type of behavior specified "step by step, partial", to be discussed in section 3.7. This is illustrated in Figure 7 for painting "The Baptism of Christ", with the different links available within context "Composition and Perspective" and within context "Paintings by Piero della Francesca".

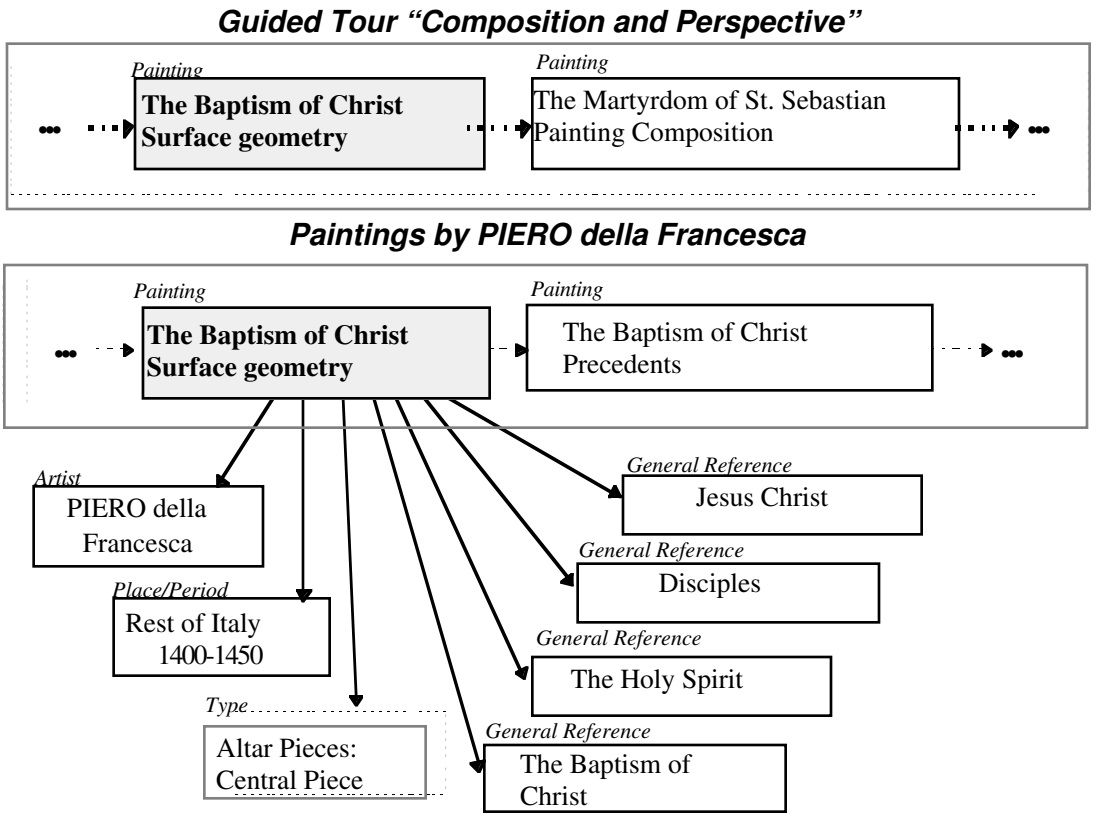


Figure 7 The links available vary according to the context within which an object is being accessed., as exemplified with painting "The Baptism of Christ", viewed in two different contexts.

3.3.2 Class

A second type of navigational context is induced by a class definition. It is formed by collecting all nodes belonging to that class; for example, all instances of class "Artist", as shown in figure 8.

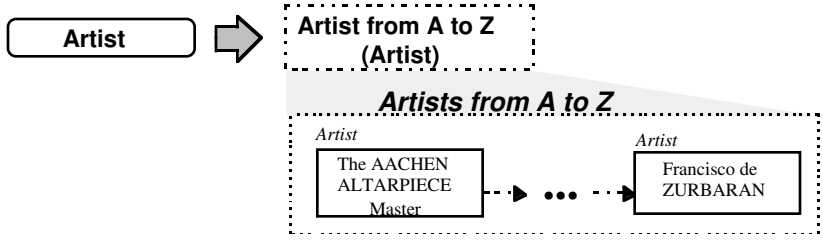


Figure 8 A Class derived navigational context.

Figure 9 contains the corresponding navigational context card.

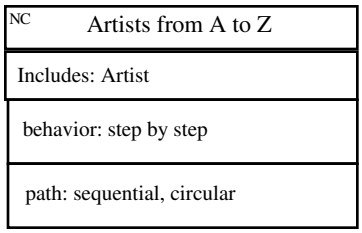


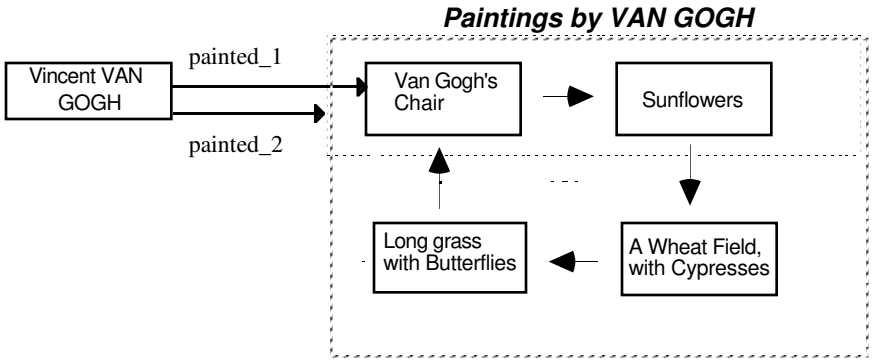
Figure 9 The navigational context card "Artists from A to Z" induced by class "Artist"

In this example, the "Artists from A to Z" navigational context allows the user to navigate sequentially from one painting to the next, going back to the first one after visiting the last one (circular path).

3.3.3 Link derived

This kind of navigational context is induced by the presence of one-to-many links generated by a conceptual link, as exemplified by link class "painted" between "Artist" and "Painting". Navigating through this link may have several meanings – the user may be taken to the first (arbitrary) object in the set of destination objects, or she may be required to explicitly choose among the several alternatives. This situation is depicted in figure 10, taking the paintings painted by Van Gogh as an example.

Link painted_1 takes the user directly to the first node in the context



Link painted_2 requires the user to choose one of the possible destination nodes in the context

Figure 10. An example of a link derived navigational context: All the paintings painted by Van Gogh

Figure 11 shows the navigational context card specifying this link derived context. Notice that the actual definition of the navigational context is the same irrespectively of the chosen navigation semantics; the only information necessary is the identification of the entry point to the context. The two possible navigation semantics are specified in the relationships part of the class card "Artist", where the notation "Paintings by <name>@e1" means context "Painting by <name>" at entry point e1. By default, if no entry point is specified, the user must choose one of the elements in the destination context (as is the case for relationship "painted_1" in the example).

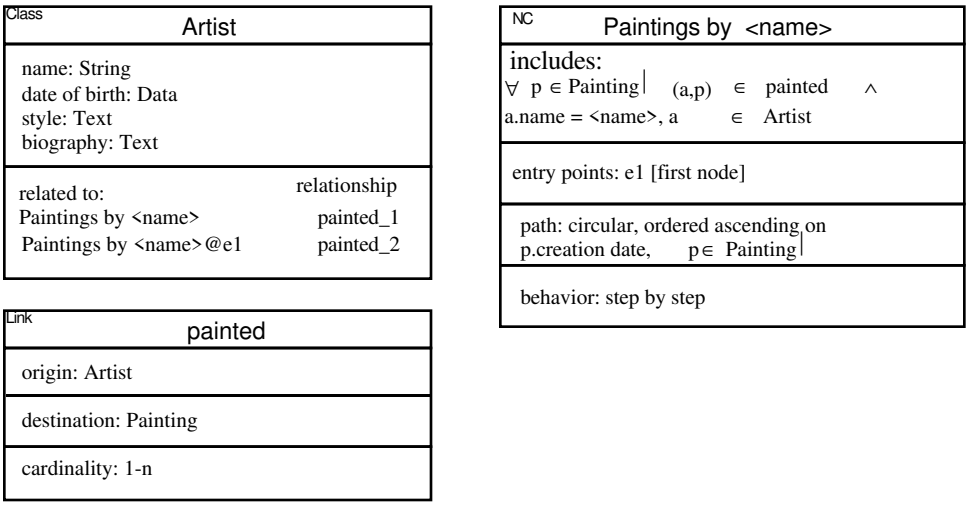


Figure 11. The navigational context card for the context "Painting by <artist>, derived from link "painted" between "Artist" and "Painting".

This example shows also the specification of an ordered path through the elements of the context – in this case, that paintings are ordered by ascending value of creation date.

3.3.4 Structure derived

We have already seen that nodes may be composites. This induces a context that collects all nodes that belong to the same composite object; the links in this context will allow structural navigation inside the composite. For instance, node "Painting" has been defined (see figure 2) as a composition of one "General Description" and n "Comments". An example instance is painting "Sunflowers", which has two "Comment" nodes besides the "General Description" node – "The Yellow House" and "Like Stained Glass"; this is illustrated in figure 12.

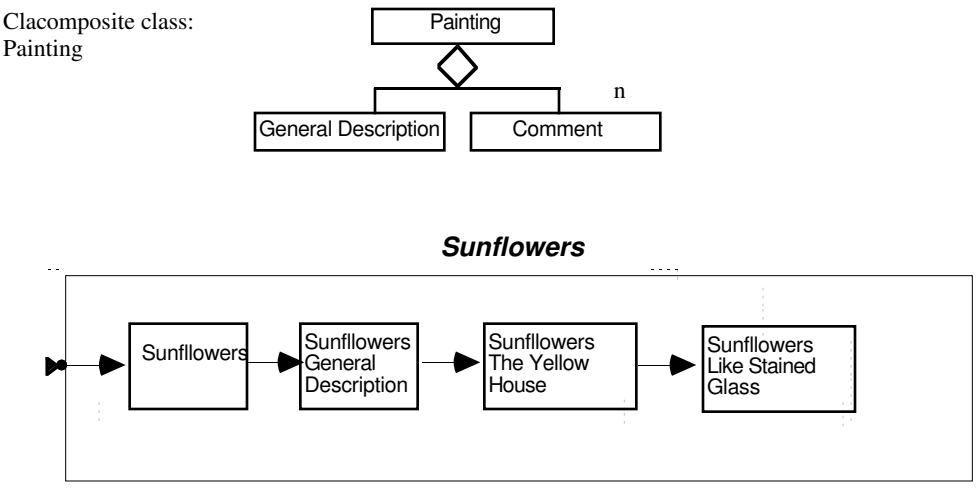


Figure 12. The structure derived context "Painting" , exemplified with the painting "Sunflowers".

Going back to the link derived context shown in figure 10 above, we can now see that each of the elements in the "Paintings by Van Gogh" context is in fact another context, the structure derived context of the corresponding painting. A sequential path can be defined through the collection of paintings in that context by concatenating the sequential path inside each of the composite nodes. Figure 13 shows the navigational context card for the "Painting" structure derived context. Notice that in this example the user can only navigate in this context starting at the root node; the application does not allow the user to jump directly to a "Comment" node when browsing through

paintings, although this is actually allowed when accessing a commentary node from within a guided tour.

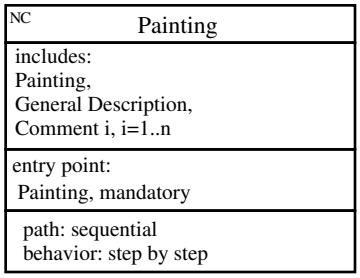
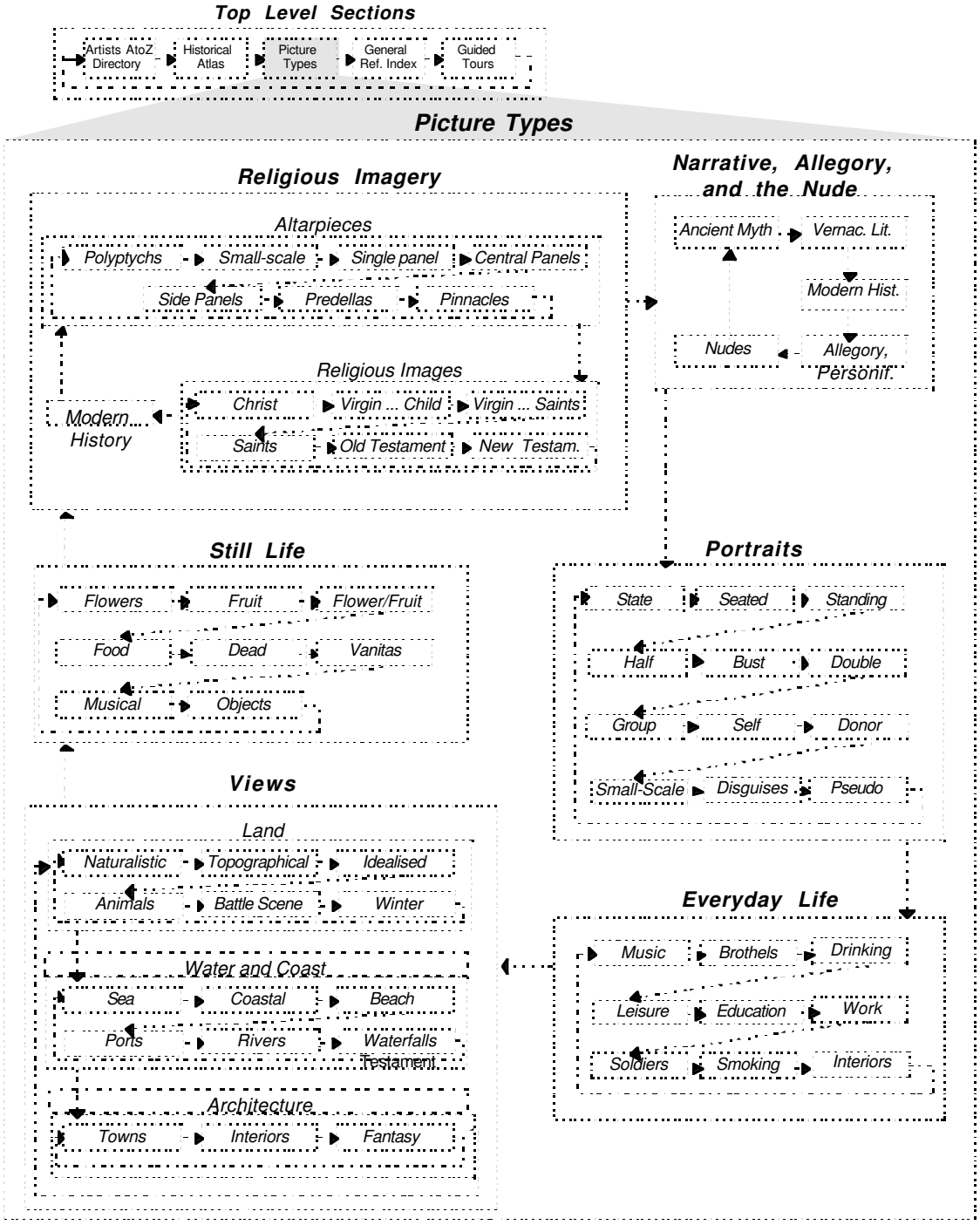


Figure 13. Navigational context card for the structure derived context “Painting”.

3.3.5 Index

One of the important parts of a hypermedia application are the access structures – indexes and guided tours – it provides. Guided tours, as we have seen, are specified using arbitrary navigational contexts. An index navigational contexts is derived from an index structure, and contains the elements pointed to by the index. This type of navigational context has a default behavior, which allows navigation from the index to any element, and from any element back to the index. Sometimes an additional behavior may be specified, whereby the user may also navigate sequentially through all the elements pointed to by the index; in this case one obtains a mixture of guided tour and index

The specification in figure 6 shows an example of an index derived navigational context, computed directly from the specification of the guided tour, and its corresponding navigational context. Figure 14 shows another example, where elements are themselves indexes. This inclusion entails nesting of navigational contexts, where the innermost elements are indexes that contain the paintings, as illustrated by the screen “Still Life: Flowers”.


Top Level Sections
Artists AtoZ Directory
Historical Atlas
Picture Types
General Ref. Index
Guided Tours
Picture Types
Religious Imagery
Altarpieces
Polyptychs
Small-scale
Single panel
Central Panels
Side Panels
Predellas
Pinnacles
Religious Images
Christ
Virgin ... Child
Virgin ... Saints
Saints
Old Testament
New Testam.
Modern History
Narrative, Allegory, and the Nude
Ancient Myth
Vernac. Lit.
Modern Hist.
Nudes
Allegory, Personif.
Still Life
Flowers
Fruit
Flower/Fruit
Food
Dead
Vanitas
Musical
Objects
Portraits
State
Seated
Standing
Half
Bust
Double
Group
Self
Donor
Small-Scale
Disguises
Pseudo
Views
Land
Naturalistic
Topographical
Idealised
Animals
Battle Scene
Winter
Water and Coast
Sea
Coastal
Beach
Ports
Rivers
Waterfalls
Architecture
Towns
Interiors
Fantasy
Everyday Life
Music
Brothels
Drinking
Leisure
Education
Work
Soldiers
Smoking
Interiors

Still Life: Flowers
PICTURE TYPES
Flower pieces originated as adjuncts to devotional paintings or portraits. By the 17th century in both Italy and Northern Europe they were a popular independent category, sometimes retaining a degree of symbolism, usually signifying transience.
Like landscapes, earlier flower pieces were often assembled from separate sketches from the life. That is why we may find in them flowers from different seasons.
19th-century flower pieces, similar in form, are more likely to have been painted from an actual bouquet.

Figure 14. An example of an index navigational context, “Types of Painting”, illustrating nesting of navigational contexts.

Figure 15 shows an example of navigational context card for the “Types of Painting” index.

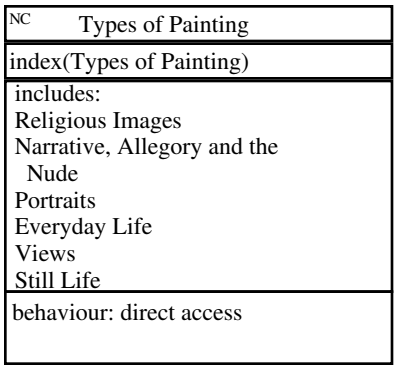


Figure 15 The navigational context card for index “Types of Painting”.

3.3.6 Session

Many applications allow the user to browse through dynamically collected information, or add information dynamically during its execution. For example, the user may want to see all the nodes that have already been visited; or may want to see all the notes she has made during navigation. A session based navigational context is built dynamically, its elements being added by evaluating some condition at runtime. Figure 16 shows a specification of a “history” session based navigational context; note that this context may be used for a simple (albeit quite frequent) type of backtracking.

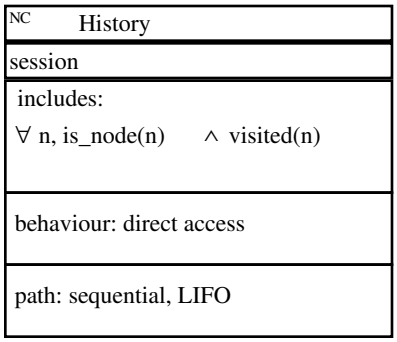


Figure 16 A navigational context card for a session based navigational context containing the history of visited nodes during a session.

3.4 Context Classes

In many situations, the basic information in a node must be complemented with additional information specific to the context in which the user is in. For example, in a guided tour the author may wish to explain why a particular piece of information is being presented; clearly, this should only be presented when the user is browsing that information in the context of the guided tour. In other situations, certain information contained in a node may be redundant, depending on the context in which it is being presented. For example, the general description of an artist could include some information like “He was one of the great impressionist painters” but, if the user is

viewing a section about the great impressionist painters and navigates to the description of this artist, this piece of information would be redundant, and should not be presented. Situations like these are modeled using the concept of Context Classes.

A Context Class is defined as a class that complements the definition of a navigational class with the addition of attributes depending on some context. This means that a context object adds attributes to a navigational object, such that these attributes are only visible within one or more specified contexts. The definition of a context class, in addition to the normal items in a class definition, includes a definition of its scope – the set of navigational contexts in which it is valid – and of its participants the set of objects in those contexts it will modify.

A context object may also be defined independently of a navigational object, but still maintaining the property of being visible only within its scope. Figure 17 shows a context class card used to specify a context class; "Scope" contains the names of the navigational contexts this context class is defined for, and "Part-of" contains the names of the object classes (if any) in these contexts to which this context class adds to.

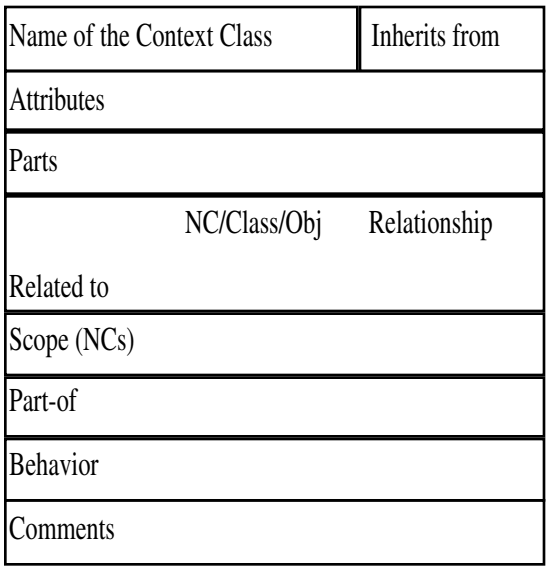


Figure 17. A context class card.

The guided tour "Composition and Perspective" in Art Gallery provides an example of context classes. When discussing a painting within this tour, two new pieces of information are added, both in the form of narration: "concept", which explains a certain concept related to the theme of the tour; and "illustration", which shows how the current node illustrates that concept. When accessing this same painting outside this tour this information is not accessible to the user. Figure 18 show the context class card for this example.

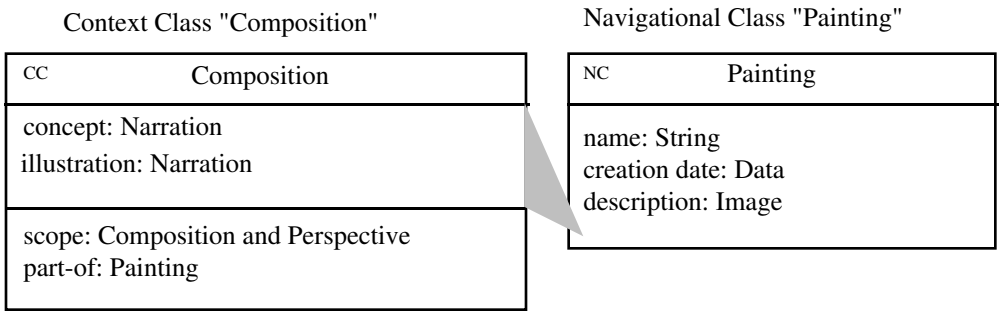


Figure 18. An example of context class:, “Composition”. When accessing “Painting” inside navigational context “Composition and Perspective”, attributes “concept” and “illustration” are accessible.

Context objects may be created or altered during runtime. Each attribute of a context object has a flag specifying if it is “read only” or “read/write”; the default flag is “read only”. This allows modeling applications in which a user is allowed to add information to the hyperbase, e.g., notes, which can be later browsed using, for instance, session based navigational contexts, presented in section 3.3.6.

Figure 19 shows an example of such notes, where a session based navigational context is defined to contain all nodes of context class “Notes on Historical Figures”. This context class adds read/write attribute “note” to class “Historical Figure”, as illustrated by the two instances “Notes on Christopher Columbus” and “Notes on Isabel I”.

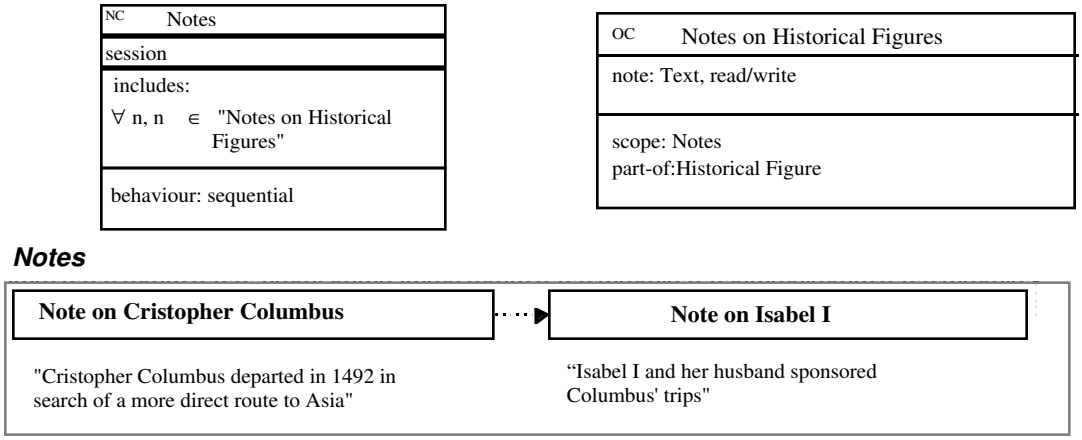


Figure 19. An example of a context class “Notes on Historical Figures” with a read/write attribute, which is added to “Historical Figure” objects in session based navigational context “Notes”.

3.6 Navigational Schema

The navigational structure of an application is specified in a schema, named navigational schema, that is made out of all navigational contexts defined for the application. Besides the individual navigational context definitions, it shows also the links connecting contexts, thus allowing a quick understanding of the navigational space available to the user. Figure 20 shows the navigational schema of the Art Gallery application, with only the names of the navigational contexts shown, and their interconnections. The navigational contexts corresponding to the picture types (see Figure 14) are not shown, and neither are all of the Guided Tours.

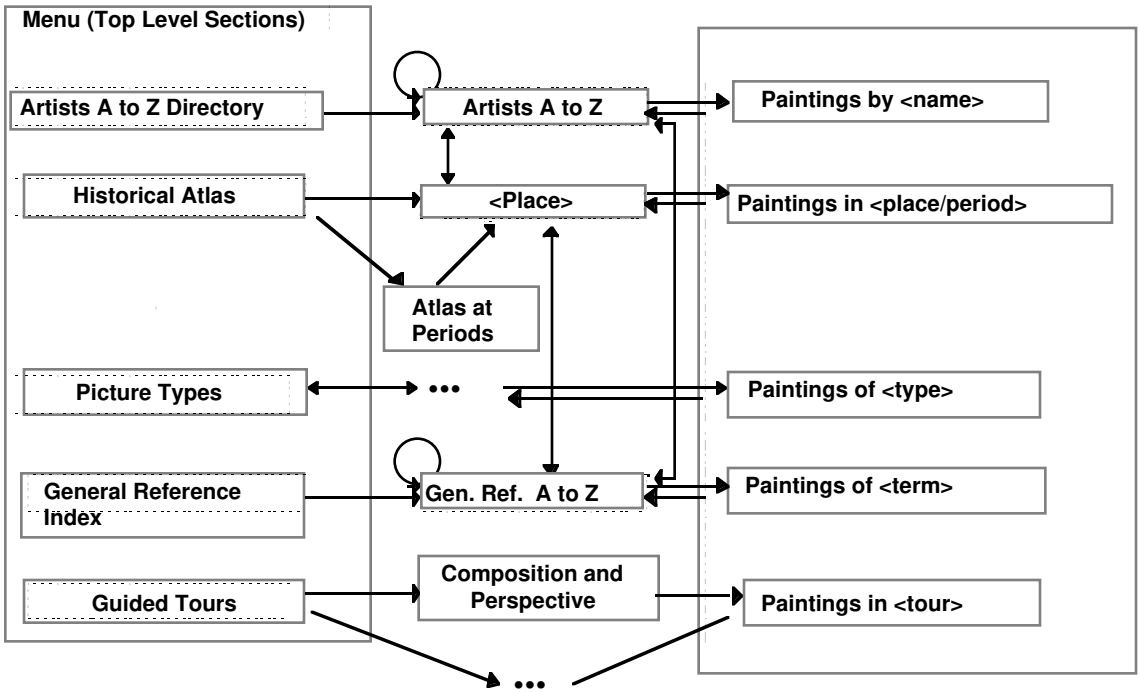


Figure 20. The navigational schema of the Art Gallery application; only the names of the navigational contexts and their interconnections are shown.

3.7 Semantics of Navigation

There are several aspects of the dynamic behavior of the application that must be specified; in the navigational design step one must define how the navigational space (i.e., the set of accessible navigational objects) changes during navigation, and what control actions the user has at each step. At this level, there is little concern with the actual appearance of nodes and controls at the interface, which will be handled at the next step in the design process.

The specification of a navigational context contains a path and a behavior description. The path specification establishes a sequence through the elements of the navigational context, that may be used by the reader when browsing within the context. Following [Zellweger 89], paths may be sequential, branching and conditional.

The simplest kind of path is the sequential, where the elements of the context are organized in a sequence. The specification may include an ordering according to some condition on the attributes of the elements, or may have an arbitrary ordering (which really means the ordering may be established based only on implementation concerns). The branching path establishes a hierarchical path connecting the elements in the context; this type of organization is modeled using nested contexts (see section 3.3.5 for an example).

A conditional path includes branching points where conditions are evaluated and the result determines the path to be followed. Conditional paths include procedural paths, where paths may be a part of another path; variable paths, where the next link to be followed is computed dynamically; and parallel paths, where many links may be followed simultaneously, possibly with some synchronization between them. A path in a navigational context specification may, in principle, be a conditional path, although we have not yet defined a notation for this type of specification.

The behavior specification for a navigational context has two components, the first describes the control of the navigation within that context, and the second describes the types of restrictions that apply.

The control component may be automatic or step by step. When the control is automatic, the user is guided automatically by the application (i.e., has no control); this may be useful in some types of presentations, for example. The step by step control allows the user to control when the next navigational step will be taken.

The navigation may be restricted in four ways: free, direct access, exclusive; and partial, whose meaning depends on the type of control. For automatic control, only the "exclusive" restriction mode applies, since the only links allowed to be followed are the ones defined in the navigational context. For step by step control, the "free" restriction mode allows the user to access any node in the context, and follow any link from the chosen node; the "direct access" mode allows access to an element of an index, but only allows following a link back to the index; the "exclusive" mode allows following the context links (i.e., the path through the elements), but does not allow following other links from each element; the "partial" mode allows following only direct links out of the node, but not the inherited ones (for example, in a composition).

To exemplify the behavior specification, a guided tour that allows the user to follow any link out of an element in the tour is "step by step, free" navigation. In other applications, to be able to follow a link out of a node in a guided tour the user must exit the tour first; this is an example of "step by step, exclusive" navigation.

The "Composition and Perspective" guided tour in Art Gallery (see figure 6) is an example of "step by step, partial" navigation, since all links out of a node - which is either a "Comment" or a "General Description" of a "Painting" – in the tour may be followed; but not those links inherited from the root of the composite object – the "Painting" itself.

There is another dimension that characterizes the style of step by step navigation. There are two basic styles reported in the literature – the more common graph-based style, found in most hypermedia applications, and the set-based style [Parunak 91]; both styles are allowed by the navigational contexts. Under the set-based navigation, after browsing a node, the user may continue examining other nodes in the same context, or may choose another context to which that node also belongs; this style of navigation is particularly suited for taxonomic reasoning.

4. RELATED WORK

As stated in the introduction, navigational contexts are inspired in the Nested Context Model (NCM) [Casanova93]. In that work, contexts are proposed as a basic structuring mechanism for hypertext systems (as opposed to applications), allowing a distributed implementation. The emphasis is on basic system architecture, and little is said about application development.

The closest work to navigational contexts is the concept of "collections" as reported in [Garzotto 94]. Although not cast in an object-oriented framework, collections play a similar role as navigational contexts. There is no close equivalent of navigational objects as mappings of conceptual objects – in fact, there is no notion of separate conceptual and navigational designs–, and collections do not allow object extension as is possible with context objects.

Collections are similar to navigational contexts in allowing attributes do be associated with the collection (containing for example meta-information). Collections may be formed in six basic ways, some of which have a similar or equivalent context type: "pick up" is equivalent to arbitrary contexts; "built-in" is equivalent to class based; "link based" is similar to link based navigational contexts, but with a different rule for forming the context out of the links; "session based" are equivalent to session based contexts. The possible navigation semantics for collections are a subset of the navigational semantics allowed by contexts.

5. CONCLUSIONS

The separation of conceptual, navigational and abstract interface design postulated by OOHDM allows a clean separation of concerns during application development. Navigational contexts provide a concise way to specify the structure of the navigational space of a hypermedia application, taking advantage of the object oriented style. Furthermore, this specification can be easily integrated with the next design step, the abstract interface design (see [Rossi 95]).

We have used the model described in this paper to a number of real life applications, some which where developed without using it (for example, Microsoft's Art Gallery, Ancient Lands, Multimedia Beethoven), and others which were developed from the beginning using it (for example, a hypermedia interface to a database of the collection of Candido Portinari, a famous Brazilian painter; several information kiosks for a large telecommunications company; an educational software for training grade school teachers in social studies). In both cases is has proved to be adequate, allowing either a clear analysis (as can be gleaned from the examples of Art Gallery shown in this paper), or a fast design cycle of new applications.

We are continuing our research on navigational contexts, looking in various directions: formal specification of navigation semantics, including multimedia synchronization aspects; intentional specification of context elements, whereby the user specifies conditions under which an element is to be included in an context; specification of integrity constraints for contexts; specification of various kinds of backtracking semantics; extension for cooperative group development.

We are also developing computer based tools to aid in the development of applications using OOHDM, eventually arriving at a Computer Aided Hypermedia Development Environment. Among other alternatives, we are exploring the use of an object oriented DBMS and associated tools as a basis for such an environment.

6. REFERENCES

[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.

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

[Garzotto 94] Garzotto, F.; Mainetti, L.; Paolini, P.; "Adding Multimedia Collections to the Dexter Model", Proceedings of ECHT'94, Edinburgh, 1994.

[Parunak91] Parunak, H. "Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning", Proceedings of Hypertext '91, San Antonio,. Dec. 1991.

[Rossi95] Rossi, G.; Schwabe, D.; Lucena, C.J. P.; Cowan, D. D., "An Object-Oriented Model for Designing the Human-Computer Interface Of Hypermedia Applications", Technical Report, Departamento de Informática, PUC-Rio, 1995. Also submitted to the International Workshop on Hypermedia Design, Montpellier, 1995.

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

[Schwabe94] Schwabe, D.; Rossi, G. "From Domain Models to Hypermedia Applications: an Object-Oriented Approach", Technical Report MCC 30-94, Departamento de Informática, PUC-Rio. Expanded version of

position paper presented at the “Methodologies for Designing and Developing Hypermedia Applications” workshop, ECHT’94.

[Schwabe95] Schwabe, D.; Rossi, G. “Building Hypermedia Applications as Navigational Views of Information Models”. Proceedings of the 28th Hawaii International Conference on Systems Science, Maui, Jan, 1995. Also Technical Report MCC 41-94, Departamento de Informática, PUC-Rio.

[Zellweger89] Zellweger, P. “Scripted Documents: A Hypermedia Path Mechanism”, Proceedings of Hypertext ‘89, Nov., 1989.