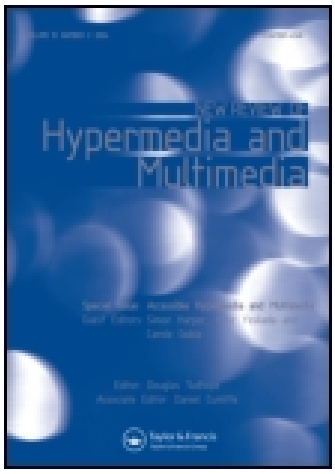


New Review of Hypermedia and Multimedia
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tham20

Integrating patterns into the hypermedia development process

Gustavo Rossi* [a] , Daniel Schwabe [b] & Fernando Lyardet [c]
[a] LIFIA, Facultad de Informática, UNLP , Argentina E-mail:
[b] Depto de Informática , PUC-Rio , Brazil E-mail:
[c] LIFIA, Facultad de Informática, UNLP , Argentina E-mail:
Published online: 20 Jul 2010.

To cite this article: Gustavo Rossi* , Daniel Schwabe & Fernando Lyardet (1999) Integrating patterns into the hypermedia development process, New Review of Hypermedia and Multimedia, 5:1, 59-80, DOI: 10.1080/13614569908914708

To link to this article: http://dx.doi.org/10.1080/13614569908914708

Integrating patterns into the hypermedia development process

Gustavo Rossi*
LIFIA, Facultad de Informática. UNLP, Argentina.
e-mail: gustavo@sol.info.unlp.edu.ar

Daniel Schwabe
Depto de Informática. PUC-Rio, Brazil.
e-mail:schwabe@inf.puc-rio.br

Fernando Lyardet
LIFIA, Facultad de Informática. UNLP, Argentina.
e-mail:fer@sol.info.unlp.edu.ar

*also at UNLM and COINCET

In this paper we show how hypermedia patterns can be combined with design methods in a synergistic way. We first review the state of the art on development methods, emphasizing the OOHDM development process; we next present design and hypermedia patterns as a conceptual tool to record and convey design experience in the hypermedia field; some examples are briefly introduced. Then we discuss two different ways of incorporating hypermedia patterns into the development life-cycle: by including them as new higher level design primitives in a method, and by using patterns as guidelines during the design process. We finally present some further work in this area.

1. INTRODUCTION

The rapid growing of the WWW, and the new applications that are constantly emerging (such as electronic commerce, Internet agents, etc.) are pushing hypermedia further than ever imagined. We are now facing new design and usability problems; thousands of users are now navigating through cutting-edge hypermedia applications that not only present passive information to the end user but also let them query and manipulate sophisticated multimedia databases. Electronic shopping, travel advising, financial applications are some of the new areas that hypermedia application designers must build.

Designing hypermedia applications is a hard task, even harder than designing conventional applications, since they address additional design concerns, such as navigation and multimedia user interfaces. Although there is no silver bullet to cope with the inherent difficulties of this new generation of hypermedia applications, we should try to benefit from experience in software engineering and address them by systematically using state-of-the-art design approaches. We should be able to seamlessly

integrate navigation with transactions or other kinds of application behavior.

We should also bridge the gap between experts and novices. Although nowadays it is straightforward to build simple Web sites using existing tools (like Microsoft's FrontPage or Macromedia's Dreamweaver) and most people can easily create and publish simple Web applications, it is also true that this strategy does not scale to corporate information systems involving Web technology (1). We should be able to convey existing design experience (many Web design problems have already been solved in the hypermedia community) to new designers, if we don't want to end with thousands of legacy Web applications that are difficult to use, maintain or extend.

We have been working with the Object-Oriented Hypermedia Design Method (OOHDM) during the last 5 years (2, 3), using it to design several CD-ROM- and Web-based hypermedia applications. OOHDM combines a set of simple but powerful abstractions, cast as object-oriented primitives, coupled with a growing catalogue of hypermedia design patterns that records the collective experience in designing and implementing this kind of application (4, 5).

In this paper we discuss the problem of effectively reusing design experience by integrating hypermedia patterns into the development process. Though this discussion is rather independent with respect to the underlying methodology, we illustrate our examples using OOHDM. In this way we provide a hook to show how development methods can be improved when design patterns are used as active guidelines during the design enterprise.

The structure of this paper is as follows: we first review the OOHDM development framework and show that even when using a systematic approach to design, we need higher level abstraction primitives to express complex situations appearing in most hypermedia applications. Then we introduce design patterns and show some hypermedia patterns we have discovered. We next discuss two different ways of integrating design patterns into the development life cycle; first by using them as active conceptual guidelines, and second by enriching the primitives repertoire in the method. We finally discuss further work we are pursuing, and present some concluding remarks.

2. THE OOHDM DEVELOPMENT PROCESS

The Object-Oriented Hypermedia Design Method is a model-based approach for building large hypermedia applications. It has been used to

design different kinds of applications such as: Web sites and information systems, interactive kiosks, multimedia presentations, etc.

OOHDM comprises four different activities namely, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. They are performed in a mix of incremental, iterative and prototype-based development styles. During each activity a set of object-oriented models describing particular design concerns are built or enriched from previous iterations. In Figure 1 we show a sketch of the models and activities in OOHDM.

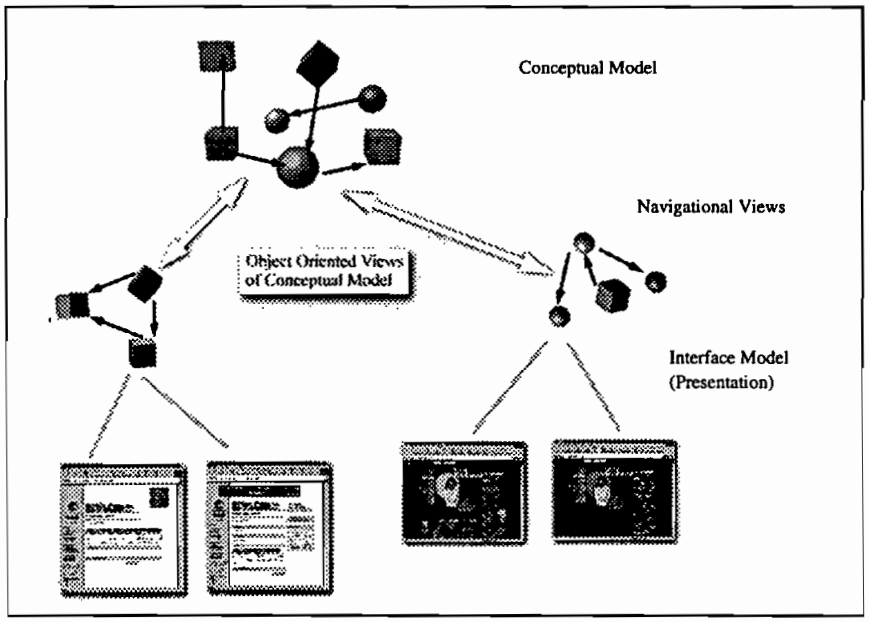


FIG. 1: *OOHDM design models*

Treating conceptual, navigational and interface design as separate activities allows us to concentrate on different concerns, one at a time. Consequently, we get more modular and reusable designs, and we obtain a framework for reasoning about the design process, encapsulating design experience specific to each activity, as we will show in the following sections.

Since design primitives can be mapped straightforwardly onto non object-oriented implementation languages or environments (such as HTML or Toolbook), OOHDM can be used regardless of whether the target system is a pure object-oriented environment one or a hybrid one (as those frequently found in the Internet). In the next sections, we describe each phase in more detail.

2.1 Conceptual modeling

The goal of the Conceptual Design activity is to build a model of the application domain, using well known object-oriented modeling principles, using a notation similar to UML (6). The product of this step is a class schema built out of Sub-Systems, Classes and Relationships. The major differences with UML are the use of multiple-valued attributes, and the use of directions explicitly in the relations.

Conceptual classes may be built using aggregation and generalization/specialization hierarchies. The main concern during this step is to capture the domain semantics as "neutrally" as possible, with very little concern for the types of users and tasks. Modeling conceptual entities as objects allows encapsulating behavior inside them, thus improving the design of complex applications (as for example E-commerce). In this way, the conceptual model can contain complex algorithms and transactions over computational software artifacts. This possibility broadens the range of applications that can be modeled using OOHDM.

When the application involves computations on objects, such as in Web-based Information Systems, the conceptual model will evolve into an object model that will run in the target environment (for example in a WWW server). Those objects can be implemented either as "pure" objects in an environment such as Visual Wave (7) or Java, or as tables in a relational database with corresponding stored procedures implementing behaviors. In this case, navigation objects will act as observers (8) over these application objects.

2.2 Navigational design

In OOHDM, an application is seen as a navigational view over the conceptual model (see Figure 1). This reflects one of the major innovations of OOHDM, which recognizes that the objects (items) the user navigates are *not* the conceptual objects, but other kinds of objects that are "built" from one or more conceptual objects. Therefore, node attributes are defined as object-oriented views over conceptual classes, using a query language similar to the one in (9). This allows a node to be defined by providing access to attributes of different related classes in the conceptual schema. This view is oriented towards a certain class of users and their respective tasks.

The navigational domain of a hypermedia application is defined by a (navigational) schema specifying navigational classes. In OOHDM there is a set of pre-defined types of navigational classes: nodes, links, anchors and access structures. The semantics of nodes, links and anchors are the usual

in hypermedia applications. Access structures, such as indexes, represent possible ways for starting navigation.

Analogously to navigation objects, links reflect conceptual relationships intended to be explored by the final users. Different applications (over the same domain) may contain different linking topologies according the user's profile. For example, an Academic Web site may have a view for students and researchers, and another view for use by administrators. In the second view, a professor's node may contain salary information, which would not be visible in the student's view. In an electronic shopping application (such as http://www.amazon.com) we may also have an administrator's view allowing him to maintain the application by navigating (different views of) the same objects that regular users explore. In both examples we should define a shared conceptual schema and build different navigational views.

OOHDM itself does not prescribe any particular procedure to synthesize navigation designs. To aid the designer, we have developed a method based on user specified scenarios to guide this process (10). According to this method, navigation design proceeds in the following steps (see Figure 2):

1. Determination of user profiles (types of users); identification of user tasks.
2. Scenario collection.
3. Analysis of scenarios, producing a simple diagrammatic representation of the navigation path described in the scenario. In most cases, in this step the designer has to fill in incomplete information, such as implied indexes, missing orderings, exception handling, etc... The designer may use Design Patterns to help in this step (see section 5). A second result of this step is a revision of the preliminary navigation class diagram, as well as a description of how each class used in the scenario is viewed in it.
4. Synthesis of a partial context diagram, specifying navigation in contexts that support the task described in the scenario. Again, design patterns describing known navigation solutions may be employed in this step. (See the discussion on navigational contexts in section 5).
5. Synthesis of final context diagrams, through a process of union and amalgamation of partial schemas produced in the analysis of each scenario in step 4. Eventually, a revision of the navigation class schema may be done. (An example of a context diagram is shown in section 5.)

After collecting scenarios, the designer represents them in a simple graphical notation, to validate it with the user. In sequence, he analyses each one of them, looking for missing information. Users typically omit several types of details, such as access structures, selection attributes to be used in indexes, orderings, treatment of exceptions, etc... The designer can

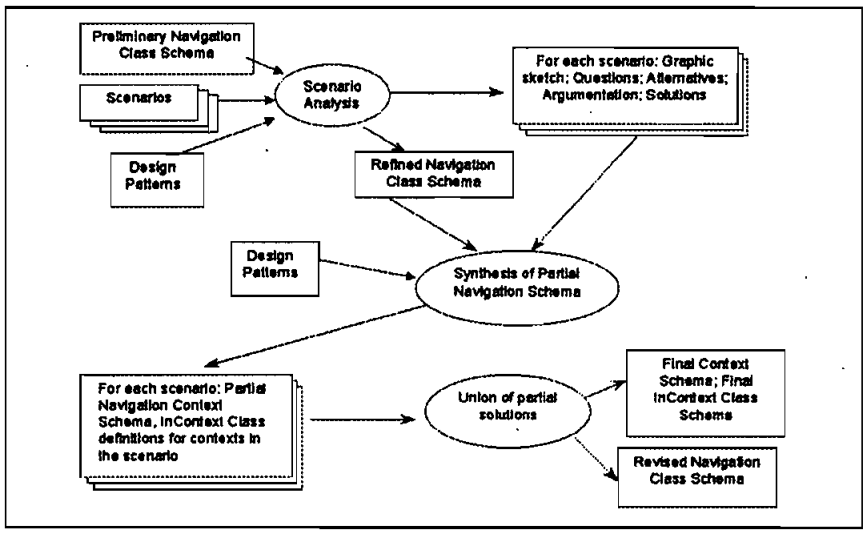


FIG. 2: *Steps in synthesizing navigation design using scenarios*

either supply the missing information out of his own understanding of the problem domain, or ask the user, or use a previously seen solution for a similar problem – effectively applying a design pattern in the process (see section 5.)

2.3 Abstract interface design

In the Abstract Interface Design activity we specify which interface objects the user will perceive. It should be recognized that there is a distinction between navigation operations and interface operations; not everything that happens in the interface is navigation related. Furthermore, it is useful to design interfaces at an abstract level, to achieve, among other things, independence of implementation environment. This strategy is useful even in applications in which nodes are blended with their interfaces (such as for example in the WWW).

The Abstract Interface Specification includes defining the way different navigational objects will look, which interface objects will activate navigation, the way in which multimedia interface objects will be synchronized and which interface transformations will take place. In OOHDM, we use the Abstract Data View (ADV) design approach for describing the user interface of a hypermedia application (11).

3. CAPTURING DESIGN RATIONALE

Using a design method like OOHDM (or others such as RMM (12) or HDM (13)) gives the designer a formal framework to reason on application

entities, hypermedia components (nodes and links) and interfaces in an implementation-independent way. They can record the application's structure using well-proven design primitives (objects in OOHDM, entities and relationships in RMM, etc), so that the design effort can be later reused for further applications. However, expert designers go far beyond the underlying hypermedia design models; when solving complex problems, they usually build more sophisticated architectures. However the rationale for these design decisions remains hidden either in the design diagrams or, even worse, in code. We need a way to record, convey and reuse this critical design experience and this is the place where design patterns come in.

3.1 Using design patterns

Design patterns have been increasingly used in software design (8), and they are now being explored also in the hypermedia field (14, 15). A design pattern systematically names, explains and evaluates an important and recurrent design in software systems. Design patterns make it easier to reuse successful designs and architectures (8). They describe problems that occur repeatedly, and describe the core of the solution to that problem, in such a way that we can use this solution many times, in different contexts and applications.

A design pattern is described by stating the context in which the pattern may be applied, the problem and interacting forces that bring it to life and the collaborating elements that make up the reusable solution. These elements are described in an abstract way because patterns are like templates that can be applied in many different situations.

Patterns show non-trivial ways of using a design paradigm. For example when programming with objects, a novice designer will always encapsulate structure, state and behavior in the same component (an object). He will use sub-classing as the only mechanism for reusing and decoupling responsibilities. Meanwhile, an expert designer usually goes beyond these simple concepts. When he finds an object whose behavior depends strongly on its state, he will decouple the state building a State class hierarchy (Pattern State in (8)). Similarly, when different algorithms may be applied to solve the same problem in a class, he will decouple those algorithms building another class hierarchy (Pattern Strategy in (8)). However, these solutions should not be applied mechanically, but only when certain conditions hold. Design patterns capture, represent and record these contexts in an abstract way, together with the solution to the problem.

It should not be surprising that this idea can be applied in the hypermedia field. Many hypermedia patterns are very similar to the original

architectural patterns (defined by C. Alexander more than 20 years ago (16)), as they show ways to build usable navigable information spaces in the same way as Alexander's patterns taught how to build urban areas and houses. Hypermedia patterns capture those situations in which the standard nodes and links structures are not enough, when the usual navigational semantics are inadequate, or when some sophisticated interface layouts and behaviors are necessary.

We have been mining and describing patterns during the last four years. Our patterns have been reviewed and discussed in pattern writer workshops (2, 17, 18) and we have integrated some of them into OOHDM, as we will show in section 5. We have already collected more than 20 patterns. A more complete catalogue is being built as part of a collective effort of the hypermedia community (15).

The OOHDM design framework has helped us to categorize hypermedia patterns into two groups:

Navigational Patterns. Patterns in this category help in organizing the navigational structure of the application in a clear and meaningful way for the intended readers. Some examples are: "Node as a Single Unit" that dictates how to decide the extent of a node; "News" explaining how to make frequent updates available in dynamic Web sites; "Active Reference" showing how to make indexes and their targets co-exist in the navigational space; "Shopping Basket" indicating how to collect nodes in a repository for further processing; "Set-based Navigation"; "Nodes in Context"; and "Landmark" (described in section 3.2 and 3.3), etc.

Interface Patterns. These patterns give some guidelines for building usable interfaces. Among these patterns we have: "Behavior Anticipation" (described in section 3.4); "Information on Demand" explaining how to make many interface objects available without causing cognitive overhead; "Behavioral Grouping" giving guidelines to organize interface objects; "Process Feed-back" showing how to make the user feel comfortable even when an action takes a long time to complete, etc.

It is obvious that patterns like the ones in (8) can be used during conceptual modeling, but we will not discuss them here, as they do not solve hypermedia-specific problems.

As a background for the forthcoming discussion, we next present a brief discussion of some hypermedia patterns, showing some simple examples in which they have been used. A more detailed description can be found in (17, 5).

3.2 Set-based navigation and nodes in context

Problem

Hypermedia applications usually have to manage collections (a set of cities, a writer's books, the results of a search operation, etc). Naive designers tend to follow closely the golden rules of hypermedia design and only define links between entities that are semantically related; for example they will find the relationship between a book and its author, a painting and its artist, etc. When the link target is not a simply node, they will define an index (for example all of Van Gogh's paintings), and the user will have to move from the index to a target. To move on to another node, he will have to backtrack to the index to find another target. This introduces an unneeded burden when the reader wants to explore the whole set of target nodes. Surprisingly, even well known commercial applications (such as amazon.com) require this type of annoying back-and-forth navigation.

Solution

Consider Sets as first class entities in a hypermedia application. Provide intra-set navigation controls to help the user get the "next" and "previous" element while he is traversing the set. Combine Set-based navigation with proper indexes to make exploration easier. The Set-based navigation pattern shows a simple example in which we will link nodes opportunistically (because they belong to the same set). In this way, two different Van Gogh paintings will be connected by a (set) link allowing them to be reached sequentially. In OOHDM we call each Set a Navigational Context.

When the same node may appear in two different sets, we use the Nodes in Context pattern; this pattern shows how to decouple basic node's contents from those related with the actual navigation context. In this way, each time a node is accessed within a particular context (set) it will be "decorated" (in the sense of the Decorator pattern in (8)) with the information corresponding to that set, such as intra-set navigation links, and contents related specifically with that context. For example, when we access a Van Gogh painting in the context of Paintings on Nature, the "next" and "previous" anchors will have a different meaning compared with the same links in the context of all of Van Gogh's paintings in chronological order. In addition, it is often desirable to provide further comments related solely to a given context. Decoupling the base information from the one corresponding to a context simplifies the design, and helps us think about these navigation paths in a more structured way. In the context of OOHDM sets may arise in different situations and can be easily derived from the structure of the conceptual and navigation schema, together with the information provided by users in scenarios.

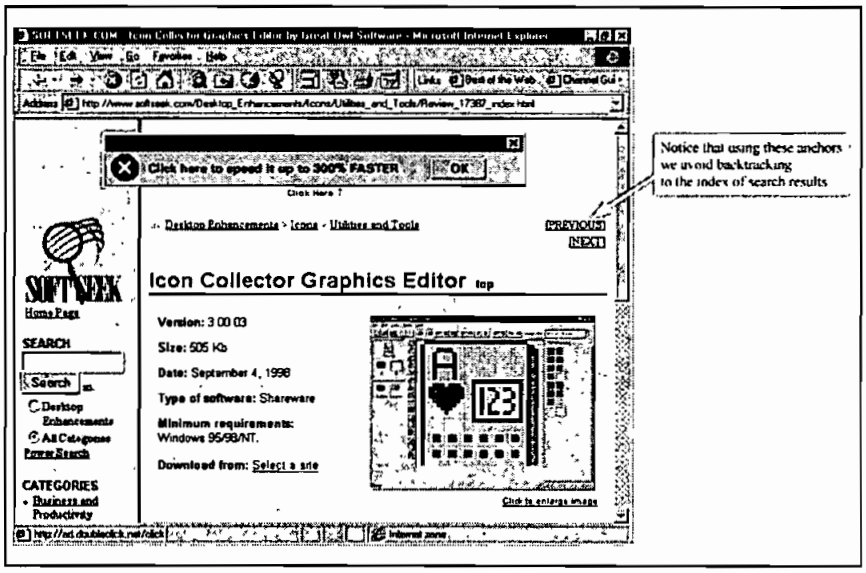


FIG. 3: *Set-based Navigation in www.softseek.com*

In Figure 3 we show how Set-Based navigation has been used in www.softseek.com to organize the results of a search. The Nodes in Context pattern is also used when the same software product appears in two different sets (query results).

3.3 Landmark

Problem

When building complex hypermedia applications, for example an electronic shopping information system, we may have different and perhaps weakly inter-related sub-systems (the book store, the CD store, etc). While designing the navigational schema, a novice designer will follow closely the semantic relationships in the domain. For example there will be links from a CD to the singer, from a book to other related ones, etc. He may also design a home page with an index to all sub-systems. However, we may want the user to be able to go easily from one sub-system to others, or to jump to his shopping list, without having to backtrack to the home page each time.

Solution

Define a set of Landmarks (for example every important sub-system), and make them perceivable and easy to access from each different node in the hyperspace. Define the Landmarks' interfaces uniformly to provide a consistent visual cue for the user. We may define different levels of "landmarking" (e.g., when sub-systems themselves posses significant sub-

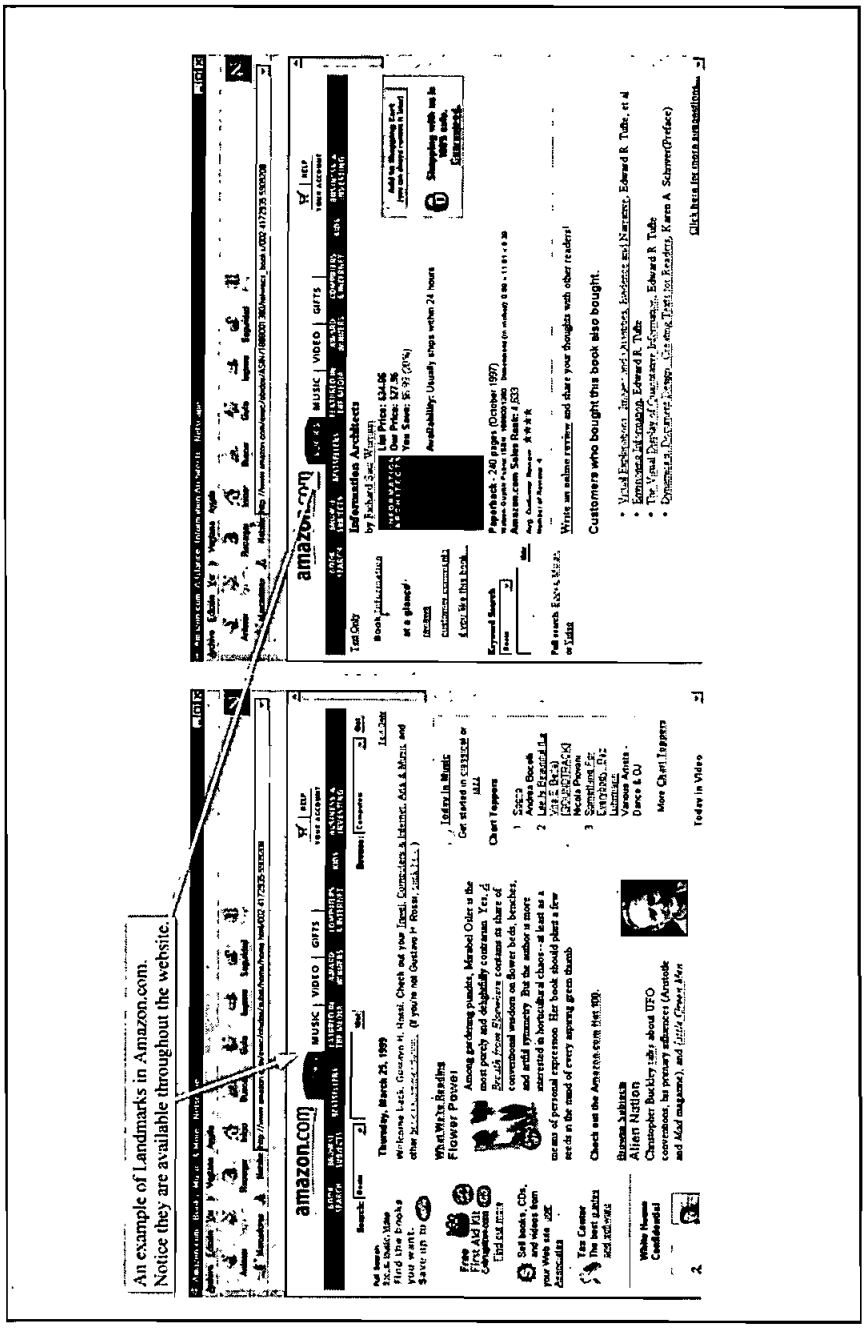


FIG. 4: Landmarks in amazon.com

sub-systems). Landmarks are widely used both in the Web and in CD-ROM-based hypermedia applications. For example in www.amazon.com landmarks are used for each different sub-system, the shopping basket and the management of the user's account (See Figure 4).

3.4 Behaviour Anticipation

Problem

Many times we need to build an interface in which we must provide many different kinds of active objects; the user may find it difficult to understand what the effect of selecting each one of them is. It is usual to find hypermedia readers wondering about whether or not to select a hot-word, a button or a media controller. Moreover, in Web applications, they may be reluctant to follow a link if they do not know in advance what are the contents of the target.

In most interactive applications, including those supporting both navigation and different kinds of media operations, such as zooming, animating, etc., this problem is critical.

Solution

Provide a perceptive feedback about the effect of selecting each interface object. Choose the kind of feedback to provide in such a way that it is non-ambiguous and complete. There are many ways of providing feedback; the simplest one is to use different icons for the cursor as it is positioned over each interface element. In most WWW browsers, for example, a cursor showing a magnifying glass may mean a zoom operation and a hand with a pointing finger signals an anchor. An additional type of feedback that can be provided is to use an area of the screen to put a small text-based explanation about the element in focus at the interface.

There are different ways of providing feedback for navigation controls. In the WWW, for example, we can only see the URL of the destination page; we can enrich this information with a short text-based explanation of the target's contents, as is done in some sites where the status bar is used to display a short explanation instead of the URL. For more recent browsers, this effect may also be obtained with the "Title" attribute of anchors in HTML, which cause the exhibition of a tool tip "balloon" with it contents when the user pauses the cursor over the link (see, for instance, http://www.useit.com/alertbox for a particularly good example of the use of this feature)

Many CD-ROM based hypermedia applications use this design pattern. For example in Microsoft's Ancient Lands, navigation is performed in two steps. When the user clicks on a navigation anchor, a short pop-up

explanation of the target node is provided. The user 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 feedback about the behavior of an interface controller. This pattern is also called "Link destination announcement" in (19).

In this paper we are not just interested in analyzing individual design patterns, but mainly in how they can be synergistically integrated into the development life cycle. In the following sections we first discuss the role of patterns into the design process, and then we show how we incorporated some design patterns into OOHDM.

4. Using patterns as active guidelines during the development process

The increased use of design patterns over the last four years has spurned an interesting debate regarding the relationship among design methods and patterns. This discussion has at least three trends:

1. When and how design knowledge is used (at least when is it expressed as a pattern).
2. How can we formalize the process of pattern mining and documentation.
3. How do we represent higher level abstractions (for example those appearing in the solution part of a pattern) using existing methods.

Though we will not give general answers to these issues in this paper, we can focus the discussion in terms of hypermedia design methods. Although our patterns have been discovered while developing and analyzing hypermedia applications using the OOHDM conceptual framework, they can be used with other methods and development processes.

These hypermedia patterns are intended to form an Alexandrian pattern language, as found in (16), and not a catalog such as the one in (8). Since we conceive the design process as an unfolding of ideas and not as a composition or gluing of pieces, the patterns we found are to be used together synergistically, in a way such that the whole is more than the sum of its parts.

Within the different activities of the OOHDM method, patterns are applied according to the kind of problems they tackle, such as conceptual modeling, navigation or interface. These groups already state a basic (though still incomplete) organization of the pattern language.

Given any of the groups of hypermedia patterns, there is an underlying order among them that allows us to understand how the design fits together. There are three basic rules to state the order of application of patterns (20):

1. Apply a pattern "A" before a pattern "B" when "A" defines a property of an element that is required to exist before applying another pattern "B". As an example, "Node as a Single Unit" has a clear precedence over the "Landmark" pattern, since the latter states which nodes should be reachable from the whole website, and those nodes result from a decision that may be taken using the former pattern.

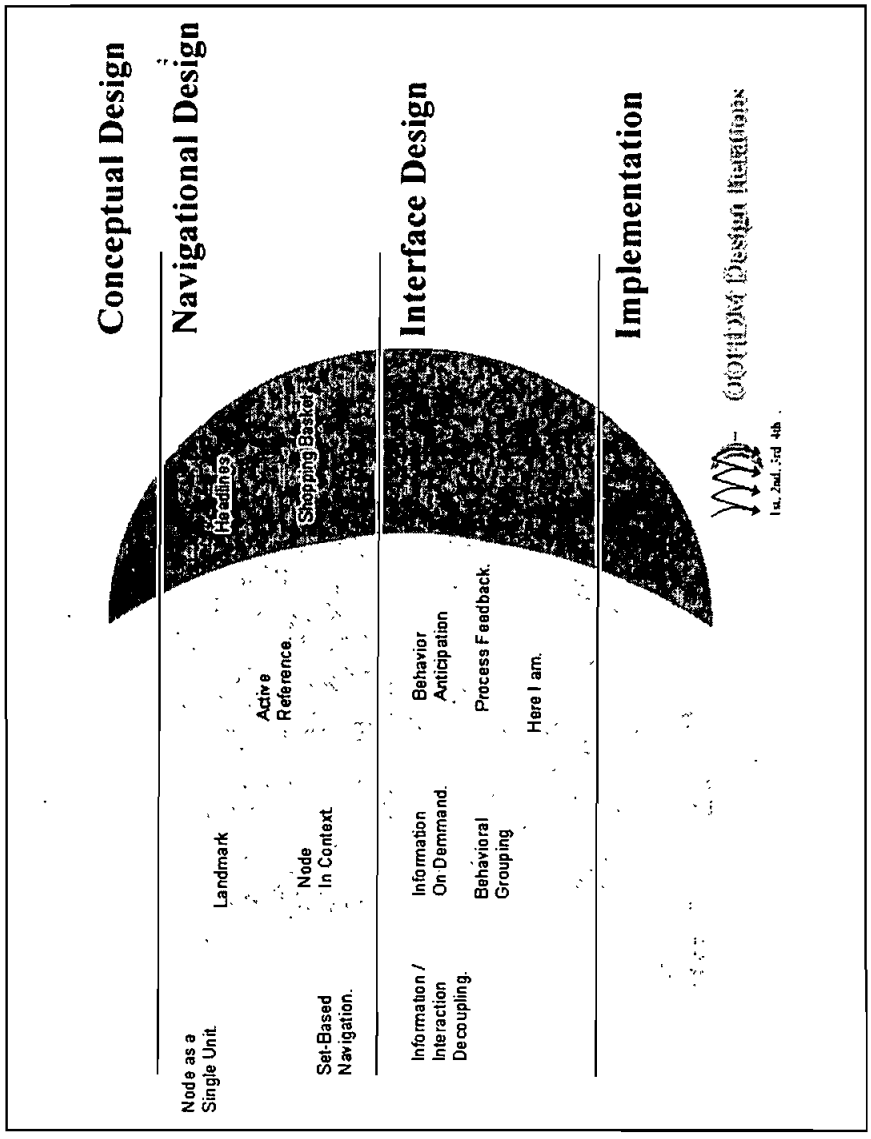


FIG. 5: *Patterns and their placement in the development process*

2. Follow the granularity level. This rule complements the previous one, because sometimes such a sequence may be not clear. For example, in Figure 5, "Information On Demand" precedes "Behavior anticipation", since the sort of problem addressed by the latter (providing the user with hints about the action performed by a UI control) is more refined than the one addressed by the former (dealing with the problem of displaying a large amount of information on a smaller display area). Neither one is a requirement of the other, but the abstraction levels of design problems they tackle are different.
3. When dealing with patterns at the same level in the pattern language try to apply them as closely as possible. Since the order of patterns in a pattern language is not a total order, it may happen that after applying a pattern "A", pattern "B", "C" or "D" can be applied. As an example, in Figure 5 we have "Information/Interaction decoupling" and/or "Behavioral Grouping". After these two patterns, we have a set of three patterns that might be applied "Behavior Anticipation" that can be complemented with the "Here I am" pattern to help the user focus action controls, and "Process Feedback" to provide the user with feedback about the (non-atomic) action being performed. There is a close relationship among them; thus applying them almost simultaneously captures their inter-relationship and complementation.

Under OOHDM, the development cycle follows the principle of fast feedback loops (21) and iterative design refinements in all of its stages. The application of patterns is compatible with this development cycle, since each iteration of design deals with finer-grained design aspects, so there is a mapping among the iterations and the patterns that might be applied in turn. In the following diagram, we show how the iterative loops and design patterns work together.

To give an example of how patterns are integrated into the OOHDM development process we briefly review the method's activities in the context of the design of an academic Web-site, and show how some patterns were applied. For the sake of conciseness we don't discuss patterns appearing at the conceptual level, as they are not specific to the hypermedia field. We also avoid presenting the detailed diagrams of the corresponding scenarios. See (10) for more detail.

Once the conceptual model is ready, and we have determined user profiles and tasks, we begin defining the navigational schema by applying the "Node as a single unit" design pattern to build nodes. Nodes in OOHDM are built as a "copy and paste" of conceptual objects, thus allowing, for example, that Node "Professor" includes the names of the research projects

in which he works. Such names belong to "Research Project" objects, however, this does not mean that we are mixing two different domain concepts, but rather "borrowing" one attribute to simplify the reader's task ("knowing about the Professors activities"). The "Nodes as a single unit" pattern emphasizes defining core nodes as self-contained entities sufficient to support user's tasks.

Proceeding in this fashion, we define the existence of Node classes such as "Course", "Professor", "Research Project", etc. Most links can be induced in a straightforward way from the relationships in the conceptual model, so there will be links to navigate from Professor to Projects, from Courses to other (pre-requisite) courses, etc...

After this initial design, we start facing typical problems that arise once we have drawn the first two scenarios: the same user (e.g. a student) may want to explore all courses of the Department, or read information about research projects or individual professors. It is obvious that both tasks, "finding courses" and "learning about research projects", may involve more than simple indexes as initial access paths. As designers, we will want to help the student to make searches or to explore the available options using different criteria. It may even happen that while exploring a course, the reader may want to read something about an associated project. To solve this design problem we can use the Landmark pattern providing (at least) two different "home pages", one for projects and one for courses. Both pages will be reachable from the whole site (see section 3.3).

Another evident requirement resulting from most tasks is allowing navigation through collections of similar nodes: "Courses", "Projects", "Professors", etc. The "Set-based navigation" pattern can be applied in a straightforward way. Since the same course may be reached in different contexts, e.g., in the list of courses in a research area, or as the set of pre-requisites of another, the "Node in Context" pattern can help to cope with this situation. By applying these patterns, it is possible to specify that a course node, when shown in the "pre-requisite courses of a given course" context, contains the number of credits the student must have to enroll in the course; whereas this information is not shown in the "courses in a research area" context.

Finally we may want to simplify the process of registering for courses. The student can choose different courses, put them in a "basket" and then register (the "Shopping basket" pattern). It may even be useful to allow the student to keep several different "Shopping Baskets", as a way to ease the task of fitting schedules.

The navigation design proceeds then by examining each task, and ensuring that there are navigation objects and paths that support it. This process is aided by the application of navigation design patterns, as outlined in the previous paragraphs.

Once the navigation design is finished, the designer starts working on the interface design. In a manner analogous to the previous phases, many different (interface) patterns can then be applied, while defining the interface appearence of the Web-site, such as "Behavior Anticipation" described in section 3.4. While some of these patterns are specific to hypermedia, most are applicable to the design of interactive systems in general.

Unfortunately most design patterns involve the interaction of many design elements and design documentation may become complex. For example, while using "Set-based navigation" and "Node in Context" we should draw design diagrams showing the navigation paths and the way in which nodes are enriched when accessed in different contexts. In the following section we show how we cast navigation patterns into new OOHDM design primitives. The design decisions outlined in the example discussed in this section will be represented as modeling choices represented directly with OOHDM notation.

5. INCORPORATING PATTERNS INTO THE PRIMITIVES' REPERTOIRE

We have been designing hypermedia applications over the past ten years. In this period, we have observed that several design patterns are so widely used that it would be useful to include them as a primitive concept in a design method. The reason behind this is that such primitives would both allow and induce designers to conceive their designs in terms of those hypermedia patterns. This would constitute a more structured basis upon which to develop designs. There are several such examples, among which we can single out three: Set-based Navigation, Nodes in Context, and Landmarks.

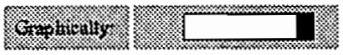
As seen in section 3.2, being able to define meaningful sets of navigation objects is a very powerful mechanism to structure the navigation space. Almost any task that users must accomplish will most likely require manipulating sets of related objects, and therefore being able to easily identify such sets, and to easily define navigation paths within such sets should simplify the designer's task. For this reason, we have defined an OOHDM primitive, "Navigational Contexts", that implements set-based navigation.

As a consequence, designers apply this design pattern directly when using OOHDM, since the navigation space is defined entirely as a collection of Navigation Contexts, enriched with related access structures (indexes). As an example, specifying a "Paintings by Van Gogh with subject Flowers" set is achieved in OOHDM by specifying a Context where its defining property is exactly this (cast in object-oriented notation).

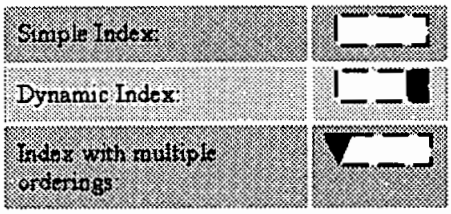
Although contexts are simply sets, there are six different typical ways to define contexts in OOHDM:

- Simple class derived – includes all objects of a class that satisfy some property; e.g., "professors with rank = associate", "paintings with painter = Van Gogh". A variant of this type is the query-based context, where the user defines the property at navigation time.
- Class derived group – is a set of simple class derived contexts, where the defining property of each context is parameterized; e.g. "professors by rank", "paintings by painter" (rank and painter can vary).
- Simple link derived – includes all objects related to a given object; e.g., "courses taught by Professor Smith", "exhibitions where Sun Flowers was presented".
- Link derived group – a set of link derived contexts, each of which is obtained by varying the source element of the link; e.g. "courses taught, by professor", "exhibitions by painting" (professor and painting can vary).
- Arbitrary – is an enumerated set; e.g., a guided tour.

- Dynamic – is a set where the elements change during navigation; e.g. history, shopping basket, etc.



In any of the above, if there is an access structure defined for it, the corresponding graphical notation contains a small black square in the upper left corner. Associated with the contexts, there are access structures (indices). They are denoted graphically by:



In OOHDM, the navigation structure of the application is defined in a context diagram, which shows all the access structures and contexts defined for this application, and the possible navigations between them. Figure 6 shows the context diagram for an Online Magazine. Notice that for each Node class (i.e. Author, Story) we have indicated different kinds of contexts. In addition, we have defined different indexes, such as the Main Menu, the Section Index Menu, etc. In particular, the Authors index allows access to two contexts: "Stories by Authors" and "Authors in Alphabetical Order". Arrows indicate both navigational relationships and possible transitions among navigational contexts.

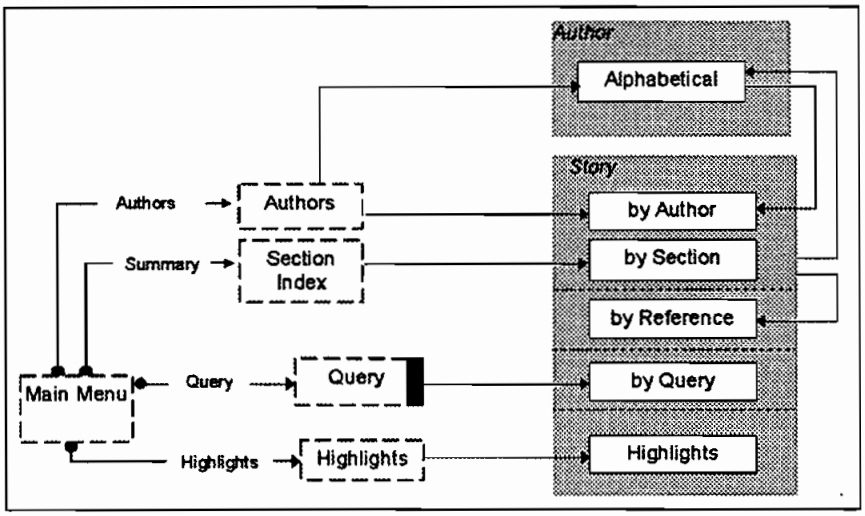


FIG. 6: *Navigational Contexts in an Online Magazine*

In the diagram above, all contexts are direct applications of "Set Based Navigation" and "Nodes in Context". Given the close relation between Set-based Navigation and Nodes in Context, we have also included another primitive in OOHDM that implements the latter, called "InContext" classes. "InContext" classes play the role of Decorators (8) thus simplifying design documents. For example, Stories in the "Highlights" context may have a short biographical note of their authors, whereas Stories in other contexts do not. In any case, it is always possible to navigate from a Story to its Author.

By allowing the designer to succinctly state the particular attributes that objects will have when accessed within given Navigation Contexts and, again, the use of the pattern becomes integral part of the design process.

Finally, as a notational convenience, we have included the ability to signal that certain objects (within certain Navigation Contexts or access structures) should be considered "Landmarks". This effectively states that these objects may be reached from any other object in the navigation space, without requiring the designer to specify every individual link. Therefore, "Landmark" has also become a design primitive used by the designers employing OOHDM. The arrows with small black circles at the origin indicate Landmarks, i.e., their destinations are accessible from anywhere in the website.

6. FURTHER WORK AND CONCLUDING REMARKS

In this paper we have discussed the interaction among hypermedia patterns and design methods. This discussion applies to all hypermedia development methods and processes, although we have used OOHDM to illustrate our analysis. Integrating patterns into the design process is a promising approach for reducing development costs, by making design reuse a reality. Moreover, hypermedia patterns leverage methods by raising the level of abstraction in which designers discuss, providing new and richer design constructs. We are pursuing our work in several complementary directions:

- We keep on seeking for new patterns by exploring successful hypermedia (and Web) applications; a step in this direction can be found in (17).
- We are analyzing how to incorporate other hypermedia patterns into the OOHDM primitives' repository; for example we are exploring how to encapsulate interface behavior (e.g. the one appearing in the "Behavior Anticipation" patterns) in order to express it as a higher level ADV-like structure.
- We are participating in the development of a patterns repository, a collective effort being undertaken by an important group of hypermedia designers (15). This repository will contain patterns, relationships among them, examples of use, surveys, etc.
- We are exploring ways to formalize our patterns, moving them from an Alexander-like notation (that may lead to ambiguities and misunderstandings) to a template-like format as the one used in (18).
- We are enriching other languages and formalisms by incorporating both the OOHDM design view and some of our patterns into its set of primitives. As an example we are working with the WCML (Web Composition Mark-up Language) (22) to enrich it with hypermedia patterns. A first approach can be found in (23).
- We are studying and modeling different application domains (like electronic commerce, on-line newspapers, etc) to design hypermedia

frameworks (24). These frameworks are generic designs that can be used as templates to instantiate different applications in the same domain. Hypermedia frameworks are usually built by composing domain-specific patterns (like Analysis Patterns (25) with more general hypermedia patterns (like Set-based navigation, active reference, etc). Frameworks are a step forward in the direction of building more abstract, reusable hypermedia architectures.

We believe that as the Web is being increasingly used for hosting critical corporate applications we will need solid approaches to cope with the inherent complexity of these kinds of systems. Such systems combine navigation and transactional behaviors and are accessed by means of multimedia interfaces. While design methods and processes are necessary, they are not sufficient: design knowledge must be captured and recorded to ensure that it can be later reused. Hypermedia patterns are an important approach for conveying design experience and to improve their use they must be strong and seamlessly integrated into the development process.

REFERENCES

1. NIELSEN, J. User interface directions for the Web. *Communications of the ACM*, January 1999, pp. 65-72.
2. ROSSI, G., SCHWABE, D. and GARRIDO, A. Towards a Pattern Language for Hypermedia Applications. *Proceedings of the 3rd. Annual Conference on Pattern Languages of Programs*, Monticello, Illinois, September 1996.
3. SCHWABE, D. ROSSI, G. An object oriented approach to web-based application design, *Theory and Practice of Object Systems*, Special Issue on the Internet, 4(4), pp.207-225, 1998.
4. ROSSI, G., SCHWABE, D., and GARRIDO, A. Design Reuse in Hypermedia Applications Development, *Proceedings of ACM International Conference on Hypertext (Hypertext'97)*, Southampton, April 7-11, 1997, ACM Press.
5. ROSSI, G., SCHWABE, D., LYARDET, F. Patterns for designing navigable spaces. In *Pattern Languages of Program Design 4*. Addison Wesley, 1999, forthcoming.
6. The Unified Modeling Language. Rational Corporation. www.rational.com/uml.htm.
7. The VisualWave Programming Environment. Parc Place Systems. In http://www.parcplace.com/products/vwave/vwv_prod.htm.
8. GAMMA, R. HELM, R. JOHNSON and J. VLISSIDES. *Design Patterns: Elements of reusable object-oriented software*, Addison Wesley, 1995.
9. KIM, W. *Advanced Database Systems*, ACM Press, 1994.
10. GÜELL B. N. *User centered design of hypermedia applications*, MSc thesis, Dept. of Informatics, PUC-Rio, 1998 (in Portuguese).
11. COWAN, D. D. P.LUCENA, C. J. Abstract Data Views, an interface specification concept to enhance design for reuse, *IEEE Transactions on Software Engineering*, 21 (3), March 1995.
12. IZAKOWITZ, E. STOHR and P. BALASUBRAMANIAM RMM: A methodology for structured hypermedia design. *Communications of the ACM*, October 1995, pp. 34-44.
13. GARZOTTO, F., PAOLINI, P., SCHWABE, D., HDM - A model based approach to hypermedia aplication design. *ACM Transactions on information Systems*, 11 (1), Jan. 1993, pp. 1-26.

14. ROSSI, G., SCHWABE, D., and GARRIDO, A., Design Reuse in Hypermedia Applications Development, *Proceedings of ACM International Conference on Hypertext (Hypertext'97)*, Southampton, April 7-11, 1997, ACM Press.
15. Second International Workshop on Hypermedia Development. Design Patterns in Hypermedia. In *http://ise.ee.uts.edu.au/hypdev/*
16. ALEXANDER, S. ISHIKAWA, M. SILVERSTEIN, M. JACOBSON, I. FIKSDAHL-KING and S. ANGEL:"a pattern language". Oxford University Press, New York 1979.
17. LYARDET, F., ROSSI G., and SCHWABE D. Patterns for dynamic websites. *Proceedings of Patterns of Programming Languages* (PloP'97), Allerton, USA, 1997.
18. LYARDET, F., ROSSI G., and SCHWABE D. Web-search patterns. *Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP'99)*, Kloster-Irhsee, Germany, 1999.
19. NANARD, M., NANARD, J. and KAHN, P. Pushing reuse in hypermedia design: golden rules, design patterns and constructive templates. *Proceedings of the Ninth ACM Conference on Hypertext (Hypertext'98)*, Pittsburgh, June 1998, pp. 11-20
20. ALEXANDER, S. ISHIKAWA, M. SILVERSTEIN, M. JACOBSON, I. FIKSDAHL-KING and S. ANGEL: *The Timeless Way of Building*. Oxford University Press, New York 1977.
21. NANARD J., NANARD M. Design environments and the hypertext design process. *Communications of the ACM*, August 1995, pp. 49-56.
22. GELLERSEN, H-W., GAEDKE, M. Object-oriented web application development. *IEEE Internet Computing*, 3 (1), January 1999, pp. 60-68.
23. GELLERSEN, H-W., LYARDET, F., GAEDKE, M., SCHWABE, D. and ROSSI, G. Patterns and Components: Capturing the Lasting amidst the Changing. *The Active Web day, British Computer Society Special Interest Group in Human Computer Interaction*, January 1999.
24. LUNA ESMERALDO, L. *Application frameworks for hypermedia*, MSc. Thesis, Dept. of Informatics, PUC-Rio, 1999 (In Portuguese).
25. FOWLER, M. *Analysis Patterns. Reusable Object Models*. Addison Wesley, 1997.