User Interface Patterns for Hypermedia Applications

Gustavo Rossi
LIFIA-Universidad Nacional de La Plata, UNLM and Conicet
Calle 50 y 115, La Plata, Buenos Aires, Argentina
(54 221) 4236585
gustavo@sol.info.unlp.edu.ar

Daniel Schwabe
Depto Informática, PUC-Rio
Rua Marques de Sao Vicente, 225,
Rio de Janeiro, Brasil
2nd line of address
(55 21) 512 2299
schwabe@inf.puc-rio.br

Fernando Lyardet
LIFIA-Universidad Nacional de La Plata
Calle 50 y 115, La Plata, Buenos Aires, Argentina
(54 221) 4 236585
fer@sol.info.unlp.edu.ar

ABSTRACT

Designing high quality visual interfaces for hypermedia applications is difficult; it involves organizing different kinds of interface objects (for example those triggering navigation), prevent the user from cognitive overhead, etc. Unfortunately, interface design methods do not capture design decisions or rationale, so it is hard to record and convey interface design expertise.

In this paper, we introduce interface patterns for hypermedia applications as a concept for reusing interface designs. The structure of this paper is as follows: first, we introduce the context in which these patterns were discovered and we give a rationale for their use. Then we present some simple but effective patterns using a standard template. We finally discuss some further issues on the use of interface patterns in hypermedia applications.

Keywords

Hypermedia Applications, Design Patterns, Interface Patterns.

1. INTRODUCTION AND BACKGROUND

During the last four years we have been developing hypermedia applications (in CD-ROM and in the Web) using the Object-Oriented Hypermedia Design Method (OOHDM). OOHDM comprises four different activities, conceptual modeling, navigation design, abstract interface design and implementation.

OOHDM explicitly separates navigation from user interface design; this means that design decisions related with the navigational topology of the application are (in a broader sense) independent respect to those regarding interface issues (See for example [12,13]). Separating navigational from user interface design allows us to define different interfaces for the same navigation structure and maximize modularity.

AVI 2000, Palermo, Italy.

OOHDM provides primitives for specifying the interface structure and behavior. Those primitives are based on Abstract Data Views (ADVs) [4]. ADVs allow the designer to describe the static aspect of the interface as an aggregation of objects; the relationships with the underlying application objects are described using Configuration Diagrams [3] and the dynamic aspects are specified using ADV-charts, a dialect of Statecharts [2]. Although we think that the ADV model is very attractive for formally describing the interface of a broad range of hypermedia (and multimedia applications), we found that many design decisions remain undocumented or hidden in code. In many cases, besides, simple interface behaviors require complex diagrams that tend to obscure the design. We believe that interface patterns are a good way to solve both problems.

Patterns have their roots in architecture [1] and they have been used in software design, in particular object-oriented software for some years now [5]. Patterns record design experience by describing recurrent problems and good and proven solutions. Patterns describe both problems and solutions in an abstract way so that they can be "instantiated" in many different situations.

Patterns complement design methods as they show solutions that go beyond the use of primitives of a method. In the case of user interface design, patterns are an attractive way of structuring guidelines in such a way that they can be applied systematically.

Recently we introduced navigational and interface patterns for hypermedia applications [9]. Navigational patterns are similar to Alexander's patterns. They describe the organization of a navigable space, the roads you can follow to reach different homes, the kind of orientation signs you will find, the short cuts, etc. They are also similar to object-oriented patterns as they show "advanced" solutions that go beyond the simple nodes-and-links metaphor. For example the *Set-based Navigation* pattern [10] explains when it is important to implement links among members of a set (e.g.: books of an author) allowing to traverse the set sequentially; meanwhile, *Nodes in Context* [10] focuses the problem that arises when the same object may belong to different sets.

Interface patterns, meanwhile, show how to organize the interface (both in structure and behavior of its parts) in order to make it more understandable and usable. Though it is clear that some interface design decisions will be related with the application's navigational topology, in this paper we try to explain interface patterns without focusing on navigation.

We have mined many patterns that show recurrent design problems and their solutions both while building the application's navigational architecture and when defining its user interface [7, 8, 9, 10]. In this paper, we present some patterns related to the design of user interfaces for hypermedia applications (in particular Web applications).

To put the patterns in context we must briefly explain which is the product of navigational design in hypermedia applications. During this activity, we obtain a set of nodes (that contain multimedia information and anchors) and links that connect them. The user interface design activity aims at defining how nodes are perceived; this involves defining the interface of their attributes and anchors. One of the forces that constantly appear in all interface patterns is that we may have many different kinds of information items in a node, which may also have different purposes. We must organize those items in a scarce perceivable space, make them understandable, avoid cognitive overhead, etc.

We next introduce some of our hypermedia interface patterns. Although these patterns are not intended to define a pattern language like the one in [1], they cover most design decisions related with architectural aspects of the interface.

We use a template that combines the original Alexandrian style with the template in [5]. For the sake of conciseness, we have simplified the template; we emphasize the problem with a motivation that includes the forces behind the pattern, and the solution.

We also include some known uses of each pattern to stress the idea that patterns must be proven, well-known solutions. The idea of our patterns is to formalize these solutions adding a brief rationale on the context in which they are applied.

The order in which the patterns are presented gives a simple guideline for their use in concrete applications. *Information on Demand* helps to decide which node attributes to show, *Information-Interaction Coupling and Decoupling*, show how to organize interface objects according to their role (input or output objects) and *Behavioral Grouping* organizes them by functionality. Finally, *Behavior Anticipation* and *Process Feedback* explain how to make the user aware of the intent of an interface object and its behavior. This order is obviously not mandatory, as these patterns do not comprise a language. We next present some of them

2. INFORMATION ON DEMAND

2.1 Intent

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?

2.2 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. Suppose for example an application on Paintings. We may want to show different attributes of a Painting such as painter's name, year, museum, technical description, etc., but this is difficult to achieve if we want to maximize the space we dedicate to the picture itself.

This problem appears when a node has an amount of information to be perceived by the reader that does not fit in one screen, or that may distract the user's attention (for example, an audio recording). Furthermore, scrolling may be often not acceptable because the reader doesn't get an 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.

It may be tempting to partition the node by using different pages for presenting the information, and defining links among these new nodes. This is also problematic because in our attempt to match design with an implementation issue, we may pollute the overall application's navigational structure. The user may get the impression of dealing with multiple entities, becoming disoriented, while in fact he is accessing another part of the same conceptual entity. We next summarize the forces behind the pattern:

- A node has an amount of information to be perceived by the reader that does not fit together in one screen, or may distract the user's attention (for example, an audio recording).
- Scrolling is often not acceptable because the reader doesn't get an overall view about what he will find in that node. Besides, he will have to scroll all the way down to see if there is something that interests him or not;

Partitioning the node into separate windows is not acceptable, since it is equivalent of replacing a node by a sub-network. This decision should not be driven by interface constraints.

2.3 Solution

Present only a sub-set of the attributes, the most important ones, and let the user control which further information is presented in the screen, by providing him active interface objects (e.g. buttons). The activation of those buttons will not trigger navigation; they just cause different attributes of the same node to be shown. This solution follows the "What you see is what you need" principle.

There are some considerations to be taken into account: for example we may use the same screen area to show different attributes, 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.

2.4 Known Uses

In Figure 1 we show an example of Information on Demand in the context of Microsoft's Frank Lloyd Wright's CD. In this case the textual explanation is superimposed on the building's image just by moving the cursor on the former attribute.

In Figure 2, we show an example of the same pattern in the WWW. The information presented to the user changes as the user moves through the list of products. Notice that the user does not need to navigate to a further page, neither he needs to scroll, as information is shown "on demand".

It is interesting to comment here that this pattern has not been very popular in the WWW as designers are not aware that they

can implement it easily, resulting in less connection time, better implementations and smarter designs. Just compare this solution with the "conventional" one based on navigation.

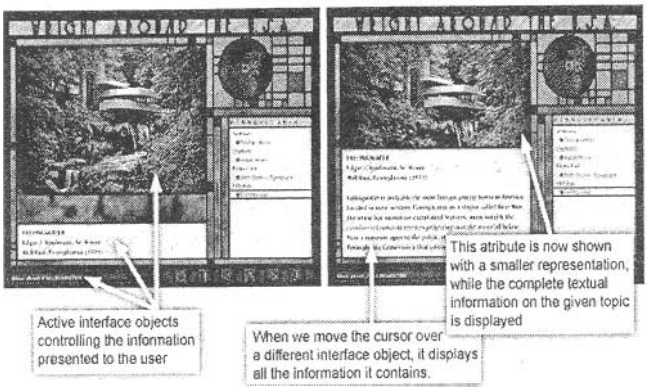

Figure 1: Information on Demand in Frank L. Right's CD

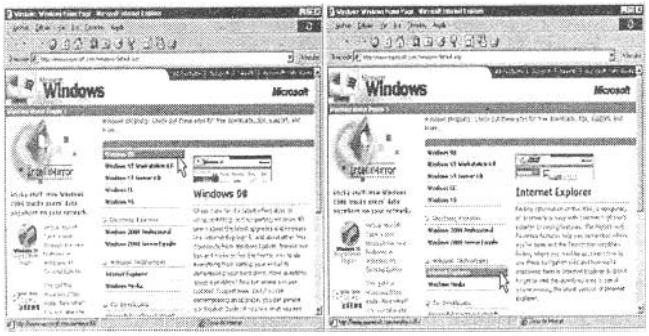

Figure 2: Information on Demand in www.microsoft.com/windows/. Whenever the user selects a different product, the related information is displayed.

Finally, in Figure 3.a and 3.b we present the elegant "Le Louvre" implementation. When the user selects the small text icon in the bottom of the screen, the focus changes. While the painting is shown in a smaller view on the right bottom, the center of the screen is used for the textual explanation.

In this final example, the user may choose to see either a textual explanation, a zoom on the painting and the scale (as compared with a human being).

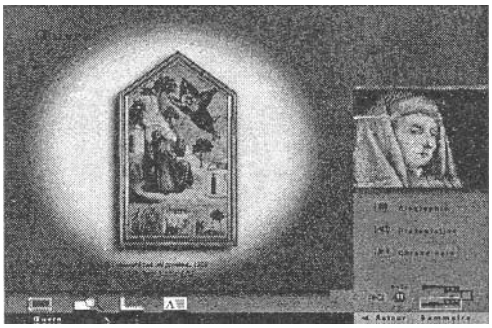

Figure 3.a: A painting in Le Louvre. The "default" view

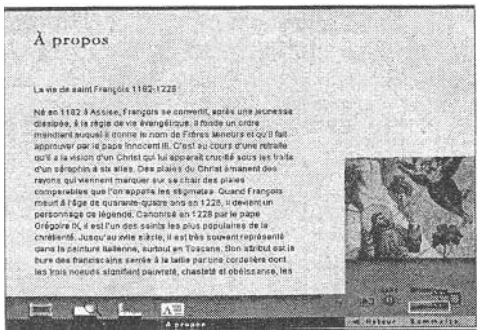

Figure 3.b: Information on Demand in Le Louvre. The icons on the bottom control the presentation of the painting.

3. INFORMATION-INTERACTION DECOUPLING

3.1 Intent

Help the user understand how to manage the interaction with the application. Differentiate the interaction controls from the information

3.2 Motivation

When a node displays different types of contents or it is linked with many other nodes, and if it supplies means of control activation other than navigation in its interface, the user may experience cognitive overhead. The examples before show clearly this problem (See Figure 2). It is also well known that when too many anchors are provided in a text, the reader is distracted and may not understand their meaning. Forces are summarized below:

- A node's interface is usually composed of interface objects displaying data (text or graphics) and objects providing control activation (at least those triggering navigation).
- When different items of data are merged with menus or widgets provided for user's control, the interaction becomes unclear.
- Hypermedia applications usually provide anchors to activate links. However, when the data to be shown is dynamically computed, anchors may have to be shown separately;

The information displayed usually changes with user interaction (activated by fixed buttons) and it may be hard to see what has changed after some control activation; it is again clearer when the "fixed part" is separated from the "dynamic part".

3.3 Solution

Separate the input communication channel from the output channel, by grouping both sets separately. Allow the "input interaction group" to remain fixed while "the output group" may react 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 usually improves the perception of a node's interface. However, as we show in the next pattern, there are situations in which we need a different solution.

3.4 Known Uses

In www.sigs.com/publications/subscriptions.html controls are located on the right and bottom. In www.amtrak.com, navigation controls are provided on the top and bottom; as a kind of

compromise, some of them are inserted in the text (See Figure 4). The same use of this pattern can be found in many Web Applications providing information about schedules, such as www.airfrance.fr. In this case for example, the schedules are presented in the center of the screen and navigation controls on the left.

Figure 4: Information Interaction decoupling in Amtrak.com. Most navigation controls are outside the information area.

4. INFORMATION-INTERACTION COUPLING

4.1 Intent

Help the user understand how to interact with the application. Make evident those controls that are related with information items

4.2 Motivation

When control of navigation or other functionality is highly dependent on particular nodes' content, separating the control from the content (as in *Information-Interaction* decoupling) may provoke reader's disorientation. Moreover, when dealing with dynamic media, such as audio or video, interface objects to play, pause or stop the media should not be decoupled from the corresponding media. Some of the forces behind this pattern are similar to the ones in *Information Interaction decoupling*. We next list them:

• A node's interface is usually composed of widgets displaying data and widgets providing control activation.

•Hypermedia applications usually provide anchors to activate links; many times these anchors depend on a particular content (a part of a map for example)

Many times, we need to provide interface objects to activate or trigger certain functionality (playing a media, initiating a query, etc). The situation may be even worse when many of these objects must appear in the same screen (for example different options in a query, different books to add to a shopping basket, etc)

4.3 Solution

Provide control interface objects close to the data that is related with the corresponding functionality. Try to use this solution with those objects providing controls for dynamic media or some specific functionality. This pattern shows a different solution with respect to the previous one (*Information-Interaction decoupling*) by considering the different forces that act on the problem. It is also interesting to note that these two patterns have a subtle intersection with *Behavioral Grouping*. This is a nice example of relationships among patterns

4.4 Known uses

In www.autoweb.com/loancalc.htm the different 'compute' buttons are located beside each possible calculation; the same organization can be found in http://www.sun.com/index.java.html (with its 'search' and 'expand' buttons). In the Amtrak.com example in Figure 4, we can see the application of this pattern together with the previous one; notice that the interface object to trigger the getSchedule query is located close to the form.

5. BEHAVIORAL GROUPING

5.1 Intent

Help the user recognize different types of controls in the interface so that he can easily understand them

5.2 Motivation

A 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 application there are different kinds of active interface objects: those that provide "general" navigation functionality, such as the "back" or "contents" buttons, anchors for returning to indexes, objects that provide navigation inside a context; objects that control the interface, etc. Even if we decide to decouple the information contents from the interaction controls, it may happen that we have much different kind of interaction activities and we should organize them. As said before, the forces in this pattern (listed below) partially overlap with the forces in the two previous patterns:

• 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 diversity of tasks to be supported does not allow solutions based on simple conventions such as "the back button is always at the right".

Control objects should not interfere with the "substantive" information being displayed.

5.3 Solution

Group control interface objects according to their functionality. For each group, define uniform interfaces to enhance comprehension. Typical groups in hypermedia applications may be: global navigation controls (back, contents, history, etc), anchors for related nodes ("See also" and other relationships with more "semantics"), interface controls (buttons implementing *Information on Demand* for example), other application functionality nor directly related with hypermedia, etc.

5.4 Known uses

In Figure 5, showing the hotmail.com site, there are three different groups of control objects: those associated with the current mailbox, those related with the email account (at the left above) and those that provide general navigation (at the left below). In Figure 3.a (Le Louvre application) there are three groups: on the left bottom those controlling what appears in the

interface, on the right bottom general controls and on the right anchors for related nodes (biography, painter, etc).

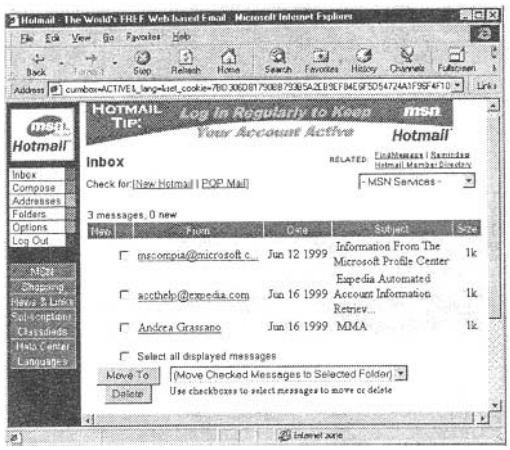

Figure 5: An example of "*Behavioral grouping*" from the Hotmail.com site

6. BEHAVIOR ANTICIPATION

6.1 Intent

Show the user the effect or consequence of activating an interface object

6.2 Motivation

Many times, when building an interface, it is necessary to combine different interface elements such as buttons, hot-words, media controls or even custom-designed controls. It is usual to find readers wondering what will happen after activating a control, and what is the exact consequence of the action he will perform. Notice that even when we group interface objects using *Behavioral Grouping*, we still need to help the user understand the meaning of each object. Forces are summarized below:

- Many different kinds of active objects may have to be provided to the user.
- The reader may be confused about which object 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 selects that option.

We must not distract the user's attention that must be focused on the application's content.

6.3 Solution

Provide feedback about the effect of activating each interface element. Choose the kind of feedback to be non-ambiguous and complete: different cursor shapes, highlighting, small text-based explanations called "tool tips". In addition, these elements can be combined with sound and animations.

If we are using the *Behavioral Grouping* interface pattern, we can select different kinds of feedback according the kind of behavior provided. For example, when the interface controls refer to a particular media such as animation, we could use a small status field for that family.

6.4 Known uses

In Figure 6, we show an example from the Microsoft Atlas Encarta97. Each time the user positions the cursor over an interface element, a tool tip pops up with an explanation about the effect of activating the control. Similar examples are available on the web. Some web-sites site uses standard GUI ToolTips and JavaScript combination to show information such as http://www.nervemag.com/ (the ToolTip appears at the bottom of the page). Another example using only JavaScript can be found in www.mercedes.com homepage (also in Figure 6).

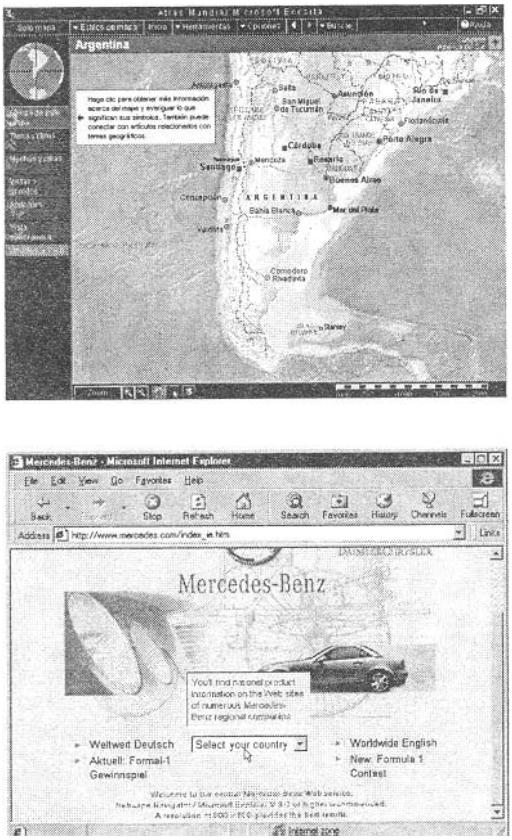

Figure 6: Examples of "*Behavior Anticipation*". Notice the information displayed as the mouse moves over the different components of the interface.

7. PROCESS FEED-BACK

7.1 Intent

Keep the user informed about the status of the interaction in such a way that he knows what to expect

7.2 Motivation

When the user interacts with a hypermedia application, it may happen that an interface action (clicking a button for example) results in a non-atomic operation. For example contacting another machine (in the case of WWW browsers), getting information from a database or loading animations are non-atomic operations. 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. The situation may get worse if the user, as he is loosing patience, selects the same or other option again. In this case, he will unknowingly queue these selections, causing an unpredictable behavior when they are dequeued. Following are the most important forces behind this pattern:

- Some navigational or interface behaviors may be non-atomic.
- The user may become impatient when he does not know what is happening.
- A non-atomic behavior may fail after it began.

7.3 Solution

Provide a continuous perceivable feedback about the status of the operation that is being performed. For non-atomic operations give information about the beginning, progression and ending of the operation. The type of feed-back depends both on the user's profile and on the kind of interaction performed. For example, while for many hypermedia applications (like Microsoft's Art Gallery or Ancient Lands) a single hourglass cursor may be enough, in WWW applications the type of feedback required may be far more elaborated. The icon on the top right of both Netscape Navigator and MS Internet Explorer play animations while processing is being done and information about the status of connection is also shown. While in Web browsers some *Process feedback* is directly provided by the browser, in most hypermedia development environments, the author can either change the cursor or define a status area to provide the feedback.

7.4 Known Uses

In Web Browsers like Netscape, process feedback is provided to show the status of the http connection when the user navigates to another web page. In most CD-ROM applications, the kind of process feedback is usually limited to changing the cursor icon. In www.expedia.com when we querying the flight or cars database (an operation that may take time), the user receives a short message saying that they are processing the query. In www.hotmail.com the process of logging out is non-atomic and the site provides a detailed description for each step.

8. DISCUSSION AND CONCLUDING REMARKS

In this paper, we have introduced hypermedia interface patterns as a powerful mechanism for recording, conveying and reusing design experience while building hypermedia applications. We have been mining these patterns during some years and have found them recurrently in many applications and domains. We have also used them in many applications. When designers are aware of successful solutions, they improve their designs. For example, using the *Interface on Demand* interface pattern not only improves comprehension (by presenting the most important items and letting the user control which information is presented). The global navigation architecture is also improved, as the designer does not need to define new nodes artificially.

Using design patterns in software design is a new and hot trend that has achieved an interesting degree of success in object-oriented applications. There are collective efforts for using patterns also in the field of human computer interaction, see for example [6]. Discovering, formalizing and using patterns during interface design is an appealing approach that, however, needs further formalization.

We are now working in two directions with respect to formalizing hypermedia interface patterns. First, we are improving our ADV notation incorporating some interface pattern as higher level primitives. For example, when we specify an interface that involves some instantiation of the Information on Demand pattern, the specification involves some (perhaps complex) state changes in the set of perceivable objects that obscures it. Even in a simple example (like the one in Figure 1) we may end with a cluttered diagram. It should be better if we could simply state that some information appears on demand and that the controlling object is some specific one. In this way, we may obtain a more concise specification without loosing formality. Unfortunately, other patterns (e.g. Behavioral Grouping) are not easy to formalize using this approach though we are looking for better ways to document them. One alternative would be, for example, grouping the diagrammatic elements in the way recommended by the pattern. We are also adapting our design approach to the set of primitives provided by the UML (Unified Modeling Language) [14] by extending the notation using stereotypes. In this way we would end with a standardize notation that incorporates interface patterns in a natural way.

Another critical aspect is how to use patterns as active guidelines during the hypermedia development process. It is clear that patterns are unconsciously used by expert designers by "matching" new problems with well-known situations that they faced in the past. (See [11]). We believe that the creation of a catalogue of interface patterns will act as a catalyst in the same way that patterns in [5] encouraged the object-oriented community to begin reflecting on their designs. In this sense, the user interface community has an important background that surely contains dozens of patterns that have not been mined yet. Even though this paper only deals with interface patterns for a sub-set of applications (hypermedia) and not necessarily "advanced" interfaces it may serve as a basis for further discussion in this area

9. REFERENCES

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

[2] L.M.F. Carneiro, M.H. Coffin, D.D. Cowan, and C.J.P Lucena: "ADVCharts: a Visual Formalism for Highly Interactive Systems", Software Engineering in Human-Computer Interaction, Cambridge University Press, 1994.

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

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

[5] E. Gamma, R. Helm, R. Johnson and J. Vlissides: ."Design Patterns. Elements of Reusable Object Oriented Software.", Addison Wesley, 1995.

[6]http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

[7] F. Lyardet, G. Rossi, D. Schwabe. "Patterns for Adding Search Capabilities to Web Information Systems", Proceedings of EuroPLoP'99. Bad-Irsee, Germany, July 1999. IEEE Computer Society Press.

[8] G. Rossi, A. Garrido, S. Carvalho: "Patterns for object-oriented hypermedia applications". In Pattern Languages of Programs II, Addison Wesley, 1996.

[9] G. Rossi, D. Schwabe and A. Garrido : Design Reuse in Hypermedia Design Applications Development Proceedings of ACM International Conference on Hypertext (Hypertext'97), Southampton, UK, 1997, ACM Press.

[10] G. Rossi, D. Schwabe and F. Lyardet: "Patterns for designing navigable information spaces". Pattern Languages of Programs IV, Addisson Wesley, 1999.

[11] G. Rossi, D. Schwabe, F. Lyardet: "Integrating Patterns into the Hypermedia Development Process". The New Review of Hypermedia and Multimedia, December 1999.

[12]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.

[13] D. Schwabe, G. Rossi: "An object-oriented approach to web-based application design". Theory and Practice of object Systems (TAPOS), October 1998.

[14] In http://www.rational.com/uml/index.jtmpl