Towards a unified catalog of hypermedia design patterns

D.M. Germán
Department of Computer Science
University of Waterloo
Waterloo Ont. N2L 3G1
dm@csg.uwaterloo.ca

D.D. Cowan
Department of Computer Science
University of Waterloo
Waterloo Ont, N2L 3G1
dcowan@csg.uwaterloo.ca

Abstract

There has been a recent increase in the number of published design patterns for hypermedia. Some of these patterns have been evolving, while others have remained untouched. This paper attempts to list all the patterns currently known, tracing the different publications in which they have appeared. The patterns are scrutinized and refined: some patterns are unified into one; some are deemed special cases of other patterns; some patterns are renamed. At the same time, we propose to rewrite the patterns in a vocabulary that is uniform, and to use similar pattern templates. We then discuss the creation of a design patterns system, which organizes the patterns and assists the designer in the process of recognizing the problems and their potential solutions. Finally, we propose a subset of the patterns which should conform a catalog of basic patterns*; this catalog will attempt to address the most common problems found during the design of hypermedia applications.*

Keywords: hypermedia design, design patterns.

1 Introduction

As the number of design patterns increases, it becomes more difficult to keep track of them, to know whether there is a pattern that solves a determined problem, and, if there exists, to find it.

In order to be useful, patterns should be organized into collections that solve related problems, and the patterns within those collections should form a cohesive group. In such a group, patterns should use a common vocabulary, should not be redundant (i.e. two patterns do not solve the same problem in similar ways) and they should be written in a way that maximizes their utility.

So far, there is no publication or repository that collates all the different design patterns. A designer with enough initiative will have to scan a stack of publications in order to find what design patterns exist, what problems they solve, and what are their proposed solutions. These is certainly impractical. Furthermore, patterns have not been scrutinized yet. They have to be analyzed and, in many cases, improved. As Gamma et al. stated “finding patterns is much easier than describing them” [7].

2 A Compendium of Patterns

The first hypermedia design patterns were presented by Rossi [19]; since then many more have been published. They range from generic “golden rules” [17] to specialized patterns for collaborative design [22]. We present a list of published patterns to date. It is based upon Annex 1 from [16] and further extended with newer design patterns. The list has been ordered by pattern name and includes a brief description and the publications in which it has appeared. In this list, there are no patterns sharing the same name. In the case of those patterns which have been published more than once by the same group of authors under the same pattern name (for example, *Active Reference* and *News*), we assume that the new versions are refinements of the same pattern, hence we consider it to be the same one. We have decided not to rewrite their descriptions. The ones listed here are the same as in the original papers (in some cases, they were abbreviated for the sake of space). As a consequence, some descriptions are too vague, others are stated as a question, and overall, there is no consistency from one pattern description to another.

Active Reference: Provides a perceivable and permanent reference about the current status of navigation. [8, 15, 19]

Avatar: How can a self-representation of users be provided in an intuitive way? [22]

Behavioral Anticipation: How do you indicate the effect or consequence of activating an interface object? [8]

Behavioral Grouping: How to organize the different types of controls in the interface so the user can easily understand them? [8]

Clustering: Avoid the presentation of more than 7 items simultaneously. [17]

Collector: How to make a set of elements behave in the same way depending on one element. [4]

Communication Channel: How can information be exchanged that is not directly related to the document content? [22]

Component Layout: Several artifacts need to be arranged in respect to their audio-visual properties. [3]

Compound: How to describe the resulting behavior when two elements are joined to work together. [4]

Constructive Templates: It is a generic specification which makes it easier for the developer to build up actual hypermedia structure and populate it with its data. [17]

Contour: Cycles overlap on each other, allowing free movement from one cycle to another. [1]

Counterpoint: Two "voices" alternate, interleaving, giving the reader the option to either follow one or to jump from one to the other. [1]

Cycle: The reader returns to a previously visited node and departs along a new path. [1]

Decorator: Provides a flexible alternative to subclassing for extended functionality. [6]

Dynamic Configuration Pattern: How to provide the user with the means to perform a selection over a set of options that might be arbitrarily large, while keeping track of them, and then validate them. [14]

Glue: Joins a number of multimedia artifacts into a single composite artifact. [3]

Group Location Awareness: How can we provide a permanent reference about the user's current locations in the collaborative hypermedia space? [22]

Hierarchical Structure through Navigation Side Bars: Provides a way to graphically distinguish between hierarchical structure and cross-references when there is only one underlying link type available, as on the Web. [5]

Hyper-Book: Presents a hypertext version of a sequential document (book, article, report). [11]

Hyper-Map: Provides an interface to geographical information. [11]

Information Factoring: Presents information needed by the reader to understand a given topic/information unit. [15]

Information on Demand: Lets users decide which items they want further described in the context of the same node. [8, 15, 19]

Information-Interaction Coupling: How do we make clear what is the object affected by a control in a node's interface? [8]

Information-Interaction Decoupling: How do you differentiate contents and various types of controls in the interface? [8]

Landmark: Provide direct access to critical sub-systems in the WIS. [20]

Link Creation Method: When is it better to create static links, and when is it preferable to create links through computations? [8]

Link Destination Announcement: Avoids unnecessary link firing by providing information about the destination. [17]

Logical Glue: Small information sets need to express meaningful structure to avoid being perceived as an arbitrary grouping. [17]

Logical Glue Consistency: Homologous strategies should be used in similar parts of the design in order to help the reader build up a mental model of the structure. [17]

Mirrorworld: Provides two or more views of the same information. [1]

Missing Link: Suggests a link that does not exist. [1]

Montage: Several distinct writing spaces appear simultaneously, maintaining their separate identities. [1]

Navigational Context: Provides the user with closed navigational subspaces containing context-related guidelines and relationships. [8, 15, 19]

Navigational Feint: Establishes the existence of a navigational opportunity that is not meant to be followed immediately. [1]

Navigational Observer: Decouples the navigation process from the perceivable record of the process. [19]

Neighborhood: Establishes an association among nodes through proximity, shared ornament, or common navigational landmarks. [1]

News: Allows easy access to new information items as the WIS grows. [14, 20]

Node Creation Method: When is it better to create nodes statically, and when is it preferable to create nodes dynamically? [8]

Node as a Single Unit: How do you decide the extent of a node? [8, 15]

Partitioned Incremental Development: Provides the basis for development of hypermedia in an incremental manner, supporting progressive integration and delivery of components. [13]

Process Feed-Back: How do we keep users informed about the status of the interaction in such a way that they know what to expect? [8]

Session: How can we structure collaboration between users and groups of users? [22]

Set-based navigation: Organizes the information in sets of related information items. Provide intra-set navigation capabilities [20]

Shopping Basket: Keeps track of user selections during navigation, making these selections persistent to process them when the user decides to. Decouple product selection from product consumption and/or processing. [20]

Sieve: Sorts readers through one or more layers of choice in order to direct them to a given section. [1]

Split/Join: Knits two or more sequences together. [1]

Tangle: Confronts the reader with a variety of links without providing clues to guide the reader's choice. [1]

Template: A need exists to produce a collection of composite artifacts similar in structure and contents. [3]

User Role: How to represent the different behaviors a user shows, depending on the collaborative context? [22]

Virtual Product: Displays a product as part of an electronic catalog. [11]

Virtual Room: How can we structure collaboration between users and groups of users in a natural and intuitive way? [22]

These patterns have come from a relatively small number of authors and, respectively, a small number of papers. The following is a list of the papers in which the afore mentioned patterns are defined. For each author or group of authors, their papers are listed and for each paper, its patterns.

It is important to mention that some papers list more patterns than the listed herein. We have considered that such patterns are not directly relevant to the field of hypermedia design and development. For example, Garrido et al. [8] includes a number of patterns intended for the development of hypermedia platforms, rather than hypermedia content.

Bernstein: [1] Contour, Counterpoint, Cycle, Mirrorworld, Missing Link, Montage, Navigational Feint, Neighborhood, Sieve, Split/Join, Tangle.

Cybulski et al.: [3] Component Layout, Glue, Template.

Diaz et al.: [4] Collector, Compound.

Gaedke et al.: [6] Decorator.

Garrido et al.: [8] Active Reference, Behavioral Anticipation, Behavioral Grouping, Information on Demand, Information-Interaction Coupling, Information-Interaction Decoupling, Link Creation Method, Navigational Context, Node Creation Method, Node as a Single Unit, Process Feed-Back.

German et al.: [11] Hyper-Book, Hyper-Map, Virtual Product.

Lowe: [13] Partitioned Incremental Development.

Lyardet et al.: [14] Dynamic Configuration Pattern, News; [15] Active Reference, Information Factoring, Information on Demand, Navigational Context, Node as a Single Unit.

Nanard et al.: [17] Clustering, Constructive Templates, Link Destination Announcement, Logical Glue, Logical Glue Consistency.

Rossi et al.: [19] Active Reference, Information on Demand, Navigational Context, Navigational Observer; [20] Landmark, News, Set-based navigation, Shopping Basket.

Schummer et al.: [22] Avatar, Communication Channel, Group Location Awareness, Session, User Role, Virtual Room.

Øesterbye: [5] Hierarchical Structure through Navigation Side Bars

It is interesting to note that the group leaded by Rossi and Schwabe are the only ones with more than one paper in this list [8, 14, 15, 19, 20], and have written close to 33% of the patterns herein.

3 Refining Known Hypermedia Design Patterns

Compared to software engineering –where there are several published books with design patterns– hypermedia design patterns are in an early stage: they are few, they are scattered in the literature and they have barely been scrutinized. Paolini and Garzotto stated [18] "we have not seen many real 'booklets' of design patterns, ready to be used".

In order to create an effective catalog of design patterns, it is necessary to collate, organize, analyze, select, combine, and perfect them. The following subsections describe tasks that should take place in order to achieve these goals.

3.1 Unifying patterns

Several patterns try to solve the same problem. In some cases, they do it with a similar approach; in other cases, they attack the same problem from different perspectives or at different stages of the design process. The patterns space should be analyzed to find patterns that are similar. Different types of unifications could take place:

- Some patterns provide different views of the same problem, but are not different enough to stand by themselves. These patterns should be unified into a single "super" pattern that includes the insight of all its components.
  - *Composite Consistence*. *Glue* tries to solve the problem of creating composite obj̃ects; *Logical Glue* specifies that groups should have a meaningful structure; *Logical Glue Consistency* specifies there should be consistency in the way groups are created. *Component Layout* indicates how "several artifacts need to be arranged in respect to their audio-visual properties". All these patterns are combined into the *Composite Consistence* pattern will attempt to solve the problem of "creating composite objects, with a meaningful internal structure and done in a consistent way across the application" by gathering the knowledge encapsulated in each one of them.
- Some patterns provide insight that can enhance similar patterns.
  - *Component Layout* and *Node as a Single Unit* should include –if deemed appropriate– the pattern *Clustering* which claims that it is optimal to create clusters of 7 items. The latter pattern is too simple to stand by itself.
- Some patterns are the same under different names:
  - *Template* and *Constructive Template* represent the same design pattern: how to instantiate multiple navigational nodes from variable data. *Constructive Template* should disappear in favor of *Template*.
- Some patterns are special cases of another:
  - *Link Destination Announcement* "avoids unnecessary link firing by providing information about its destination" while *Behavioral Anticipation* "indicates the effect or consequences of activating an interface object". Clearly, the latter comprises the former. Nonetheless both are important and should not be combined into one pattern; instead, it should be made explicit that *Link Destination Announcement* is a particular case of *Behavioral Anticipation*. The latter should be listed in the *related patterns* section of the former.
  - *Shopping Basket* is an specialized version of *Collector*.
  - *Hierarchical Structure through Navigation Side Bars* is a special case of *Navigational Context*.

These steps will lead to fewer and better patterns. At the same time, some patterns will appear to be more important and generic than others.

3.2 Renaming some patterns

As Schmidt states in [21] "pattern names should be chosen carefully". A pattern name should be precise and concise, and it should clearly convey its purpose. There are several cases in which it is clear that a better name will improve a pattern dramatically:

- *Representation of the User*. According to the Oxford English Dictionary, one of the definitions of the word avatar is "manifestation in human form; incarnation". Avatar is a poetic name for a pattern that attempts to provide a representation of the user in the hypermedia application. Unfortunately, avatar is not a common word –except, probably, in the virtual reality arena– and as a consequence, many designers draw no information from its name. A better name for *Avatar* could be *Representation of the User*.
- *Hyper-translation of Sequential Book*. In some cases, pattern names convey a different meaning from what the pattern really is. *Hyper-book* should be named *Hyper-translation of Sequential Book* to stress the fact that it describes the process of translating a book to its hypermedia version, and not the creation or presentation of a hyper-book which is created specifically for a hypermedia platform.

3.3 Using a common vocabulary

The hypermedia designers in general have not decided on a common vocabulary and this is reflected in the pattern descriptions. For instance, in *Node as Single Unit* nodes are composed of “objects’ attributes”; while in *Glue*, nodes are “composite artifacts” which are composed of “atomic artifacts” or “composite artifacts”; *Compound* calls them “elements”. In order to unify the patterns it is necessary to use a common hypermedia language. Paolini and Garzotto [18] suggested the use the concepts, primitives, and notation of HDM –Hypertext Design Model [9]. Several patterns (Rossi’s, Lyardet’s, Garrido’s, and Schummer’s) have been described using OOHDM. Whichever notation is used, it has to be used uniformly in all the design patterns. We suggest the use of the notation from the widely accepted Dexter Reference Model [12] with enhancements from OOHDM and HDM, which are the two predominant design methodologies.

3.4 Rewriting some patterns in their entirety

In some cases, the patterns are ambiguous or obscure. For example, the problem description of *Collector* states “How to make a set of elements behave in the same way depending on one element”; its motivation reads: “the shopping basket is used for collecting things which then ’follow’ the shopping basket whenever it is is moved”; later, in its solution, it claims: “the shopping [basket] acts as a collector while the products are the collectables” [4]. This pattern’s problem description would be better if it was rewritten as “one that deals with the process of allowing the user to select and collect ’objects’ which follow the reader and they can later can be reviewed”. In the case of the *Shopping Basket* (a special case of collector, as we described earlier), the objects are further reviewed for inspection, removal of objects, calculation of invoice, etc.

Other patterns are described with a succinct manner that makes it difficult to understand the pattern quickly. The first paragraph of the description of *Tangle* [1] states:

> *The tangle confronts the reader with a variety of links without providing sufficient clues to guide the reader’s choice. Tangles can be used purely for their value as intellectual amusement, but also appear in more serious roles. In particular, tangles can help [to] intentionally disorient readers in order to make them more receptive to a new argument or an unexpected conclusion.*

This section of the pattern could be rewritten as:

Name:	Tangle
Intent:	To disorient readers in order to make them more receptive to a new argument or an unexpected conclusion.
Solution:	To confront the reader with a variety of links without providing sufficient clues to guide the reader’s choice.

The information is the same, but its new organization makes it easier to browse. This is especially valuable in large catalog of patterns, in which the designer might not have enough time to read the entire description of each pattern. All Bernstein’s patterns [1] should be rewritten.

4 Patterns should be classified and organized into a pattern system

The list in section 2 is classified by name and it does not gather similar patterns together. Users trying to find a solution to a given problem will have to scan the complete list first, even if there is no pattern for the problem that they face. In order to be useful, a catalog should be organized and its patterns classified. With a good classification, designers will find the pattern they need easier or realize there is no pattern for their problems; without having to check every single pattern. Finally, by creating collections of similar patterns, these can be compared promoting refinement and potential unification.

Buschmann et al. [2] defines a pattern system in the scope of software architectures. Their definition can be easily adapted to hypermedia application: a pattern system is as a collection of patterns, together with guidelines for their implementation, combination and practical use in hypermedia development.

Buschmann et al. describe six conditions that a pattern system should satisfy:

1. It should comprise a sufficient base of patterns.
2. It should describe all its patterns uniformly.
3. It should expose the various relationships between patterns.
4. It should organize its constituent patterns.
5. It should support the construction of hypermedia systems.
6. It should support its own evolution.

The 51 patterns listed in this paper arguably form a sufficient base of patterns to start the creation of a pattern system. Section 2 attempts to address point two.

Many authors have proposed pattern classifications. Garrido et al. [8] and Rossi et al. [19] divide them based upon the stage of the design process in which they are most likely used: “navigational design” and “interface design”, which correspond to the navigational design and interface design stages of OOHDM. Germán and Cowan [10] proposed a classification based upon the objective of the pattern: “presentational”, “structural”, and “support”. Bernstein’s patterns [1] are clearly rhetorical although they can be perceived as structural. Nanard and Nanard [17] proposed a new type of patterns called “golden rules”. Lyardet et al. [14] proposed to classify them based on the issues they target: “information organization”, “interface organization” and “implementation”. Lowe [13] proposed a new category called “process patterns”. Each of these papers classifies only the patterns that it presents. None attempts a wider taxonomy.

Paolini and Garzotto [18] proposed a global classification based upon whether the patterns specify “user requirements”, “generic design ideas”, and “well defined design ideas”; unfortunately, they did not classify any design patterns. Nanard and Nanard [16] proposed a classification based on 3 major dimensions: “hypermedia design and development”, “hypermedia application”, and “hypermedia system”; each of these dimensions is further divided into subdimensions (e.g. hypermedia system is divided into architecture, interaction and production). Although they include a list of design patterns, they did not try to classify them.

Each of these classifications provides valuable insight. It is important, however, to create one classification that is as comprehensive as possible and, at the same time, conceptually simple. An adequate classification will make it easier for designers to identify, given a problem, the proper design pattern to solve it.

The creation of a taxonomy for design patterns is a thorny issue. Patterns can be classified in an seemingly infinite number of ways. For instance, they can be classified by problem they try to solve; by the stage of the design process in which they are most likely to be used; depending upon the application domain they can be used, etc.

A successful taxonomy should be simple and have as few classifications as possible. Such classifications should gather patterns that satisfy certain properties. The question that arises is how to define such classifications and properties and how to determine if a pattern belongs to one.

A framework for the development of hypermedia applications can be the answer. A framework will incorporate design patterns into the development process, in order to determine when and how can patterns be used. The design process will provide natural classifications depending on stage of the design process they can be used, the problems typically found during development, and how to recognize which pattern to use given the circumstances.

Rossi and Schwabe have been working on one based on OOHDM [8, 19, 20]. Unfortunately, they have not yet incorporated patterns written by other authors into their framework.

4.1 A basic design patterns system

It is important to have many patterns that solve many problems. In order to be useful, however, they should be categorized in order of importance: some patterns are more likely to be used than others. The obvious result of this categorization is the creation of a basic design patterns system. Basic design patterns are those which are commonly used in any type of hypermedia design; therefore, are helpful to any designer. Such system should incorporate only a reduced number of patterns in order to make it easy for a designer to remember them and apply them successfully.

More specialized systems can be created which target more specific problems, including those in specific application domains.

We propose the following patterns as basic. We collate them in groups to make this list more readable:

- Architectural. Patterns which assist the developer in the design of the overall structure of the application – its graph structure: *Cycle, Counterpoint, Mirrorworld, Tangle, Sieve, Montage, Missing Link, Neighborhood, Split/Join*.

- Component Construction. These patterns solve problems related to the way basic components are combined into more complex ones: *Node as a single unit, Component Layout, Composite Consistence*.

- Navigation. These patterns address problems related to the way an application is crosslinked and the way a reader is guided through it: *Active Reference, Navigational Context, Navigational Feint, Navigational Observer, News, Node as a Single Unit, Decorator, Set-based Navigation, Landmark*.

- Presentation. They are related to the way the content is presented to the reader in the final run-time system: *Behavioral Grouping, Information-Interaction Coupling, Information-Interaction Decoupling, Decorator*.

- Behavior/User Interaction: Patterns that solve problems related to the way the user and the application interact: *Behavior Anticipation, Information on Demand, Link Destination Announcement, Process Feed-Back, Collector*.

This list does not pretend to be definite. As more patterns are discovered and many are further refined, this list will change. The Gamma et al. collection (or *Gang-of-Four patterns*, the first widely accepted catalog of design patterns for software engineering) went through a period of evolution from their first appearance, in 1991, to their final publication, in 1995. For instance, some patterns were added; some had their names changed; and the average size of each pattern grew from two to ten pages.

5 Conclusions

As the software engineering community has done, it is important to promote the discovery and refinement of design patterns. This could be accomplished with "Pattern Conferences" in which the goal is to find, describe, and refine design patterns. Another option is the collaborative effort of pattern writers using the Internet as a communication medium.

The refinement process involves the following tasks:

- The unification of patterns.
- Some patterns should be renamed.
- The vocabulary used to describe them should be standardized.
- Some patterns should be rewritten in their entirety.
- A catalog of primary patterns should be created.
- They should be classified and cataloged.

Finally, the advice of designers is needed. Apparently, all design patterns have been created by researchers in the area of hypermedia. These researches are likely designers themselves. Nonetheless, their research orientation will make their views different from the professionals in the field. Because design patterns are intended to be used by hypermedia designers, it is indispensable that they are involved in their discovery and refinement.

The ultimate goal of this process is to produce ready-to-use catalogs of hypermedia design patterns.

We are setting up a repository of patterns on the World-Wide Web (http://aries17.uwaterloo.ca/hdp/index.html), in which we list each of the patterns here included. Our goal is that this repository serves as a communication medium between pattern miners –those who discover them– and pattern users. It will also keep track of new patterns and changes on the current ones. Finally, using user feedback, we expect to be able to refine their descriptions and select those ones which are the most useful to the largest percentage of developers, and use these patterns as a basis for a basic patterns system.

Acknowledgments

We want to thank our reviewers, whose long, insightful comments were very helpful in improving the quality of this manuscript.

References

[1] M. Bernstein. Patterns of hypertext. In *Proceedings of the Ninth ACM Conference on Hypertext*, Hypermedia Application Design, pages 21–29, 1998.

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stad. *Pattern-Oriented Software Architecture – A System of Patterns*. John Wiley, 1996.

[3] J. L. Cybulski and T. Linden. Composing Multimedia Artefacts for Reuse. In *Proceedings of The 4th Pattern Languages of Programming Conference*, 1998.

[4] A. Diaz and R. Melster. Patterns for Modelling Behavior in Virtual Environment Applications. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[5] K. Øesterbye. Hierarchical structure through navigation side bars. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[6] M. Gaedke, F. Lyardet, and H. Werner-Gellersen. Hypermedia Patterns and Components for Building better Web Information Systems. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Professional Computer Series. Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[8] A. Garrido, G. Rossi, and D. Schwabe. Pattern Systems for Hypermedia. In *Proceedings of The 3th Pattern Languages of Programming Conference*. University of Washington, 1997.

[9] F. Garzotto, P. Paolini, and D. Schwabe. HDM – A model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1):1–26, 1993.

[10] D. M. German and D. Cowan. Hypermedia Design Patterns. In *7th. Mini Euro Conference on Decision Support Systems, Groupware, Multimedia and Electronic Commerce*, April 1997.

[11] D. M. German and D. Cowan. Three Hypermedia Design Patterns. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[12] F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, Feb. 1994.

[13] D. B. Lowe. Hypermedia Process Assessment Tasks: Patterns of Inspection. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[14] F. Lyardet, G. Rossi, and D. Schwabe. Patterns for Dynamic Websites. In *Proceedings of The 4th Pattern Languages of Programming Conference*, 1998.

[15] F. Lyardet, G. Rossi, and D. Schwabe. Using Design Patterns in Educational Multimedia Applications. In *Proceedings of EDMedia'98*, 1998.

[16] J. Nanard and M. Nanard. Toward an Hypermedia Design Patterns Space. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[17] M. Nanard and J. Nanard. Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*, pages 11–20. ACM Press, June 1998.

[18] P. Paolini and F. Garzotto. Design Patters for the WWW hypermedia: problems and proposals. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.

[19] G. Rossi, D. Schwabe, and A. Garrido. Design Reuse in Hypermedia Applications Development. In *Proceedings of the Eighth ACM Conference on Hypertext*, Hypertext Design, pages 57–66, 1997.

[20] G. Rossi, D. Schwabe, and F. Lyardet. Improving web information systems with navigational patterns. In *Proceedings of the 8th International World Wide Web Conference*. W3C, Elsevier, May 1999.

[21] D. C. Schmidt. Using design patterns to develop reusable object-oriented communication software. *Communications of the ACM*, 38(10):65–74, Oct. 1995.

[22] J. Schummer and C. Schuckmann. Collaborative Hypermedia Design Patterns in OOHDM. In *2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia*, 1999.