Pattern Systems for Hypermedia

Alejandra Garrido (*)[1], Gustavo Rossi (*) and Daniel Schwabe (**)[2]

(*) LIFIA, Facultad de Ciencias Exactas. UNLP. Argentina.

E-mail:[garrido,gustavo]@sol.info.unlp.edu.ar

(**) Depto de Informática. PUC-Rio, Brazil.

E-mail: schwabe@inf.puc-rio.br

Gustavo Rossi is also at CONICET and UNLM.

Abstract

The hypermedia domain is currently receiving much attention, mostly due to the new generation of open systems, i.e., those that allow the connection among applications in the same or different machines (intranets) or those that publish the interface of an application in a WWW's browser. However, current applications in this domain are not taking profit of all benefits that characterize hypermedia applications, and maintenance is very difficult to achieve. Building large hypermedia applications is a hard task, and although there exist many hypermedia design methodologies [Schwabe96, Izakowitz95], we also need design patterns that convey the expertise in the domain. We present in this paper three pattern systems intended to provide guidance for different aspects of hypermedia applications; the first one is concerned with the development of software support for hypermedia in the context of object-oriented applications; the second one deals with organizing the navigational structures in a clear and meaningful way for intended readers and the third one comprises patterns for building effective graphical interfaces.

Introduction

Applications inside the Hypermedia domain space can be organized in two categories, namely Hypermedia Systems and Hypermedia Applications. The second category is itself sub-divided according to different design concerns, as outlined in Figure 1. Accordingly, we present three pattern systems : 1) Patterns for hypermedia systems, 2) Navigational design patterns and 3) Interface patterns. These categories were defined based on our experience in designing hypermedia applications by applying object-oriented methods and patterns [Rossi96a, Rossi96b, Rossi97], consistently with our goal to present the set of patterns according to the different tasks and interests of users.

It is worth noting that all these patterns can be equally applied both to "conventional" hypermedia applications, to static web sites and to sophisticated applications running on web browsers accessign web servers (e.g. intranet applications). We assume that a user of these patterns has background knowledge about the basic concepts of the hypermedia model, i.e.,

[1] Now at Computer Science Department, UIUC, email: garrido@chip.cs.uiuc.edu

[2] Partially supported by Conselho Nacional de Desenvolvimento Científico - CNPq - Brazil

understands what a node is, how they are connected by links and what the role of anchors in nodes is.

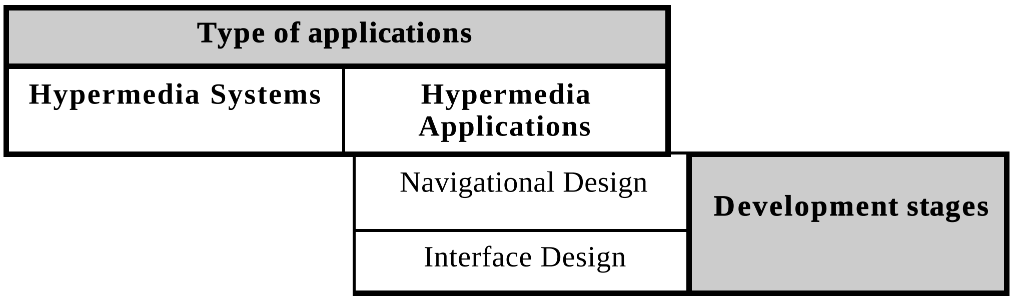


Figure 1: Hypermedia patterns space

Pattern systems

1. Patterns for hypermedia systems

This category encompasses design patterns that can be used to construct hypermedia systems, i.e., the environments used to build hypermedia applications, or to extend conventional applications with hypermedia functionality [Garrido96]. They have a specific audience: hypermedia system developers. Although these patterns were discovered in object-oriented environments, it is not difficult to apply the core of the solution to other design strategies. Many hypermedia systems and some well known hypermedia design methodologies use some of the patterns in this category, as noted in each pattern description.

An outline of pattern's interaction for this category is shown in Figure 2; arrows indicate patterns that typically used in conjunction with one another.

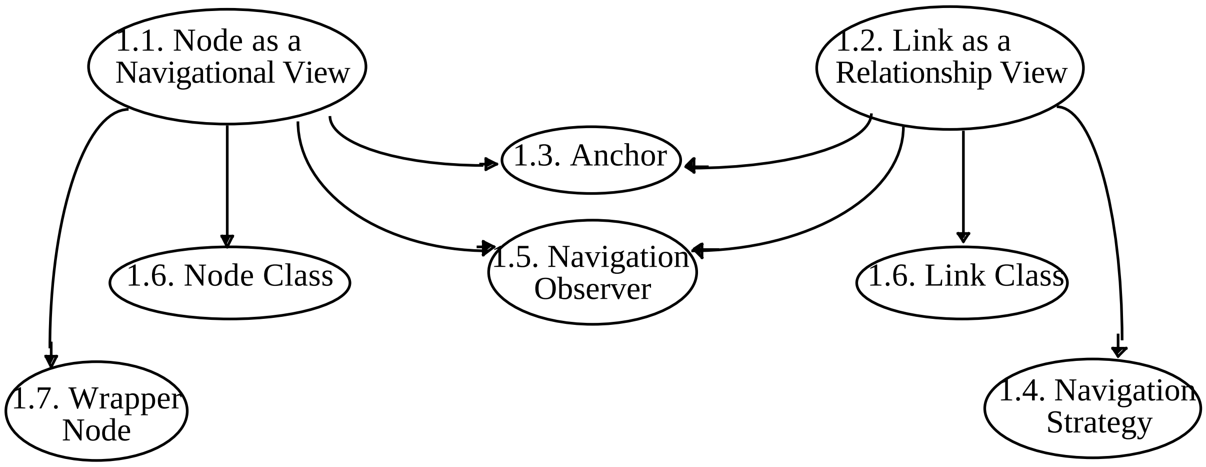


Figure 2: Interaction between patterns for hypermedia systems

1.1. Node as a Navigational View

Problem: How to add navigation capabilities to the components of an object-oriented (OO) application, therefore adding hypermedia functionality to the application ?

Forces:

- We want to add hypermedia funtionality to existing applications
- Original interface behavior must be preserved, and hypermedia behavior in must be added
- Modification of objects in the original application is undesirable
- Redefining the original GUI to include hypermedia capabilities (bookmarks, backtracking, history maintenance, to name a few) is undesirable and many times unfeasible
- It is difficult to include dynamic links to the existing interface

Solution: Define a navigational layer between the application to be enhanced and its graphical interface, build up of object's observers that are called *nodes*. Implement the navigational behavior in nodes. Then define each node's GUI adding means of activating node's behavior.

This solution implies defining a hypermedia node as dependent of an object or group of objects, thus separating hypertext functionality from the behavior of the application and its interface.

Known uses: The OOHDM methodology defines the concept of Node as a navigational view over a conceptual model [Schwabe96]. In [Bieber95] the authors present an architecture for adding hypertext functionality, where nodes are defined as representations of the objects of interest to the application. A similar approach has been used in the Devise Hypermedia Model [Gronbaek94b].

Related patterns: Navigation is performed by Links (1.2) and activated by Anchors (1.3). Nodes are Observers [Gamma95].

1.2. Link as a Relationship View

Problem: How to represent the relationships among the components of an application in a hypermedia view of those components, and allow to navigate those relationships?

Forces:

- Semantics of relationships in the application should be maintained
- Application objects should not be burdened with navigation actions (deactivating the origin node, recording the navigation step, activating the destination node, etc...)
- Relationship information should be maintained separately from the objects themselves

Solution: Define links as first-class citizens as well as nodes; make each link represent a relationship of interest (in terms of navigation) between two or more objects, which must also have nodes as navigational views over them. Let links be responsible for the navigation process described above. During this process, the destination of a link should be computed on demand by querying the objects involved in the relationship, and obtaining the node that “views” the target object (in the sense described in 1.1.).

Known uses: In [Lange94] the EORM method is presented for extracting links from relations among objects in a storage layer. The OOHDM methodology defines Link as the navigational representation of relations in a conceptual model [Schwabe96]. In [Bieber95], Links represent relationships derived from an underlying model.

Related patterns: Links are activated through Anchors (1.3) in the origin Node (1.1).

1.3. Anchor

Problem: How to activate links from within a node?

Forces:

- Some mechanism should be provided in the interface of a node in order to activate each link departing from it
- The relationship that a particular link maps can be from the entire source node to the destination;
- The relationship that a particular link maps can be from an aspect of that node to the destination;
- The relationship that a particular link maps can be global to the application, i.e., from any node describing some concept, do the destination.
- If link markers are not maintained with the data, which is typically a better implementation [Davis94], its location within an aspect must be computed

Solution: Define “Anchor” as the representation of a link within a node, and make it responsible of link activation. An anchor relates a link marker to the link itself, so the link is independent of any means of activating it.

Anchors could be sub-classified as DHM proposes [Gronbaek94a], in : *whole-component anchors* as those that are not dependent of a node’s content, *marked anchors* for those links defined from a node’s aspect and activated through a link marker attached to a node’s content, and *unmarked anchors* for those anchors which location inside a node’s content is computed on demand [Davis94]. Marked and unmarked anchors actually delegate to the link marker the identification of the position dependent of the data type.

Known uses: Dexter Reference Model [Haslasz94], hypermedia design methodologies as HDM [Garzotto91], OOHDM [Schwabe96], and hypermedia systems as Devise Hypermedia Model [Gronbaek94b].

Related patterns: Anchors activate Links (1.2.) from the origin Node (1.1).

1.4. Navigation Strategy

Problem: how to allow both static and dynamic computation of a link destination to coexist in a hypermedia system?

Forces:

- The semantics of relations denoted by certain links depend on knowledge of the domain of application
- It it is often impossible or not cost effective to automatically determine the presence of links, so the designer must be allowed to statically insert hand made links
- It is desirable to allow the presence of links to be determined automatically via some computation, normally based on a node's contents, as in the case of large textual bases where the occurrence of a given string denotes the presence of a link.
- In many cases it is desirable to allow both kinds of links to coexist.
- Some hypermedia systems also allow links to remain dangling , i.e, without the specification of their endpoint.
- Some systems allow the creation of the destination node upon link activation.

Solution: Decouple links from the way their destination is obtained, defining a separate hierarchy of navigation strategies. Strategies should include: fixed endpoints, computed endpoints, dangling endpoints and lazy end-point creation. For further explanation, see [Rossi96].

Related patterns: This pattern is used when the pattern Link (1.2) is used.

1.5. Navigation Observer

Problem: How to create a perceivable record of the navigation process ?

Forces:

- Hypermedia applications should record the state of the navigation in a user-perceptible way.
- As navigation progresses, this record must be updated automatically.
- It should be possible to define the history of navigation in different ways, e.g., an ordered list of visited nodes, an ordered list of visited links, a colored graph, etc.

Solution: Define a History object for each session in which the hypermedia is opened. History objects collect the information about navigation by interacting with nodes and links. Different histories should be created for different sessions in a concurrent environment. Moreover, the GUI with which a particular history will be perceivable should be defined separately and so be able to be changed independently.

Define history's GUIs as observers (pattern Observer) of history objects. For further explanation, see [Rossi96].

Related patterns: this pattern is used when the patterns Node (1.1) and Link (1.2) are used.

1.6. Node Class, Link Class

Problem: How can node types be defined, according to the classes of an OO application that is extended with hypermedia functionality, or according to types defined by a data base, thus automating or easing node creation?

Forces:

- The concept of entity type or node class as the representation of a group of nodes with the same properties, or aspects, and related to nodes of other types in the same way is present in many hypermedia design methodologies (HDM, OOHDM, RMM).
- Creating a new class for each group of similar nodes would duplicate the classes of the extended application, giving rise to a class explosion
- One node should be automatically generated for each object of the corresponding (conceptual) class when a node class is defined as a view over (mapping) one or more classes of the underlying application,. If a new object of an application class is created, a node of the corresponding node class should be created.
- The same forces apply to links classified by link classes, the latter representing a relationship between a pair of node classes.

Solution: Define a class NodeClass whose instances act as templates (classes) for Nodes. Using this pattern we decouple Node from NodeClass; Node contains basic hypermedia functionality (reacting to anchor activation for example) while instances of NodeClass act as the template for data and behavior. Make instances of NodeClass responsible of node's creation.

In the same way, define LinkClass as a class whose instances act as Type-objects for instances of Link.

Known uses: Design methodologies as HDM [Garzotto91], OOHDM [Schwabe96], RMM [Isakowitz95], and hypermedia systems as [Bieber95]. Link classes are defined in the same way in [Lange94].

Related patterns: NodeClass and LinkClass can be seen as an instantiation of the Type-Object pattern [Johnson97] for the hypermedia domain. However, more than providing a type for Nodes and Links, those classes are responsible for the creation and maintenance of the first ones. The creation is performed using the Prototype pattern [Gamma95].

In the context of the pattern system, 'Node Class' is highly related to 'Node as a Navigational View' (1.1), and 'Link Class' to 'Link as a Relationship View' (1.2.).

1.7. Wrapper Node

Problem: How can we add navigational behavior to existing GUIs?

Forces:

- Many existing applications have a GUI specially adapted to perform specific tasks, and it may not be feasible to modify it.

- We may need to extend such kind of interface with hypermedia functionality.
- Creating a new interface by sub-classifying the existing one implies redefining the hypermedia protocol for each interface of each node, which is not a good solution.
- It is desirable to integrate the existing interface within the hypermedia space in some way that it can also be used outside it.

Solution: The solution is to decorate the interface with a "node wrapper". The node wrapper will capture those interface actions dealing with navigation and activate it, passing to the decorated interface all other actions. The node wrapper has an associated group of widget's wrappers for each widget in the decorated interface. Those widget's wrappers are hypermedia-aware, that is, anchors for links can be defined and activated within their data.

The main difference among nodes and wrapper nodes is that in the former we do not suppose an existing interface, and thus the interface for them is built up from hypermedia-aware widgets. While Nodes as navigational views resemble the Observer Design Pattern, Node Wrappers resemble the Decorator design pattern.

Known uses: This pattern is used by the Microcosm system [Davis94] to provide hypermedia functionality to third party applications such as word processors, spread-sheets, CAD editors, etc.

Related patterns: Decorator [Gamma95] and Node as a Navigational View (1.1)

2. Navigational design patterns

These patterns are meant for hypermedia application designers, regardless of the system they are using for the implementation. For this reason, these patterns have a more abstract solution than the patterns in the previous category, although they do not lack accuracy in describing the elements involved and their necessary interactions. Patterns in this category help in organizing the navigational structure of a hypermedia application in a clear and meaningful way for the intended readers. They address recurrent problems whose solution determines the degree of success of hypermedia applications. They do not care about the final look of the elements in an interface but about the organization of navigation through the hypermedia application's components.

Unless otherwise specified, the issues discussed in this pattern system are widely proposed by design methodologies and papers in the field, as [Nielsen97]. Multiple examples can be found in the World Wide Web.

Pattern 2.1 helps determining the extent of a node; 2.2 and 2.3 address ways of creating nodes and links; 2.4 helps in organizing the navigation space; and 2.5 shows how to help the navigation process.

2.1 Node as a single unit

Problem: How do you decide the extent of a node?

Motivation: A node should encompass a self-contained "unit" of information, that should make sense for a set of users performing a set of tasks in a given domain.

For example, it is quite common to find web pages that are very long, including different merged topics, some of which may not be relevant to the task at hand. For example in http://www.cs.brown.edu/memex/, a single page shows a big picture at the beginning, then information of what Memex and Beyond is about, then a global index of topics is presented, a description of what is still missing comes after that, details about the kinds of navigation that appear in the web site, etc. Conversely, fragmenting a topic ("unit") over several pages makes reading and printing more difficult, as the reader must navigate from one fragment to the next in order to see the entire "unit".

Forces:

- The number of different "topics" is large
- The amount of data in a topic is large
- The reader must be presented with a unit that makes sense to him, to help him accomplish his tasks
- The reader should not have to navigate over too many nodes to obtain the relevant information for a task
- It is difficult to decide on the number of related objects that must be observed through a node in an OO application that has been extended with hypermedia functionality.

Solution: Make a node a single unit, self-contained entity of information, focus on a certain topic [Nielsen97]. All data that is relevant for that entity should be included the same node. When extending an OO application, you will generally map one application object to one node. Nevertheless, a whole entity is usually represented by more than one object, i.e., the information is split among several objects that have to work together (e.g., an object and its state object). In this case it is better to map all strongly related objects to the same node, if it is meaningful to see all the information together.

If the data to show is brief, make it an annotation to be accessed from a global link.

Related patterns: Node as a Navigational View (1.1) should be applied before Node as a single unit in case of the extension of an OO application. Use the Interface on Demand pattern (3.1) to organize data in a node's interface. In this pattern system, this pattern gives rise to Node Creation Method (2.2).

2.2 Node creation method

Problem: When is it better to create nodes statically, and when is it preferable to create nodes dynamically ?

Motivation: In many applications, the contents of a node do not change, or change very little, as is the case of, most web pages found in the WWW today. On the other hand, in functionally rich applications, where information is captured from databases, nodes must be defined dynamically.

Forces:

- Aesthetic aspect of nodes are better handled by static, hand-crafted nodes
- Applications accessing data residing on databases are better handled with dynamically created nodes
- Dynamic node creation may be computation-intensive

Solution : Nodes must be dynamically created when the application's data and functionality will be constantly updated, when readers can modify, create, or compose new nodes, and when instant update is needed. Otherwise, when there is no underlying application, and the set of nodes is limited and fixed, manual creation is the most acceptable solution.

In some cases the computation of nodes on demand (when they are the destination of a navigation step) can be a heavy process, and so it is better to pre-compute the nodes by running the defining queries at definition time, and storing the results. This approach is only feasible when data on the data base changes at a low enough rate.

Related patterns: Nodes are computed when they can be defined as Navigational Views (pattern 1.1). Also, this pattern is highly related with Link creation method (2.3).

2.3 Link creation method

Problem: When is it better to define static links, and when is it preferable to create links through computations?

Motivation: Similarly to 'Node creation method', sometimes is hard to decide the pros and cons of defining links by hand or by means of a computation.

Forces:

- Invariant relationships in the problem domain are best represented by static links
- Dynamic link generation may be either unfeasible or too expensive given weakly structured source data
- Hand creation of links is more error prone
- Dynamic link creation is natural when nodes are created through queries over data in databases
- Link consistency is easier to maintain for dynamically created links
- Dynamic link creation is more cost effective for large numbers of links of the same type
- Arbitrarily defined links can only be created by hand
- Dynamically created links can result in more links than the reader can comprehend (over-completeness [Thistlewaite97])

Solution: Static links are used in closed, static applications, when maintainability in case of future change is not an issue, and nodes are also statically created. Static links may be used in dynamic applications when the definition of a link-computation is too complex, the endpoint node is very difficult to be changed, or the link is only temporary.

Computed links should be preferred in dynamic applications where new nodes are also created dynamically, data is volatile, and maintainability must be efficiently achieved. Use also dynamic links, even with statically created nodes, when it is possible to define link types, and when the number of links of each type to create is considerable (i.e., the effort require to write a computation to automatically create the links is less than the effort to manually instantiate all links in a relationship between large sets of nodes).

Related patterns: Links are computed when they can be defined as Relationship Views (pattern 1.2). As said before, this pattern is highly related with ‘Node creation method’ (2.2).

2.4 Navigational context

Problem: How to organize the application’s navigational structure, providing guidelines, information and relationships that depend on the current state of navigation, in such a way that information can be better presented and comprehended ?

Motivation: Hypermedia applications usually involve dealing with collections of nodes (e.g., Paintings, Cities, Persons, etc.). These collections may be explored in different ways, according to the task the user is performing. For example, we may want to explore Books of an author, Books on a certain period of time or literary movement, etc., and it is desirable to give the user different kinds of feed-back in different contexts, while allowing him to move easily from node to node.

Suppose for example that in an application about Literature, involving writers, their works, literary movements and genders, we reach “William Shakespeare”, choose to navigate to his books, and then arrive at “Romeo and Juliet”. However, we can also reach “Romeo and Juliet” while exploring Books written in the Romanticism period, or navigating through Tragedies. It is clear that we will explore the same object under three different perspectives; for example while accessing it as a Shakespeare’s work we would like to read some comments about its relationships with other works by Shakespeare (as it relation with “Love’s Labour’s Lost” because both incorporate sonnets into their structure), and also to have easy access to the next book he wrote. Meanwhile, as a Romantic work, it would be fine to read (or see) something about the Romanticism period and be able to access other works on the same movement (perhaps not Shakespeare’s). This means that we will need not only to present the information in a different way in all cases but also to provide different links or indexes.

Forces:

- we want to access *the same* hypermedia component under different contexts;
- we need to group nodes under the same context and navigate through them using *contextual* links;
- it’s impractical and it may shield inconsistency to have different objects to represent the same component but in each different context;
- the current context of navigation is not a concern of the node; for example it is not correct that a node contains contextual links because those links depend on context information, beyond the node’s data.

- web pages usually lose contextual information, and this usually provokes reader disorientation.

Solution: Decouple the navigational objects from the context in which they are to be explored, and define objects' peculiarities as Decorators [Gamma95], that enrich the navigational interface when the object is visited in that context.

Navigational Contexts are composed of a set of Nodes (like Books) and Context Links (links that connect objects in a context). Nodes are decorated with additional information about a particular context and additional anchors for context links. The navigational context may also contain information about the context itself (for example an explanation about Romantic Books) that will be shown in a particular Context Node. That node may provide an index to all nodes in the context or a link to the first one.

A diagram of the interacting elements is shown in Figure 3, where the context node provides an Index to the nodes and also each decorator provides an anchor to the 'next' node in the context.

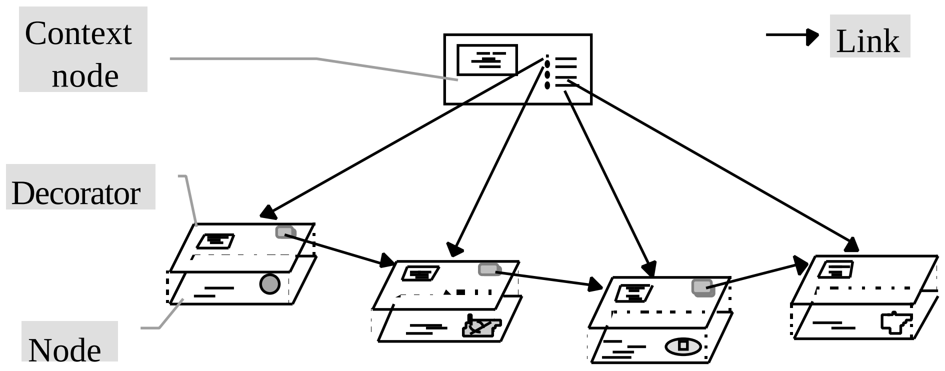


Figure 3: Diagram of Navigational Context pattern

Finding navigational contexts is important because they are high level architectural constructs that help organizing navigation in such a way that we can describe the navigational structure of a hypermedia application not only as a set of nodes and links but also as a set of contexts in which those nodes will be accessed.

Known uses:

Navigational Contexts has been used in many successful hypermedia applications as: Microsoft's "Art Gallery", Portinari [Lanzelotte93], Dorling Kindersley's "The Way Thing Works". OOHDM use Navigational Context as a design concept.

Related Patterns: Node as a single unit (2.1), and Active Reference (2.5).

2.5 Active reference

Problem: How can we provide a perceivable and permanent reference about the current status of navigation, combining an orientation tool with an easy way to navigate to a set of related nodes, at the same or higher level of abstraction?

Motivation: In many hypermedia applications (particularly those involving spatial or time structures) we need to provide the reader with a way to understand where he is and help him decide where to go next. The usual solution would include an index (or other access structure) to the elements we intend the user to navigate. However, this solution will require the user to backtrack from the current node to the index to see where he is or to move to another node, while ensuring that its current position is highlighted in the index. These navigational operations: moving backward to the index and forward to the target may disorient the user.

Forces:

· Navigation through diverse concepts, of diversified themes, at different levels of abstraction or composition, is known to induce readers to become "lost in the cyberspace". Therefore, a reference of current state of navigation is needed.

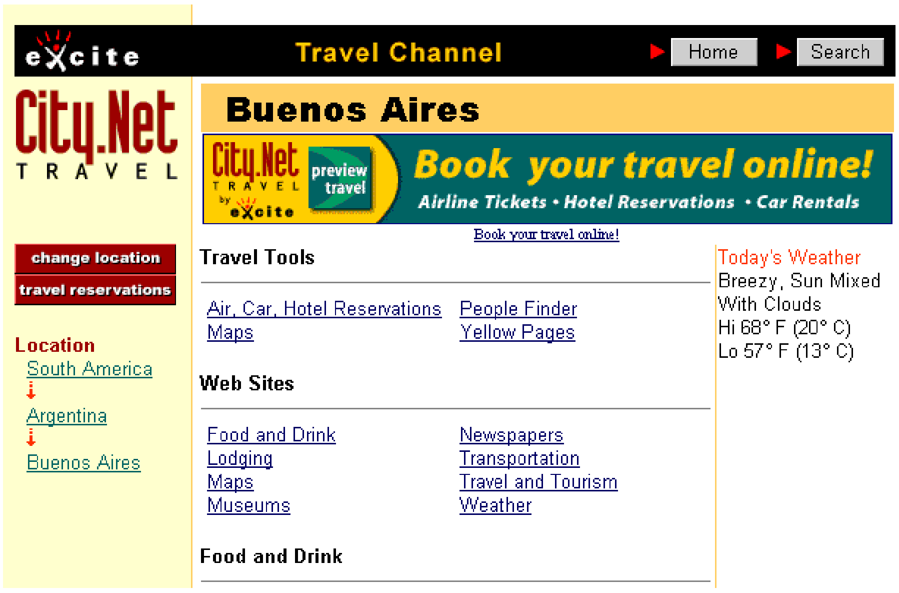
· The solution of using an index to nodes on a certain theme or of a certain type provides a shortcut to arrive to that set of nodes, but once the reader is in one of those nodes, the reference is lost.

· The history of navigation can be of same help, but it usually considers all nodes at the same level of abstraction, without any guidance of composition or theme level.

Solution: A good solution is to maintain an active and perceivable navigational object acting as an index for other navigational objects (either nodes or sub-indexes). This object remains perceivable together with target objects, letting the user either explore those objects or select another related target. In this way we will be able to interact with both the index and the target nodes.

An example applying this solution can be found in City.Net web site (http://city.net). As we can see in Figure 4, the reader has a permanent reference about where he is located while he explores cities in the world (see 'Location: South America ->Argentina ->Buenos Aires' at the left). In particular he can choose to go to the region in which the city is located. The pattern is only used partially here, by showing the geographical hierarchy to the city (even if the user did not visit all those sites). A more sophisticated implementation would be to provide an index to all cities in the region. In http://www.citynet.com/, an index to all cities in a region can be see when a region is selected, though the reference is lost inside a city.

When we use Active Reference the reader has a perceivable and permanent record about the current status of navigation and, in this way, we not only provide an orientation tool but also make them available while navigating the target nodes.


excite Travel Channel Home Search

City.Net TRAVEL

change location

travel reservations

Location

South America

Argentina

Buenos Aires

Buenos Aires

Book your travel online! Airline Tickets • Hotel Reservations • Car Rentals

Book your travel online!

Travel Tools

Air, Car, Hotel Reservations	People Finder
Maps	Yellow Pages

Web Sites

Food and Drink	Newspapers
Lodging	Transportation
Maps	Travel and Tourism
Museums	Weather

Food and Drink

Today's Weather
Breezy, Sun Mixed With Clouds
Hi 68° F (20° C)
Lo 57° F (13° C)

Figure 4: Example of Active Reference pattern in http://city.net/countries/argentina/buenos_aires/

Another interesting example can be seen in Le Louvre CD-Rom, where the user can explore different rooms in the museum from a visual index as we show in Figure 5.

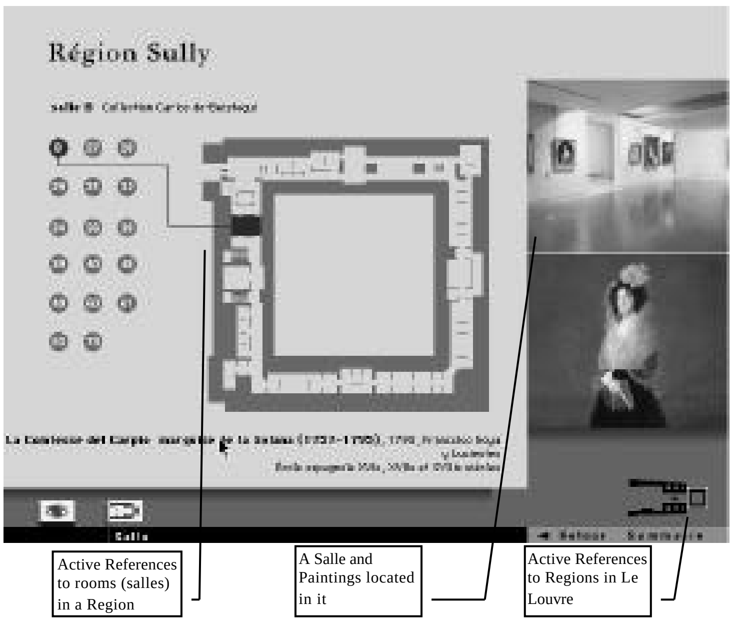

Région Sully
Active References to rooms (salles) in a Region
A Salle and Paintings located in it
Active References to Regions in Le Louvre

Figure 5: Active Reference in Le Louvre

Notice that the strategy in this instantiation of Active Reference is rather different with respect to the one in City.Net. In this case the user can see the whole navigational space: rooms in Region Sully and the whole museum and can select a new region or a new room in the current region, and explore it. Though this is only one possible way to navigate through rooms and regions, it provides the user with a complete sense of where each room is located. In this example the user could even get more information about a region (using the Interface on Demand pattern) or about a painting, navigating to that painting.

Known uses: In Microsoft's "The ultimate Frank Lloyd Wright" it has been applied to show his buildings in different states in the USA. In Microsoft's Multimedia Beethoven we can access an active reference of the 9 Th. Symphony. In Grolier's Encyclopedia the reader has permanent access to a dictionary and to nodes accessed from the dictionary. This pattern is used in the web as seen in the example above (City.Net - http://city.net).

Related Patterns: Active References are created inside Navigational Contexts (2.4).

3. Interface patterns

These patterns are meant for hypermedia GUI designers. They are abstract and so independent of the environment used for the implementation.

Graphical interface design is a complex task, mainly involved with finding the right combination of elements (both in their number and in their spatial relations), in such way that those elements interact for an effective presentation of the information. This pattern language could be also used outside hypermedia applications, in the broader context of GUI design.

Patterns in this category can be briefly described as follows. "Information on Demand" helps with the organization of information that does not fit on one screen; "Information-Interaction Decoupling" and "Information-Interaction Coupling" help to organize the relative position of control features with respect to other information; "Behavioral Grouping" tells you how to organize several control objects so the user can easily understand their meaning; "Behavior Anticipation" and "Process Feedback" help in showing the user the results of control actions.

3.1. Information on Demand

Problem: How to organize the interface in such a way that we can make perceivable all the information in a node, taking into account both aesthetic and cognitive aspects?

Motivation: We usually find ourselves struggling to decide how to show the attributes and anchors in a node. Unfortunately, the screen is usually smaller than what we need and many times we cannot make use of other media (such as simultaneously playing an audio tape and showing an image) either for technological or cognitive reasons.

Forces:

- A node has an amount of information to be perceived by the reader that does not all fit together in one screen , or may distract the user's attention (for example, an audio recording);
- Scrolling is often not acceptable because: firstly, the reader doesn't get a overall view about what he will find in that node; he will have to scroll all the way down to see if there is something that interests him or not; secondly, and following from the previous problem, we are not giving the reader the freedom to choose the attributes of the node he is interested in;

- Partitioning the node into separate windows is not acceptable, since it is equivalent of replacing a node by a composite, which is contrary to what is postulated in pattern 2.1.

Solution: Present only a sub-set of the attributes, the mot important ones, and let the user control which further information is presented in the screen, by providing him with active interface objects (e.g. buttons). The activation of those buttons does not produce navigation; they just cause different attributes of the same node to be shown. This just follows the "What you see is what you need" principle. There are some considerations to be taken into account in this solution: for example we may use the same screen area to show different attributes (as exemplified below with Figure 6 and 7); we may even select some attributes and allow them to appear together in the screen. When dealing with other kind of media attributes we must analyze the situation carefully: for example an audio recording does not use the screen; however it may also distract the user's attention so it is wise to give the user the chance to activate/deactivate it being played. Other attributes will need to be synchronized such as an animation and a recording explaining it and so they must be perceived together.

Known uses: Information on Demand is used in Microsoft's Art Gallery, to provide further reading about a Painting; in the "Passage to Vietnam" CD ROM, it is applied to let the user read further information about a photograph. It is also used widely in "Le Louvre"

Related Patterns: Node as a single unit (2.1), and Information-Interaction coupling (3.3).

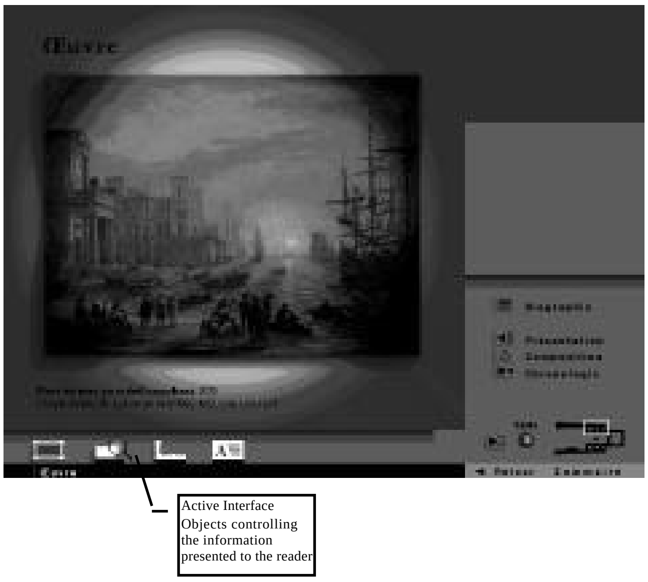

Œuvre
Active Interface Objects controlling the information presented to the reader

Figure 6: Interface showing the painting as the main interface object

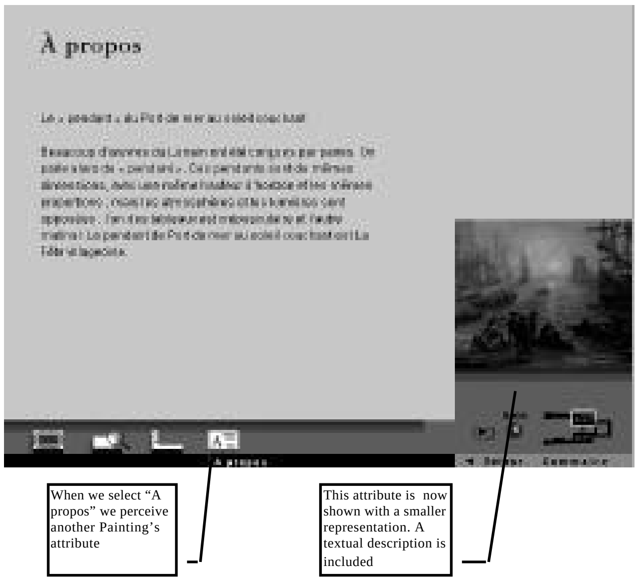

À propos
When we select "A propos" we perceive another Painting's attribute
This attribute is now shown with a smaller representation. A textual description is included

Figure 7: Interface on Demand. A textual attribute is perceived

3.2. Information-Interaction decoupling

Problem: How do you differentiate contents and various types of controls in the interface ?

Motivation: A node in a complex application displays different contents, and is related to many other nodes – thus providing many anchors. Moreover, if the node supplies in its interface means of control activation other than navigation (see Information on Demand), the user may experience cognitive overhead. It is well known that when too many anchors are provided in a text, the reader is distracted and cannot take profit of all of them. In case of graphics, parts of them usually provide some control activation, that may be hard to foresee (http://www.autoweb.com/ is an example of the occurrence of both problems).

Forces:

- A node's interface is usually composed of widgets displaying data and widgets providing control activation (at least those of navigation);
- Information (data) conveyed mixed with control information (such as menus or control widgets) is confusing
- Hypertext many times provides anchors to activate links merged with the data displayed, such as anchors in HTML documents. However, when either the anchor or the data to be shown is computed dynamically, anchors and data must be shown separately;
- It is oftentimes hard to see what has changed and what has not changed in a window refresh after some control activation; this can become clearer when the "fixed part" (unchanging) is separated from the "dynamic part".
- Control objects should not interfere with the "substantive" information being displayed.

Solution: Separate the input communication channels from the output channels, by grouping both sets separately. Allow the "input interaction group" to remain fixed while "the output group" reacts dynamically to the control activation. Within the output group, it is also convenient to differentiate the "substantive information" (i.e., content) from the "status information".

This solution not only improves the perception of a node's interface, but also the efficiency of the implementation.

Known uses: http://www.sigs.com/publications/subscriptions.html separate controls to the right and bottom. In http://www.amtrak.com controls are provided to the left in the home page, and at the bottom in the others.

Related Patterns: Information-Interaction coupling (3.3) is the dual pattern, that for the same problem, but under different forces, provides a different solution. Behavioral Grouping refines the organization of control objects.

3.3 Information-Interaction coupling

Problem: How do we make clear what is the object affected by a control in a node's interface?

Motivation : When control of navigation or other behavior is dependent on a node's content, separating the control from the content may provoke reader's disorientation. For example, in http://www4.zdnet.com/anchordesk/story/story_980.html, the last link “talk back” should be near the “talk back” list above.

Forces :

· A node’s interface is usually composed of widgets displaying data and widgets providing control activation (at least those of navigation) to manipulate the data;

· Different aspects or data is usually displayed in the same graphical interface, control is provided for more than one aspect and control is dependent on the particular aspect;

· Hypertext usually provides anchors to activate links over the data displayed when the data is fixed ;

· When different control channels with the same name must be provided for different aspects (such as an ‘add’ or ‘animate’ function) and these controls are grouped in an area of the screen, the name of each control channel must be artificially changed in order to differentiate them; therefore the names of control become long and unnatural.

Solution : Provide control channels close to the data that each affects, either by menus or buttons. (as for example in http://www.autoweb.com/loancalc.htm). Nevertheless it is important to note that menus are many times overlooked and buttons are better recognized.

Known uses: http://www.autoweb.com/loancalc.htm (the different ‘compute’ buttons beside each possible calculation), or http://www.sun.com/index.java.html (with its ‘search’ and ‘expand’ buttons).

Related Patterns: Information-Interaction decoupling (3.2) is the dual pattern that, for the same problem, but under different forces, provides a different solution.

3.4 Behavioral Grouping

Problem: How to organize the different types of controls in the interface so the user can easily understand them?

Motivation: Another problem we usually face when building the interface of a hypermedia application is how to organize control objects (such as anchors, buttons, etc.) to produce a meaningful interface. In a typical hypermedia application there are different kinds of active interface objects: those that provide "general" navigation, such as the “back” button, or anchors for returning to indexes; objects that provide navigation inside a context; objects that control the interface (as in "Information by Demand"), etc.

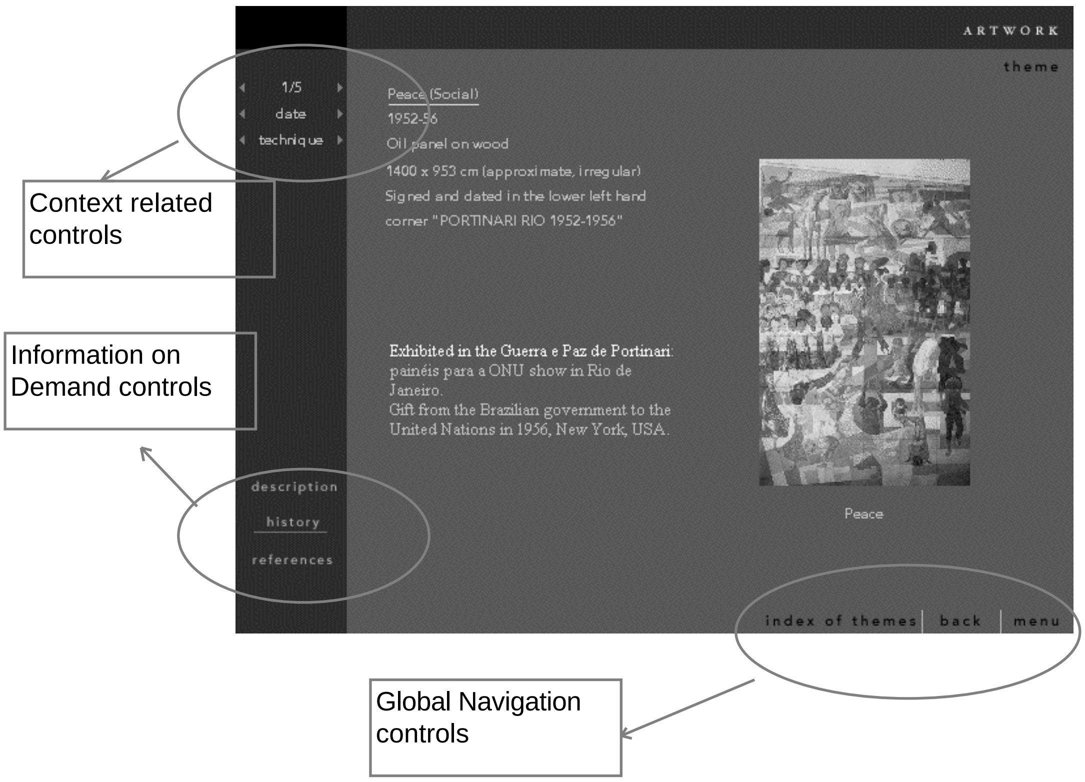
Forces:

- A node’s interface may have many different kinds of control objects, providing different functionality associated with possibly unrelated kinds of tasks.
- The variety of functions and deversity of tasks to be support does not allow solutions based on simple conventions such as “the back button is always at the right”.
- The control objects should not interfere with the “substantive” information being displayed.

Solution: Group control interface objects according to their functionality in global, contextual, structural and application objects, and make each group perceivable in a different screen area. Provide similar interface appearance inside each group to enhance comprehension. In Figure 8 we see an example of Behavioral Grouping in the context of Portinari. There are three screen areas for control objects: one in which "general" anchors are presented (at the bottom right), another for context anchors (at the left, on top) and another one for pure interface effects (at the left, below). Anchors for application links are presented as hotwords in the text as shown in Figure 8.

Known uses: This solution has been used in many commercial applications such as Art Gallery and Le Louvre. In Art Gallery for example, this pattern is used together with both Information by Demand and Information-Interaction Decoupling.

Related Patterns : Information-Interaction Decoupling (3.2) addresses the separation of data and control.



Peace (Social)
1952-56
Oil panel on wood
1400 x 953 cm (approximate, irregular)
Signed and dated in the lower left hand corner "PORTINARI RIO 1952-1956"

Exhibited in the Guerra e Paz de Portinari: painéis para a ONU show in Rio de Janeiro.
Gift from the Brazilian government to the United Nations in 1956, New York, USA.

Figure 8: Example of Behavioral Grouping in the Portinari Project Aplication

3.5. Behavior anticipation

Problem: How do you indicate the effect or consequence of activating an interface object?

Motivation: Many times we need to build an interface in which we must provide many different kinds of active objects; even though they can be grouped (according to “Behavioral Grouping” pattern), the user may find it difficult to understand what is the effect of activating one of them. It is usual to find hypermedia readers wondering about having selected or not a hotword, a button or a media controller, etc. In applications supporting both navigation and different kinds of media operations such as zooming, animating, etc., this problem is critical.

Forces:

- Many different kinds of active objects must be provided to the user.
- The reader may be confused about what to select.
- Even if we provide good icons they may be not enough to give the user a feeling of what will happen when he select that option.
- We must not distract the user's attention that must be focused on the application's content.

Solution: Provide differentiated feed-back about the effect of selecting each interface object. Choose the kind of feed-back to provide in such a way that it is non-ambiguous and complete. There are many ways of providing feed-back: different icons for the cursor as it is usual in WWW browsers; highligting; blinking; color changes; etc...; another type of feedback is to use an area of the screen to put a small text-based explanation. If possible, combine auditive and visual feedback.

If we are using the Behavioral Grouping interface pattern we can select different kinds of feed-back according the kind of behavior provided; when interface controls refer to a particular media such as an animation we could use a small status field for that family.

For navigation controls there are other ways of providing feed-back. In most WWW browsers, for example, we can only see the URL of the destination page in the status bar; we can enrich this information with a short text-based explanation of the target's contents , as is done in some sites (similar to "tool tips" in the Windows interface).

Known Uses: Many CD-Rom based hypermedia applications use this design pattern. For example in Microsoft's Ancient Lands, navigation is performed in a two steps basis. When you click a navigation anchor, a short pop-up explanation of the target node is provided. We can then select to remain in the current node or click again to navigate. In many WWW sites, Java applets are provided to give an instant feed-back of one interface control's behavior.

Related Patterns:

Process Feed-back (3.6) can be easily provided when we used with Behavioral Grouping to put together all active interface objects providing similar functionality. Behavior anticipation is a kind of Process Feed-back.

3.6 Process Feed-Back

Problem: How do we keep the user informed about the status of the interaction in such a way that he knows what to expect.

Motivation: When the user interacts with a hypermedia application, it may happen that many options result in non-atomic operations, such as contacting another machine (in the case of WWW browsers) or getting information from a database. In such cases the user may feel that he did not choose the correct option or that he made a mistake or even that the system is not working fine. The situation may get worse if the user, as he is loosing patience, begins selecting the same or other option, unknowingly queuing these selections and causing an unpredictable behavior when they are dequeued.

Forces:

- Some navigational or interface behavior may be non-atomic.
- Non-atomic operations may take variable amout of time to execute.
- A non-atomic behavior may fail after it began.
- The user may become impatient when he does not know what is happening.

Solution: Provide a constant perceivable feed-back about the status of the operation that is being performed, indicating progress in the case of non-atomic operations. Analyze which operations are atomic and do not need to be tracked. For operations that are non-atomic give information about the beginning, progression and ending of the operation. The kind of feed-back depends both on the user's profile and on the kind of interaction performed.

For example, whereas for many hypermedia applications (like Microsoft's Art Gallery or Ancient Lands) a single hour glass cursor may be enough, in WWW applications the kind of feed-back is far more elaborated: "looking up host", "host contacted", "transferring", "done" are displayed in the status bar; the icon on the top right of both Netscape Navigator and MS Internet Explorer play animations while processing is being done.

While in Web browsers Process feed-back is directly provided by the browser (though we can improve it with content related information such as "query being processed"), in most hypermedia development environments, the author must either change the cursor or define a status area to provide the feed-back.

Related Patterns: Both 'Process feed-back' and 'Behavior Anticipation' (3.5) address a similar problem: giving prompt feed-back to the user about the actions he is performing.

Known Uses: In Web Browsers like Netscape Navigator or MS Explorer, process feed-back is provided to show the status of the http connection when the user navigates to another web page. In most CD-Rom applications process feed-back is usually limited to changing the cursor icon.

Acknowledgement - The authors wish to thank the shepherd, Regine Meunier, for the excellent suggestions that improved the clarity of the paper.

References

[Bieber95] M. Bieber and C. Kacmar. "Designing Hypertext Support for Computational Applications". *Communications of the ACM 38* (8), August, 1995.

[Davis94] H. Davis, S. Knight y W. Hall. “Light Hypermedia Link Services: A Study of Third-Party Application Integration”. *Proceedings of the ACM European Conference on Hypermedia Technology.* Edinburgh, Scotland, 1994.

[Gamma95] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison Wesley. 1994.

[Garrido96] A. Garrido and G. Rossi. “A Framework for Extending Object-Oriented Applications with Hypermedia Functionality”. *New Review of Hypermedia and Multimedia Journal*, Taylor Graham Publishing, Vol. 2, 1996, pp. 25-41.

[Garzotto91] F. Garzotto, P. Paolini and D. Schwabe. “HDM - A Model for the Design of Hypertext Applications”. *Proceedings of Hypertext’91.* December 1991, pp. 313-328.

[Grønbaek94a] K. Grønbaek and R. Trigg. “Design Issues for a Dexter-Based Hypermedia System”. *Communications of the ACM 37 (2),* February 1994.

[Grønbaek94b] K. Grønbaek. “Composites in a Dexter-Based Hypermedia Framework”. *Proceedings of the European Conference on Hypermedia Technology,* ECHT’94, Edinburgh, Scotland, -, September 1994, pp. 59-69.

[Halasz94] Halasz e M. Schwartz: "The Dexter Hypertext Reference Model". Comm. of the ACM, February 1994, pp. 30-39.

[Isakowitz95] T. Isakowitz, E. Stohr and P. Balasubramanian. “RMM: A Methodology for Structured Hypermedia Design”. *Communications of the ACM 38* (8), 1995, pp. 34-44.

[Johnson97] R. Johnson and B. Woolf. “The Type-Object Pattern”. *To appear in Pattern Languages of Program Design. Vol. 3.* Addison-Wesley, 1997.

[Lanzelotte93] Lanzelotte, R.S.G.; M.P Marques, M.C.G. Penna, J.C. Portinari, I.D. Ruiz and D. Schwabe: "The Portinari Project: Science and Art team up together to help cultural projects". Proceedings of the 2nd International Conference on Hypermedia and Interactivity in Museums (ICHIM'93), Cambridge, UK, September 1993.

[Lange94] D. Lange. “An Object-Oriented Design Method for Hypermedia Information Systems”. Proceedings of the Twenty-seventh Annual Hawaii International Conference on System Sciences, Hawaii, January 1994.

[Nielsen97] Jakob Nielsen. “Be Succinct! (Writing for the Web)”. Alertbox for March 15, 1997. (http://www.useit.com/alertbox/9703b.html).

[Rossi96a] G. Rossi, A. Garrido and S. Carvalho. “Design Pattern for Object-Oriented Hypermedia Applications. *Pattern Languages of Program Design, Vol. 2,* chapter 11, pp. 177-191. Vlissides, Coplien y Kerth editors, Addison-Wesley, 1996.

[Rossi96b] G. Rossi, D. Schwabe and A. Garrido. “Towards a Pattern Language for Hypermedia Applications”. The 3rd. Pattern Languages of Programming Conference (Washington University technical report #WUCS-97-07), February, 1997.

[Rossi97] G. Rossi, D. Schwabe and A. Garrido. “Design Reuse in Hypermedia Application Development”. Proceedings of The Eight ACM Conference on Hypertext, Hypertext’97. Southampton, United Kingdom, April 1997.

[Schwabe96] Schwabe, D., Rossi, G. y Barbosa, S. “Systematic Hypermedia Design with OOHDM”. *Proceedings of the ACM International Conference on Hypertext, Hypertext'96,* (Washington, March 1996).

[Thistlewaite97] P. Thistlewaite, Automatic Construction and Management of Large Open Webs, to appear in M. Agosti and J. Allan (eds), Special Issue on Methods and Tools for the Automatic Construction of Hypermedia, IP&M.