

Patterns for Designing Navigable Information Spaces

Gustavo Rossi (*), Daniel Schwabe(**), Fernando Lyardet (*)

(*) LIFIA, Depto de Informática, Fac. de Cs. Exactas. UNLP

E-mail: {gustavo, fer}@sol.info.unlp.edu.ar

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

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

() also at UNLM and COINCET.

Abstract

This paper presents several design patterns for the hypermedia domain: Navigational Context, Active Reference, Landmark, News and Shopping Basket. They are part of a pattern language for hypermedia applications and address the design of healthy navigational structures. They can be applied in stand-alone applications or in dynamic Web sites or Information Systems.

1. Introduction. Designing High Quality Hypermedia Applications.

Hypermedia applications provide the user with navigational access to an information base. These applications are usually built by specifying a set of nodes related through links. Nodes represent unstructured multimedia information such as text, images, video and animations and provide navigation anchors that are usually perceived in the interface as buttons, hot-words or other reactive interface objects; by activating anchors we can follow links to other nodes. A successful hypermedia application allows an end user to perform a task by exploring the navigational space, finding suitable paths to the desired information, without suffering disorientation or cognitive overhead.

Examples of hypermedia applications can be found in many interactive CD-ROM based encyclopedia and information systems. Most WWW sites have an underlying hypermedia structure; as these applications have evolved, providing access to dynamic and growing navigable spaces, hypermedia design has clearly become a critical problem.

Even when non-OO hypermedia platforms are used for creating hypermedia applications (e.g. Toolbook, HTML-based authoring tools, etc.), it should be possible to apply well-known software engineering practices during hypermedia design and implementation. The World Wide Web is a good source to find both well and poorly designed hypermedia documents.

In the last three years we have been developing hypermedia applications using the Object-Oriented Hypermedia Design Methodology (OOHDM) [Schwabe 95, Schwabe 96, Schwabe98], a method which aims to provide a set of design models and guidelines to build high quality hypermedia applications. We found many interesting patterns of navigation and interface structures both in the systems we developed and in other successful commercial applications.

Our purpose in this paper is to show how we can apply the ideas underlying patterns and pattern languages to the design of well-structured hypermedia applications. The paper focuses on some simple and powerful patterns: *Navigational Context*, *Active Reference*, *Landmark*, *News and Shopping Basket*. They are part of a larger Pattern Language that also includes architectural patterns and patterns for user interface design; these patterns can be found in [Garrido97].

2. Context. The problems of hypermedia design.

One of the keys distinguishing features of hypermedia applications is the notion of navigation. While designing the navigational structure of the application, we have to take into account the types of intended users, and the set of tasks they are supposed to perform using the application. We describe the navigational structure of a hypermedia application by defining navigational classes that constitute a (customized) view over the application domain. This view is defined using a set of base classes such as Node, Link and Index.

Nodes are described by two types of attributes: content attributes and anchors. Content attributes store the information to be presented to the user, while anchors are the origin of links. Anchors are usually implemented at the interface level as reactive objects that, when stimulated, trigger navigation. Nodes are connected by links, and these are described by their origin and target nodes. Indexes meanwhile provide shortcuts to the desired information.

Navigational design poses many problems for the designer, such as preventing the user from being disoriented while navigating, giving him a sense of location at all times. In summary, a hypermedia application will be successful if the underlying information architecture allows the user to find what he wants easily, without feeling overwhelmed or having to browse through dozens of pages before completing his task. Patterns in this paper address the construction of sound navigation architectures.

Good designers tend to apply carefully some navigation design principles by making each node reflect an object of interest in the domain, and by linking objects that have strong semantic relationships. Expert designers go beyond that, by creating more elaborated navigation architectures.

To understand the difference between the naive use of the nodes and links data model and the way in which outstanding hypermedia applications are designed, we can make an analogy with some GOF [Gamma95] design patterns. A naive object-oriented designer will follow closely main O-O definitions by locating the state, structure and behavior of one domain object into the same application object. He would never separate state or structure (as in the State or Bridge design patterns), nor would he allocate behaviors in different classes (as in the Strategy or Command patterns).

Patterns in this paper are in some sense analogous to those in the GOF catalog; they show how to build navigable information spaces by pushing the simple hypermedia paradigm one step further. Each pattern in this paper is described using a combination of the GOF and Alexandrian styles. We do not address implementation issues because they depend strongly on technologies that are changing fast (like existing versions of HTML, XML or existing hypermedia authoring environments). In fact, the implementation strategies for these design patterns generate themselves new patterns (or implementation idioms), that are not addressed in this work.

3. Design Patterns for Hypermedia

3.1 Navigational Context Patterns

These two patterns address the problem of dealing with collections of nodes; they are usually applied together though we have found individual uses of each one of them.

3.1.1 Set-based Navigation

Intent:

Provide the user with closed navigational subspaces that can be easily navigated.

The Problem:

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 Paintings of a Painter, Paintings on a certain subject, etc., and it is desirable to let the user move easily from node to node. Since the links among set members do not usually reflect semantic relationships they are seldom implemented in hypermedia applications. The usual strategy followed by designers consists in providing an index to set members (see for example the results of a Netscape Search); users must then go back to the index to navigate to the next member of the set. Therefore, the usual simplicity of the hypertext navigation metaphor is lost.

Solution:

Consider set-based navigation a “first-class” navigation strategy. Group nodes in meaningful sets and provide intra-set navigation facilities, such as indexes and links for letting the user navigate to the “next” and “previous” elements of the current one. When meaningful, provide access to the first element of the set. We call these navigable sets “navigational contexts”. In Figure 1, we show a schematic view of a navigational context.

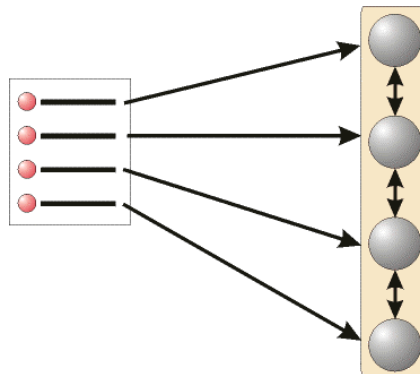


Figure 1: Set-based navigation

Navigation inside contexts complements conventional semantic links, as, for example, those connecting a Painting with its Painter or Painters with their biographies. In other words, the reader can browse through the set or leave it to explore other nodes (or eventually other sets). There are many different ways of defining contexts:

Class based – Objects in this kind of context belong to the same class C and are selected by giving a property P that must be satisfied by all its elements; for example “all Paintings on nature”

Link based – Objects in this kind of context are of the same class and are selected when they belong to a 1-to-n link. For example, “all Paintings by Picasso”. Note that a particular case of this type is the context formed by all elements that are part of a composite object.

Enumerated - In this kind of context, elements are explicitly enumerated, and may belong to different classes. A typical example is a Guided Tour, such as the one in a Museum showing different Artworks.

Dynamic - In this kind of context elements are inserted during navigation. An example of this type of context is the “history” maintained by many browsers.

Known uses:

Set-based Navigation has been used in many successful hypermedia applications. For example in Microsoft's Art Gallery, paintings of the London National Gallery are presented in different categories: by place, period and subject. Paintings in these categories can be browsed sequentially as shown in Figure 2.

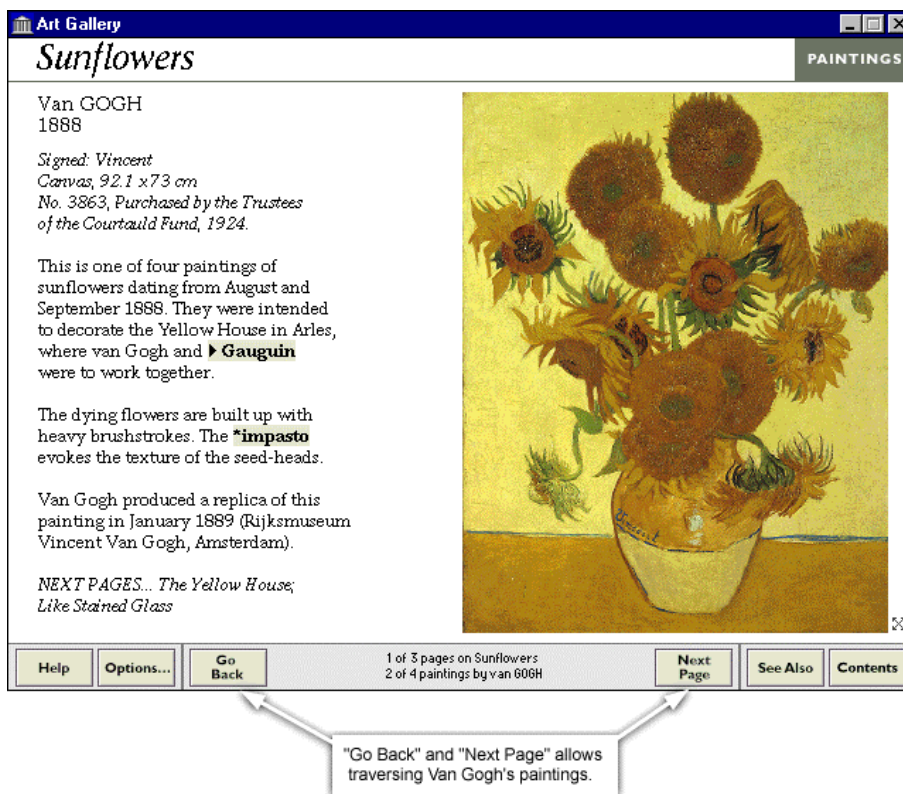


Figure 2: Navigating sequentially through paintings in Art Gallery

3.1.2 Nodes in Context

Intent:

Allow the same node to appear in different navigational contexts modifying its appearance and connections to other nodes according to the current context.

The Problem:

Suppose we are using context-based navigation in a gallery with paintings. When we reach “Van Gogh” we choose to navigate to his paintings, and then arrive at “Sun Flowers”. However, we can also reach “Sun Flowers” while exploring Paintings about nature. It is clear that we will explore the same object under two different perspectives. For example while accessing it as a work by Van Gogh we would like to read some comments about its relationships with other paintings by Van Gogh; we would also like to have easy access to other paintings he painted. Meanwhile, as a painting on nature, it would be fine to read (or see) something about

that subject and be able to access other paintings on the same theme (perhaps not Van Gogh's). This means that we will need not only to present the information in a different way in both cases, but also to provide different links or indexes.

Notice that we need to access *the same* hypermedia nodes under different contexts; those nodes should be connected with (contextual) links, e.g. it is desirable that one can move to the "next" painting in the same collections. However, though the context in which a node is being accessed is not a concern of the node, it is impractical and it may shield inconsistencies to have different objects to represent the same component but in each different context.

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. In this way Nodes in Context allows that the same object may provide different information, including links, according to the context in which it is being accessed.

In Art Gallery for example, the anchors shown in Figure 2 allow navigating through the current context; e.g. if we accessed Sun Flowers as a painting on nature the "next" button allows us to see another painting on nature. Meanwhile if we reach Sun Flowers as a Van Gogh's painting, the meaning of "next" changes to allow us to navigate to another of his paintings. Contextual information is presented to the reader in the bottom center pane.

Navigational Contexts patterns show a way of organizing the application's navigational structure. Navigational Contexts are composed of a set of Nodes (like Paintings) 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 (using Nodes in Context). An index node may provide links to all nodes in the context or a link to the first one.

Getting back to the "Van Gogh" example, we may think about 'Painting' and 'Author' as types of application nodes. We will have two possible navigational contexts: 'Theme-related context' and 'Author-related context'. Accordingly, we will define Theme-related-context decorators and Author-related-context decorators defining the pertaining information that a painting will show in each context, and anchors for the 'previous' and 'next' painting in the same context. A diagram of the interacting elements is shown in Figure 3, where the context node has an Index to the nodes and each decorator provides an anchor to the 'next' node in the context.

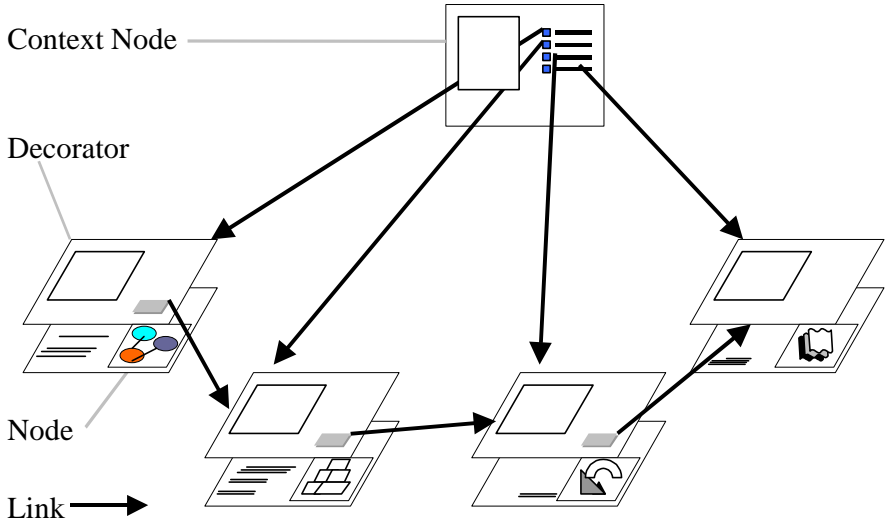


Figure 3: Diagram of Nodes in Context pattern.

Known uses:

As mentioned before, Microsoft's Art Gallery uses both Navigational Context patterns. Navigational Contexts can be also found in the Portinari Project [Lanzelotte93] and in Dorling Kindersley's "The Way Things Work".

3.2 Active Reference**Intent:**

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

The Problem:

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. For example in a digital Museum, we would want the reader to see the artworks and meanwhile he should know in which place of the museum he is. The usual naive 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 his current position is highlighted in the index. These navigational operations: moving backward to the index and forward to the target may disorient the reader.

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. Note that we are slightly changing the usual navigation style in hypermedia (Web) applications in which when we depart from a node (or index) it is closed and we can only return to it by backtracking from the current one.

An example applying this solution can be found in City.Net web site (<http://www.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.city.net.com/>, an index to all cities in a region can be seen 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 status of navigation and, in this way, we not only provide an orientation tool but also make them available while navigating the target nodes.

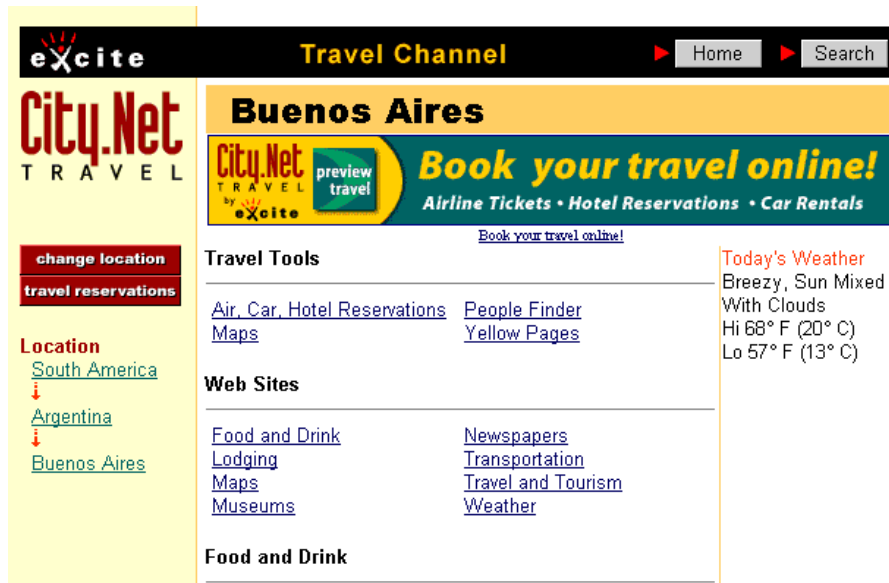


Figure 4: Example of Active Reference pattern in http://city.net/countries/argentina/buenos_aires/.

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.

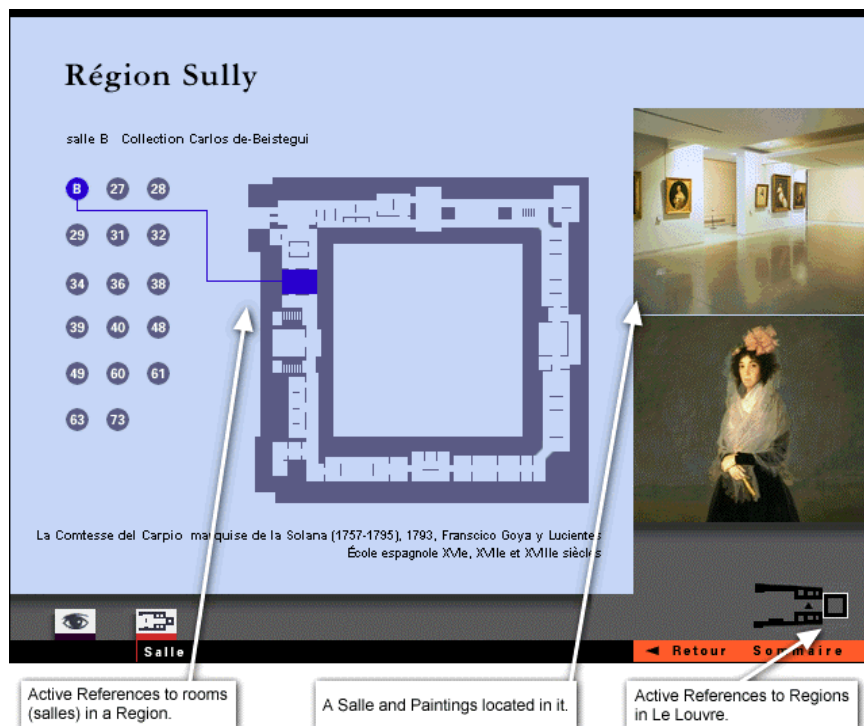


Figure 5: Active Reference in Le Louvre.

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 [Garrido97]) or about a painting, navigating to that painting.

This example shows an interesting rationale for applying “Active Reference”. In Le Louvre we want the end user to explore not only paintings but mainly the whole Museum; notice that in Figure 5, the spatial structure has been given greater relevance than the individual painting (though this one can be further seen in the whole page).

Known uses:

In Microsoft’s “The ultimate Frank Lloyd Wright” active reference 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 9th. 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>).

3.3 News

Intent:

Given a large and dynamic hypermedia application (for example a Web-site) provide the users with information about new items that have been added.

The Problem

Most large web-sites are tree-structured; this way of organizing the information provides a simple navigation metaphor though, these information spaces tend to be large, and are hardly ever completely navigated by a single user.

Suppose that we are building a site offering a great variety of hardware products (like printers, scanners, etc.). In this context, new information may be frequently added. Since most users navigate a web-site neither thoroughly nor regularly, this becomes an issue to be considered because the success of a new product may be closely related to the customer’s knowledge of its existence. Clearly, vital commercial information like this cannot be left to chance that it will be discovered. On the other hand, trying to solve this problem poses a design challenge for web designers who, must balance between a well-structured web-site where information is organized in items with sub-items, etc. and, a structure-less, star-shaped navigational structure where all information is reachable from the home page. The latter approach is clearly not desirable because the usability of the site is greatly reduced and it may become unmanageable as it grows. Therefore, how is the user provided with instant feedback of any recent changes or additions to the information available, while maintaining a well-structured web-site?

Solution:

Structure the home page in such way that a space is devoted to the newest additions, presenting descriptive “headlines” regarding them. Use those headlines as anchors to link them with their related pages. This approach allows the designer to preserve a good organization of the information while giving users feedback of the changes that take place within the web-site. News implement shortcuts to information that may be located in the leaves of a tree-structured site, without compromising the underlying structure. Notice that the navigational structure of the application is slightly affected by the addition of (temporary) links from one node to others.

Users can get the information advertised by news by browsing the site in the usual way. In fact, as new headlines appear, old ones may be moved away and replaced. In Figure 6 we show three web-sites where the headlines pattern is used extensively for corporate announcements, <http://www.ibm.com>, <http://www.inprise.com> and <http://www.sun.com>. An additional,

complementary solution is to add a "new" indicator (a splash symbol, changing color, etc...) next to new or recently changed items wherever they naturally occur in the application.

Known Uses:

News is used in hundreds of web-sites and applications such as <http://www.nga.gov> where it is used to announce new collections and the current tours available.

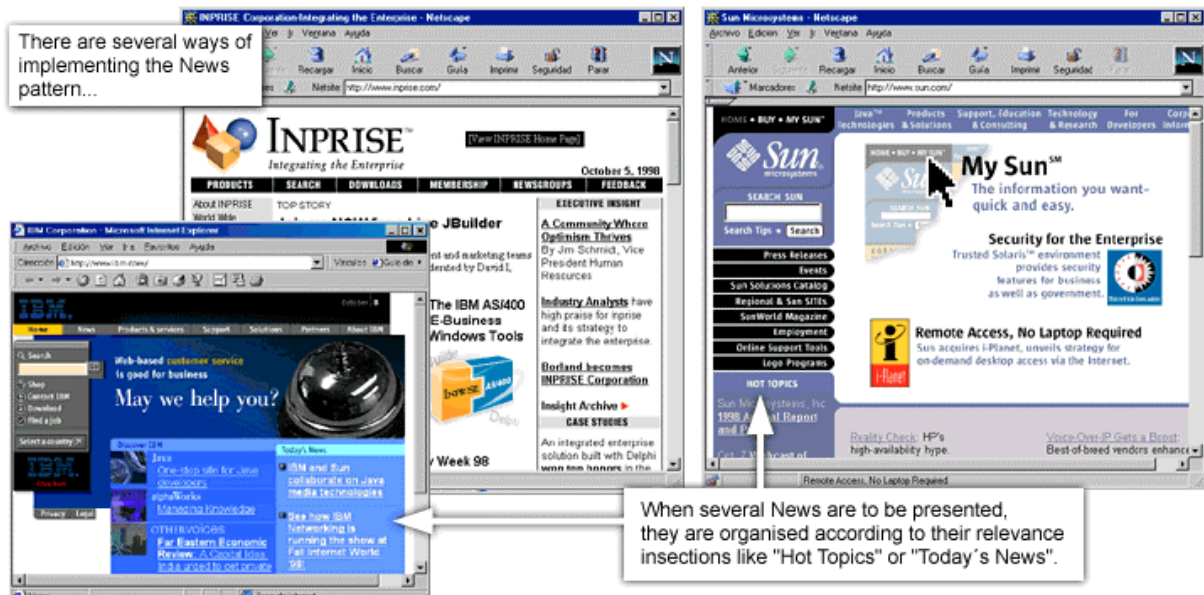


Figure 6: Three examples of websites using the News pattern: www.ibm.com, www.inprise.com and www.sun.com

3.4 Landmark

Intent:

Provide easy access to different though unrelated subsystems in a hypermedia application.

The Problem:

Suppose we are building a Web-Information System for an electronic shopping such as www.amazon.com. By entering the site, we can build many different products such as videos, books or CDs. We can explore the products and besides we provide links to recommendations, comments on the products, news, etc. When we describe the navigational schema (i.e. the network of nodes and links types), we try to follow closely those relationships existing in the underlying object model; for example we can navigate from an author to his books, from a CD to the list of songs it includes. We can go from a book to some comments previous readers did, read about related books, etc. However, we may want that at any moment the reader can jump to the music or book (sub) stores or to his shopping basket. If we build the navigational schema linking every navigational class (such as book, comment, news, songs, etc) to the Music Store, the Book Store or the Shopping Basket we will end with a spaghetti-like and difficult to understand schema; those links are clearly outside the model.

Solution:

Define a set of landmarks and make them accessible from every node in the network. Make the interface of links to a landmark look uniform. In this way users will have a consistent visual

cue about the landmark. We may have different levels of landmarking according to the hypertext area we are visiting. In Figure 7 we can see an example of Landmarks in the Amazon bookstore.

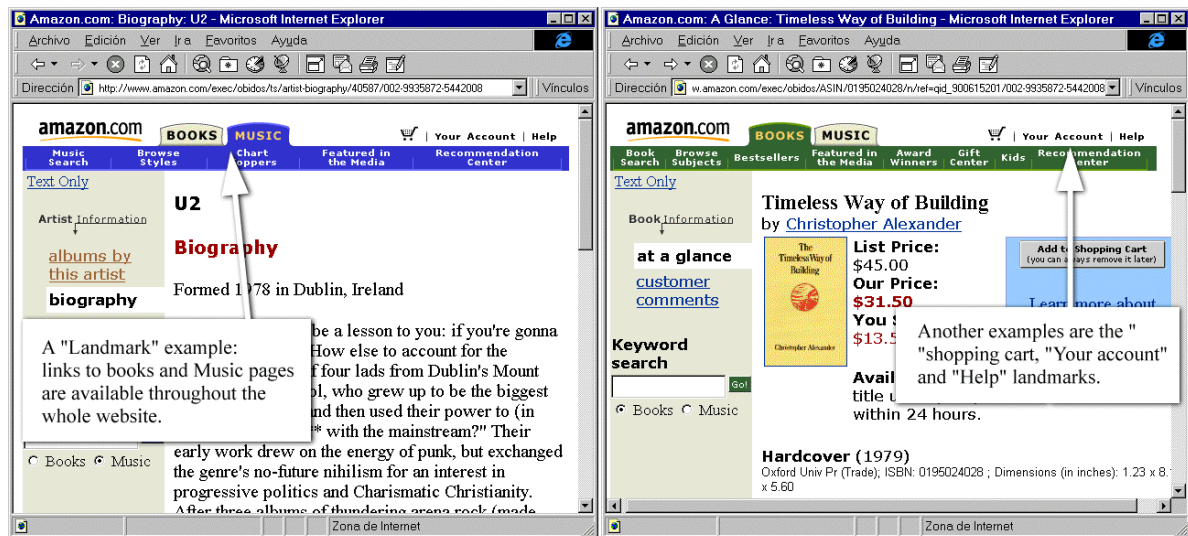


Figure 7: Landmarks in Amazon.com

Known uses:

Landmarks can be found in different Web applications such as www.inprise.com. In Macaulay's "The way things work" the reader can permanently jump to Inventions, Principles of Science and World of Machines.

3.5 Shopping Basket

Intent:

Keep track of user selections during navigation, making these selections persistent to process them when the user decides it.

The Problem:

Electronic-commerce is now a reality in the Web. However, users often want to navigate through the e-market to decide what they will buy and in what moment. In www.amazon.com for example a user can browse through hundreds of books or CDs and choose a sub-set of them to be bought. The naive solution would be either to ask the user to buy the product in the moment he finds it or force him to bookmark all the products he likes and buy them in another navigation session. It is clear that these approaches are not suited to cases in which we want to buy dozens of different products as it is quite uncomfortable for the user (and even for the shop) to require one transaction for each product. This alternative has another drawback, as we need to navigate to the "check-out" page many times thus wasting connection time.

Meanwhile, we should be aware that these e-commerce restrictions should not interfere with the overall Web-site navigational structure.

Solution:

Provide the users with a metaphor similar to bookmarking, by allowing him to select the products he wants to buy as they are traversed. Provide a "persistent" store for those items (a

shopping basket) that can be accessed as another navigation object and associate processing operations to the basket such as eliminate an item, change quantities, check-out, etc.

This solution is easy to implement by adding an interface object (usually a button) to every available product in a shopping site. When the user select this product, it is added to his shopping list. Later the user can navigate to the shopping basket where he will find either a detailed description of each product or a summarized one plus an anchor to the product page.

In many Web-sites the shopping basket facility can be enriched with validation operations. For example if a customer is planning his vacation with an Internet travel agent service, the travel agent may be capable of doing some checking on the arrival and departure times on hotel reservations and viability of flight connections.

The result is a very natural approach since it resembles the way people buy at the supermarket, adding products to their shopping cart while they walk.

Known Uses:

There are many examples of this pattern available on the web. One of them is the Amazon Bookstore (<http://www.amazon.com>) as shown in Figure 8.

Another example is Business travel (<http://www.biztravel.com>) where the user adds different destinations to a business tour, together with car and hotel reservations. The system checks, among other things, the dates of departure and arrival from the different destinations (you can't make a hotel reservation if you are leaving before arriving).

A completely different example is PublishersDepot (<http://www.publishersdepot.com/>), an image bank where the users can search for different kinds of images (textures, backgrounds, etc) and put them on a list. Furthermore, the users are able to create different lists of selected pictures that are persistent; the web-site keeps those lists of selections made by a user and every time the user logs in, additions or deletions can be made to these lists.

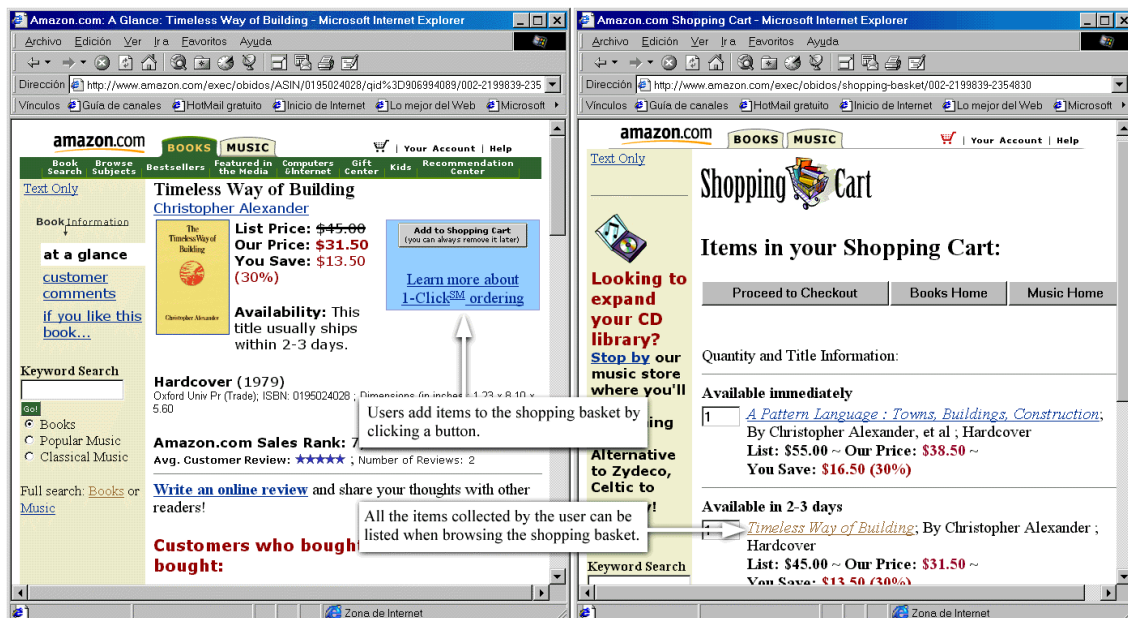


Figure 8: An example of Shopping Basket in www.amazon.com. Customers select a book by clicking on a button to add the current item to the shopping basket. All selected items are shown in the basket page.

4-Making Navigation Patterns work synergistically

Defining the navigational structure of a hypermedia application requires not only defining meaningful nodes and connecting them judiciously by reflecting the semantic relationships in the application domain. We also need to help the user in finding the desired information and make him feel comfortable while he navigates. Providing orientation tools like maps or shortcuts like indexes is important, however more subtle organization problems must usually be solved. In this paper, we described some patterns that go beyond the simple nodes and link hypertext model and that show that usual navigation strategies can be improved by following the solutions in those patterns. Navigational Context patterns such as “Set-based Navigation” allow providing navigation paths through members of a set thus enriching the navigation architecture in a structured way. When the same node may belong to different sets, the “Nodes in Context” pattern shows a way to decouple the node from the context in which it is accessed by using the Decorator design pattern. The “Active Reference” pattern meanwhile shows a way to change the usual navigational semantics of hypermedia applications by allowing indexes or other access structures to actively co-exist with target nodes. When the application is itself divided in several sub-systems, the Landmark pattern provides a cue to make those sub-systems readily accessible from any part of the application. “News” and “Shopping Basket” are patterns for dynamic Web information systems. While “News” shows how to make updates or additions easy to find, “Shopping Basket” lets keep track of users selections to process them altogether when the user decides to do that. This pattern system is complemented with a set of interface patterns that shows how to cope with the problem of designing multimedia interfaces for navigational objects in hypermedia applications.

5-References

- [Gamma 95] E. Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns: Elements of reusable object-oriented software", Addison Wesley, 1995.
- [Rossi 96] G. Rossi, A. Garrido and S. Carvalho: "Design Patterns for Object-Oriented Hypermedia Applications". Pattern Languages of Programs 2, Vlissides, Coplien and Kerth *eds.*, Addison Wesley, 1996.
- [Garrido97] A. Garrido, G. Rossi and D. Schwabe: “Pattern Systems for hypermedia”. Proceedings of PloP’97, Allerton, USA, 1997.
- [Lanzelotte93] R.S.G. Lanzelotte, M.P Marques, M.C.G. Penna, J.C. Portinari, I.D. Ruiz e D. Schwabe: "The Portinari Project: Science and Art team up together t help cultural projects". Proceedings of the 2nd International Conference on Hypermedia and Interactivity in Museums (ICHIM'93), Cambridge, UK, September 1993.
- [Lyardet98] F. Lyardet, Gustavo Rossi and D. Schwabe: “Patterns for Dynamic Websites”. Proceedings of PloP’98, Allerton, USA, 1997.
- [Schwabe95] D. Schwabe and G. Rossi: The Object-Oriented Hypermedia Design Model. Communications of the ACM, August 1995, pp 45-46.
- [Schwabe96] D. Schwabe, G. Rossi and S. Barbosa: "Systematic Hypermedia Design with OOHDm". Proceedings of the ACM International Conference on Hypertext (Hypertext'96), Washington, March 1996.

[Schwabe98] D. Schwabe and G. Rossi: "An Object Oriented Approach to Web-Based Application Design". TAPOS (Theory and Practice of Object Systems), Wiley and Sons, 1998, forthcoming.