A System of Patterns for Web Navigation

By

Mohammed Abul Khayes Akanda
B.Sc., Bangladesh University of Engineering and Technology, 1999

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Mohammed Abul Khayes Akanda, 2003

University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy
or other means, without permission of the author.
ABSTRACT

Design patterns have the potential to provide the reuse of the same solution over and over again in different instances of the same problem, to improve the quality of an application and to reduce the cost of design and development. Design patterns are becoming popular for both software and hypermedia applications. The number of hypermedia design patterns is increasing and different authors are writing them in different notations. To be effective, these patterns should be collected into a repository writing them in a uniform notation and the relationships among the patterns should be identified which will increase the understandability and usability of those patterns. In this thesis, design patterns are collected by tracing different publications; and a system of patterns for Web navigation is created by forming a catalog in a uniform notation, identifying the relationships among the patterns, describing their evolution and refinements and providing guidelines for implementation of the patterns. The patterns included in this system are implemented into a framework that is intended to be used to navigate an information space on the Web.
CONTENTS

LIST OF FIGURES ............................................................... vi
ACKNOWLEDGMENTS ............................................................ viii

1. Introduction ..................................................................... 1
   1.1 Design Patterns for Hypermedia ................................. 1
       1.1.1 Navigational Design Pattern .............................. 1
   1.2 Thesis Organization .................................................... 2

2. Background Study .......................................................... 3
   2.1 Motivation ................................................................... 3
       2.1.1 Design Pattern Language ................................ 4
       2.1.2 Discovering Patterns ......................................... 6
       2.1.3 Selecting a Pattern ............................................ 7
       2.1.4 Frameworks and Patterns ................................. 8
       2.1.5 Using a Pattern ................................................ 9
   2.2 Hypermedia ............................................................... 10
       2.2.1 Web Information Systems ................................. 11
       2.2.2 Classification of Hypermedia Design Patterns .... 12
       2.2.3 Functional Characteristics of Hypermedia Application and Design Patterns ......................................................... 14

3. System of Patterns .......................................................... 16
   3.1 System of Patterns ...................................................... 16
   3.2 A System of Patterns for Web Navigation ..................... 17
       3.2.1 Base Patterns ................................................... 18
       3.2.2 Uniform Description ........................................... 19
       3.2.3 Relationships among the patterns ......................... 19
       3.2.4 Support for the construction of hypermedia systems .................. 19
       3.2.5 The refinement of the hypermedia design patterns ........ 19
   3.3 Evolution .................................................................. 24
4. Catalog of Patterns .................................................. 26
   4.1 Introduction .................................................. 26
      4.1.1 The UWE Methodology ..................................... 27
         4.1.1.1 Modeling Elements ................................... 27
   4.2 Relationships among the patterns .......................... 29
   4.3 Set Based Navigation ........................................ 30
   4.4 Landmark .................................................... 32
   4.5 Active Reference ............................................ 34
   4.6 Visited Objects .............................................. 37
   4.7 News ........................................................ 39
   4.8 Guided Tour .................................................. 42
   4.9 Simple Search Interface ..................................... 45
   4.10 Selectable Search Space .................................... 49
   4.11 Shopping Basket ............................................ 52

5. Java Implementation .............................................. 56
   5.1 Introduction .................................................. 56
   5.2 Component: .................................................... 56
      5.2.1 Navigational Framework ................................ 57
   5.3 Parameterization ............................................. 57
   5.4 Objectives of the framework ............................... 58
   5.5 N-tier Architecture .......................................... 61
   5.6 Using the patterns .......................................... 62
      5.6.1 Integration of the patterns ...................... 63
      5.6.2 Case Study-1: .......................................... 63
      5.6.3 Case Study-2: .......................................... 64
   5.7 Restrictions .................................................. 65

6. Conclusions ...................................................... 68
   6.1 Observations ................................................ 69
   6.2 Contributions ............................................... 71
   6.3 Future Work ................................................ 71

A. A Compendium of Patterns ...................................... 77
   A.1 List of Patterns ............................................. 77
B. Interfaces ................................................................. 85

B.1 Interfaces of the Components ................................. 85
LIST OF FIGURES

2.1 Hypermedia Pattern Space ........................................... 13
3.1 Double Tab in www.amazon.com ...................................... 20
3.2 Double Tab in www.apple.com ........................................ 21
3.3 Unified Search .......................................................... 22
4.1 Navigational Class ...................................................... 28
4.2 Presentation Class ...................................................... 29
4.3 Design Pattern Relationships ......................................... 30
4.4 The Structure of Set Based Navigation ............................... 31
4.5 Set Based Navigation in http://www.chapters.indigo.ca .......... 32
4.6 The Structure of Landmark ............................................ 33
4.7 Landmark in http://www.chapters.ca ................................. 34
4.8 The hierarchical structure in an example information space .... 35
4.9 The Structure of Active Reference .................................. 36
4.10 Active Reference in www.chapterns.ca ............................... 37
4.11 The Structure of Visited Objects ..................................... 38
4.12 Visited Objects in www.amazon.com .................................. 39
4.13 The structure of News .................................................. 40
4.14 News in www.chapters.ca ............................................. 41
4.15 The navigation structure using Guided Tour ....................... 43
4.16 The structure of Guided Tour ......................................... 43
4.17 Guided Tour in http://www.globeandmail.com ................. 45
4.18 The Structure of Simple Search Interface ......................... 47
4.19 Simple Search Interface in the Web site of the University of Victoria .... 48
4.20 Structured and Unstructured Information ............................ 50
ACKNOWLEDGMENTS

I would like to express my deepest thanks and gratitude for my supervisor, Dr. Daniel M. German, for his intellectual and financial support as well as his guidance in writing this thesis. I would also like to thank Dr. Daniel Hoffman for his valuable co-operation. I thank all the other members of my supervisory committee, and all the other faculty and staff, especially David Manning.

I thank the other members of the View lab: Stephen, Humberto, Wendy, Abram and Tracy. I also thank all the members of the MTV lab, especially Durga and Belaid. I thank all my teachers and friends throughout the world. I thank all the Bangladeshis in Victoria, especially Newaz bhai, Suraiya Madam, Zaman bhai, Aumi bhai, Sabbir bhai, and Kajal bhai. I thank Dr. Ekram and Dr. Akbar. I thank Dr. Watheq El-Kharashi for his valuable advice. I also thank my roommates Aron and Keegan.

Special thanks to Wendy and Keegan for proof reading this thesis.

I gratefully acknowledge the fellowship award and other financial assistance that I have received from the University of Victoria during my graduate studies.

I express my deepest gratitude, respect and love for my parents who provide me with great encouragement. I express my deepest love for my brothers, sister, nephew and nieces. I also thank all of my other relatives.
1. Introduction

1.1 Design Patterns for Hypermedia

Christopher Alexander says, "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [AIS77]. Patterns are gaining popularity in hypermedia applications as well as software applications. Hypermedia authors are now acquiring experience to build good hypermedia systems and assess their quality. This experience should be compiled and organized in order to be shared with other authors. Hypermedia designers are now focusing on documenting their design experiences as patterns [GC99].

1.1.1 Navigational Design Pattern

Navigational patterns are like Alexandrian patterns in urban architecture. The patterns in urban architecture describe the roads you can follow to reach different homes, the kind of orientation signs you will find, short-cuts, etc. The navigational patterns will provide a road map to explore the information space. The navigational structure of an information space can be improved by using navigational patterns. They help to build usable navigational architecture ensuring information is easy to find and the users will be able to explore the information space with less cognitive overhead [RSL99].

The goal of this thesis is to organize and refine the patterns, and to create a system of patterns for Web navigation to navigate information on the Web doing the following:

- collecting the hypermedia design patterns known to date by tracing different publications;

- creating a catalog of patterns included in the system of patterns for Web navigation by writing them in a uniform notation;

- identifying the relationships between the patterns;
• describing their refinement and evolution;

• providing a component oriented approach for the reusable implementation of those patterns; and

• developing applications by combining the patterns.

1.2 Thesis Organization

Chapter 2 describes the current state of the literature in the area. It describes the patterns, significance of the patterns, their classification, and languages. Chapter 3 describes the system of patterns for Web navigation, their refinement and evolution. Chapter 4 contains a catalog of patterns. It also depicts the relationships between the patterns. Chapter 5 describes the implementation approach. Chapter 6 describes the conclusions. Appendix A contains a list of all hypermedia patterns published in the literature. Appendix B contains the interfaces of all the developed components.
2. Background Study

2.1 Motivation

The idea of patterns itself originated in the urban architecture during 1970 and Alexander was talking about patterns in buildings and towns. But what he says is true about object-oriented design patterns too [GHJV96]. In object oriented design, the solutions are expressed in terms of objects and interfaces instead of walls and doors, but at the core both kinds of patterns describe a solution to a problem in a context. A design pattern attempts to collect experience from the expert in order to pass on to other experts or novices in the field, hence avoiding re-invention by others [AG03]. By explaining and evaluating recurrent problems and proven solutions, they can record design experience. They describe those problems and the core of their solutions in such a way that it can be used with different instantiations of the same problem.

Design patterns complement design methods as they show solutions that go beyond the use of primitives of a method [RSL99]. A naive object-oriented designer may tend to closely follow the main concepts of the object paradigm, encapsulating structure and algorithms in the same object. But complex problems require more advanced solutions, like the ones that appear in patterns such as Bridge, Strategy, or State. [GHJV96]. For complex problems, the representation, the algorithms or the state of the object are further objectified and defined outside the object in the context of a separate hierarchy. Patterns act as micro-architecture in a system and can be defined at different abstraction levels and they improve system modularity and ease of extension. Patterns enable widespread reuse of software architectures and reusing architectures is more valuable than reusing algorithms or data structures [RSG97].

The documentation of a pattern contains which kinds of problems originate this micro-architecture, when it should be used and what is the trade-off when using it. As explained in [GHJV96], patterns are neither original nor new solutions to software problems; instead they are well proven and used strategies employed in successful developments. As
such, patterns are not invented but discovered or mined. These patterns may appear to be
obvious to expert designers, but they will constitute a valuable source of design expertise
to less experienced designers. Patterns explicitly capture the knowledge used by the ex-
pert designer and can thus provide training for a new designer. Patterns are already tested
and proven techniques, and the development of the patterns implies that several different
solutions have been tried and they avoid the expense of trying new solutions for a partic-
ular problem. The main goal of design patterns is reuse of design experience. The design
knowledge of experienced designers is documented as a pattern in order to pass it on to oth-
ers. Hence design patterns improve communication among the designers and developers.
They increase the quality of an application, minimize the amount of new development, and
hence they reduce the cost of design and development. Patterns also describe the trade-off
and design alternatives. One caveat to this is that the reuse provided by the pattern will of-
ten require a degree of adaptation and designers need to be careful that the effort required
to adapt it does not exceed the effort required to develop the solution from scratch [LH99].

An application must evolve over time to adapt to new requirements and with changes
in existing requirements. The design of the application should be robust to such changes.
Sometimes it may require redesign. Redesign affects many parts of the application and
unanticipated changes are invariably expensive. This redesign can be avoided by using
design patterns. Design patterns ensure that a system can change in specific ways. Each
pattern lets some aspect of system structure vary independently of other aspects, thereby
making a system more robust to a particular kind of change [GHJV96].

2.1.1 Design Pattern Language

It is possible to ensure and improve the quality of an application by using patterns. To
be effective and to maximize the usefulness of those patterns, they should be described in
a consistent format. The patterns can be described using several notations, the most widely
used ones are: original Alexandrian notation and Gamma notation (Gang-of-Four format).

According to the Alexandrian notation, a pattern has four essential elements, as stated
in [GHJV96]:
Pattern name: The pattern name is used to describe a design problem, its solutions, and consequences in a word or two. A pattern name should be chosen carefully and should be precise, concise and clearly convey its purpose. Naming a pattern increases design vocabulary. It lets designers a higher level of abstraction. Patterns makes it easier to think about designs and to communicate them and their trade-offs to others. Finding good names is one of the hardest parts in developing a catalog.

Problem: It describes the problem, its context, and when to apply the pattern. It might describe specific design problems such as how to represent algorithms as objects and class or object structures that are symptomatic of an inflexible design. Sometimes the problem will include a list of conditions that must be met before it makes sense to apply the pattern.

Solution: The solution describes the elements that make up the design; their relationships, responsibilities, and collaborations.

Consequences: The consequences are the results and trade-offs of applying the pattern and they are critical for evaluating design alternatives and for understanding the costs and benefits of applying the pattern.

According to Gamma et al. [GHJV96], a pattern should be described using the following template:

Pattern Name: The pattern's name conveys the essence of the pattern succinctly. A good name is vital, because it will become part of the design vocabulary.

Intent: A short statement that answers the questions- what does the design pattern do? what is its rational and intent? what particular design issue or problem does it address?

Motivation: A scenario that illustrates a design problem and how the class and object structures in the pattern solve the problem. The scenario will help the user understand the more abstract description of the pattern that follows.
Applicability: What are the situations in which the design pattern can be applied? What are examples of poor designs that the pattern can address? How can you recognize these situations?

Structure: It shows a graphical representation of the classes in the pattern.

Participants: It describes the classes and/or objects participating in the design pattern and their responsibilities.

Collaboration: A description of how the participants are related or collaborate to carry out their responsibilities.

Consequences: It is a description of the benefits and the trade offs that result from applying the pattern.

Known Uses: A list of well-known applications of the pattern.

2.1.2 Discovering Patterns

Patterns record design experiences of the expert designer. They are the already tested and proven techniques and they are discovered, not invented. In discovering new patterns, one should look at as many existing systems as possible and identify the problems that occur in different applications and the solutions provided by those applications. Evaluate the solutions of a particular problem provided by different applications. The problem and the best solution can be recorded as a pattern. While recording a pattern, one must focus on the structure of the pattern. A pattern shows the structured exposition of a solution to a problem in a context and the more information the pattern of interest has, the more important the structure becomes. Consistent structure provides uniformity to patterns and allows people to compare them easily. One can follow either the structure of the original Alexandrian patterns or the structure of Gamma et al. [GHJV96]. In discovering patterns, one must be careful about possible duplicacy of patterns. A particular pattern must be distinct and strong enough to stand by itself. If two patterns solve the same or similar problems, they should be merged into one pattern.
2.1.3 Selecting a Pattern

The number of design patterns is increasing. There is now a big collection of patterns and the question is how to select a pattern for a particular problem at hand. There are several different approaches to select the right pattern for the problem. The following approaches are described in [GHJV96]:

1. Problem Solving: The most important point in selecting a pattern is to consider how the pattern solves a given problem. Design patterns help in finding appropriate objects, determining object granularity, specifying object interfaces, and several other ways in which design patterns solve design problems. These help in selecting the right pattern for a particular problem in consideration.

2. Intent: The intent section of a pattern describes what particular design issue or problem it addresses. In selecting a pattern for a particular problem, one should scan the intent section of the patterns very carefully.

3. Relationships among the patterns: The relationships of the patterns guide in selecting the right pattern. One should look all the related patterns of the pattern to be selected.

4. Patterns of like purpose: The patterns can be grouped based on their purpose like presentational, structural and support [D. 97] or based on their practicality like usability patterns, navigational patterns, patterns for e-commerce etc. In selecting the right pattern, one should look at all the patterns in a group.

5. Causes of redesign: An application should evolve based on new requirements and changes to existing requirements and the design of the application should be robust to such changes. In most of the cases, hypermedia applications are built from scratch without following a design methodology; they lack a proper structure and so new requirements and changes to existing requirements cause the developers to redesign the application. Patterns can help to avoid this redesign by letting some aspect in the architecture of the system that can vary independently of the other aspects making the system more robust to a particular kind of change. The hypermedia designer
should use and choose the patterns considering the possible evolution of the system over time.

2.1.4 Frameworks and Patterns

A framework is a set of cooperating classes, some of which may be abstract, that make up a reusable design for a specific class of software. Frameworks are not necessarily domain specific, but they are usually concept specific [Szy99]. The framework dictates the architecture of an application and also defines its overall structure and partitioning into classes and objects. It depicts how the classes and objects collaborate. A framework captures the common design decisions of its application domain and emphasizes design reuse over code reuse [GHJV96].

The main contribution of a framework to an application is the architecture of the application; the main focus of the designer of a framework is to design the framework to be as flexible and extensible as possible so that one architecture will work for many applications in the domain. Applications are dependent on the framework for their design and they must evolve with that of the framework.

A framework that uses design patterns is far more likely to achieve high levels of design and code reuse than one without design patterns [GHJV96]. A framework usually incorporates several design patterns and those patterns make the architecture of the framework suitable for many different applications without redesign. It will provide more benefits if the framework is documented with the design patterns it uses, because people who know the patterns gain insight into the framework faster. Frameworks often pose a steep learning curve and design patterns can make it less steep by making key elements of the framework's design more explicit. However, behind the similarities between the patterns and frameworks, they are different in the following three ways [GHJV96]:

1. **Design Patterns are more abstract than frameworks.** Frameworks can be embodied in code. They can be written in programming languages, executed and reused directly. But in the case of design patterns, only examples can be embodied in code.
2. Design patterns are smaller architectural elements than frameworks. A mature framework usually contains a number of design patterns, but the reverse is not true.

3. Design patterns are less specialized than frameworks. Frameworks always have an application domain, but the design patterns can be used in any kind of application.

2.1.5 Using a Pattern

After selection of a pattern for an application, the question is how to use the pattern in the application. Following are the different steps in accordance with steps described in [GHJV96] to be followed in using a pattern effectively:

1. Applicability and Consequences: The applicability and consequences section of the pattern show where the pattern can be used and what are the trade-offs of using that pattern. Before using the pattern, one should pay particular attention to the applicability and consequences sections.

2. Structure, Participants, and Collaborations: One can identify the classes/objects of the pattern by looking at the structure and participants of the pattern, and can depict the relationships of those classes/objects by looking at the collaborations.

3. Implementation Guidelines: A system of patterns will provide some guidelines for the implementation of the patterns and one can use those guidelines to instantiate the pattern in a specific application.

4. Define classes: After identification of the possible classes and their relationships, one should define the classes, their interfaces, and implement the operations of those classes.

5. Integration with others: A pattern will be a part of the whole application and the whole application may contain a number of patterns. One has to be careful how the code of the pattern will match with the other patterns and within the whole applications.
2.2 Hypermedia

The World Wide Web is the most common type of hypermedia application and it demonstrates the most important characteristics of a hypermedia application that is allowing the users to navigate through an information space using associative linking which leads to ideas such as non-linearity and inter-activity.

The primary characteristics of hypermedia are:

**Non-linearity:** The traditional media is linear, where there is a single path through the material. Hypermedia applications provide non-linearity, i.e., they provide multiple possible paths through the information. The user can interactively browse through the information following any of the possibly very large number of paths instead of a predetermined path.

**Various Types of media:** Hypermedia applications support the use of multiple types of media and access mechanisms. We can convey information by using media types which are best suited to the particular information[HL99].

Among the secondary characteristics, the integration of media into a cohesive whole is the most important one. As explained in [HL99], ”a major flaw in many existing hypermedia applications is that much of the information, especially non-textual media, has been patch-worked together. It is not sufficient to simply create non-linear structures– they need to be created in a way which is meaningful to the user. Developers also need to consider the mechanisms which will be used in developing and maintaining non-linear structures”.

By Lowe and Hall [HL99], hypermedia is defined as: “An application which uses associative relationships among information contained within multiple media data for the purpose of facilitating access to, and manipulation of, the information encapsulated by the data. The key elements are access to and manipulation of information and associative relationships amongst the information”.

Applications in hypermedia domain are organized in two categories [GRS97]: **Hypermedia systems** and **Hypermedia applications. Hypermedia systems** are mostly the tools and infrastructure used by developers in creating or supporting applications. In some cases,
those systems are also used as a delivery and presentation system for the application. Different types of hypermedia systems are the development systems - such as authoring tools, media editing packages, database systems, etc., the delivery and support systems - such as Web servers, CGI scripts, database engines, etc., the presentation systems - such as Web browsers. Hypermedia applications are the end result of the hypermedia development process, the product which has been developed for end-users [LH99]. For example, Web sites are hypermedia applications. Developers create those Web sites using a variety of development systems and then supported by the infrastructure of the World Wide Web and by the delivery and support systems, and accessed by the presentation systems. Again, hypermedia applications can be sub-divided by two design concerns: Navigation Design and Interface Design.

2.2.1 Web Information Systems

Web Information Systems (WIS) are the information systems that are constructed using Web Technology. The main design concerns for an Web Information Systems are [RSL99]:

1. Architecture. Designing the architecture is the most challenging issue in developing Web Information Systems. The architecture should focus on good separation of concerns, support evolution, and the kind of interface such as communication, formatting etc. with legacy applications such as databases.

2. Information Navigation It is a prime concern of a Web Information Systems to organize its information in a way such that the user can navigate the information space easily. It should be ensured that the information is easy to find and the user will not experience cognitive overhead in exploring the information space.

3. Information presentation Information presentation is one of the most important issues in Web Information Systems. It focuses on how users will experience the information being presented while exploring the information space. The information
presentation should support the information interpretation. Users will not be able to use the information if they can't sense it.

Another important issue is to present both navigation and application functionality in an organized, coherent way. We can use different hypermedia design methods such as OOHDM [SRB96, SR98], RMM [ISB95], W3DT [TL97] to model different aspects of a WIS. These methods provide a clear separation of concerns among architectural or conceptual design, navigation design, user interface design, and implementation [RSL99].

The most common types of Web Information Systems are the E-commerce applications, Educational Multimedia applications. Navigational patterns appear recurrently in Web Information Systems and they can assist in addressing the above mentioned design issues.

2.2.2 Classification of Hypermedia Design Patterns

In order to be effective, patterns should be organized into a catalog and its patterns should be classified. With a good classification, designers will be able to know if there is a pattern for their problem and if there is, they can find it easily. It will also promote refinement and potential unification of patterns. As explained in [GC00], patterns can be classified in a seemingly infinite number of ways. For instance, they can be classified by the problem they try to solve, by the stage of the design process in which they are most likely to be used, or by the application domain they can be used in, etc. A successful taxonomy should be simple and have as few classifications as possible.

Rossi et al. [RSG97] classified the patterns based on the different dimensions of hypermedia with:

Hypermedia Systems: Patterns of this category can be used to develop hypermedia systems. Hypermedia system developers are the audience of this category. Examples of pattern in this category are the Navigation Observer, Navigation Strategy.

Hypermedia Applications: Hypermedia application designers use these patterns regardless of the system they are using for the implementation. They further sub-divide
the patterns in this category based on two design concerns: Navigation Design and Interface Design.

**Navigation Design:** Patterns of this category help in organizing the navigational structure of the application. Examples of pattern in this category are Active Reference, Set Based Navigation, Landmark etc.

**Interface Design:** Patterns of this category organize the information in such a way that interface elements interact for an effective presentation of data. Examples of pattern in this category are Information On Demand, Information-interaction Coupling.

![Hypermedia Pattern Space](image)

**Figure 2.1: Hypermedia Pattern Space**

German *et al.* [D. 97] classified the patterns based on the objective of the pattern: presentational, structural, and support.

Nanard *et al.* [NN99] classified the patterns based on 3 dimensional space: hypermedia system, hypermedia application, and hypermedia design and development.

In [ACM03], the patterns have been classified into three categories:

1. Patterns concerning interface and lay-out aspects.
2. Patterns involving structure of information and navigation dynamics.
3. Patterns more content oriented, the patterns that pay more attention to the message strategy, in general, to the content of the application.
**Interface/Layout patterns** These patterns deal with the physical arrangement of the objects on the screen. Examples of this category are: active reference, behavioral grouping, behavioral anticipation.

**Structure/Navigation patterns** These patterns concern the structure of the information space and navigation through the information space. They are structure oriented, analyzing application components and their relationships, as well they can cope with navigational problems. Examples of this category are: Collection Center, Guided Tour, Landmark.

**Content Oriented Patterns** These patterns deal with the overall organization of information in the application, paying more attention to the communicative features of the hypermedia. Examples of this pattern are: advising, analyse organize synthesize, complex entity.

2.2.3 **Functional Characteristics of Hypermedia Application and Design Patterns**

The functional characteristics of a hypermedia application affect the functionalities provided by that application and those are directly visible to any user of the application. The following description explains the most important functional characteristics of hypermedia applications and the patterns that support those characteristics.

**Navigability:** The most important characteristic of any hypermedia applications is to support navigation around its information space. Hypermedia applications provide association-based navigation. Navigation mechanisms must be able to be applied not only based on text, but also based on graphics, images, animations [LH99]. The navigational structure of any hypermedia application can be improved by using navigational design patterns. Some of the supporting patterns for these characteristics are Set Based Navigation, Landmark, Active Reference, News, Shopping Basket, Simple Search Interface.

**Information maps and overviews:** These characteristics of hypermedia applications assist the users in understanding the overall structure of the information space and also
showing the user's location within this space [LH99]. The patterns supporting this characteristics are Active Reference, Guided Tour, Navigational Context.

**Information trails:** This characteristic allows users to remember where the user has been, and allow the user to return to those places [LH99]. The patterns supporting this characteristic are Active Reference, History, Shopping basket.

**Searching and Indexing:** Hypermedia applications support association-based navigation, but it is not sufficient. They must support searching and indexing in order to effectively achieve the goal of identifying information. They should support sophisticated searching and indexing mechanisms to handle contextual searches and non-textual media [LH99]. The patterns supporting this characteristic are Simple Search Interface, Selectable Search Space, Index Navigation.

**Information contextualisation:** The context in which the information being presented affects the user's ability to identify what is appropriate and what is not [LH99]. This characteristic is supported by Node as a Single Unit, Clustering, Collector, Collection Center.

**Presentation:** Information presentation is critical in identifying appropriate information, how to control and use it in an effective fashion. To use the information properly, users should be able to sense it and this depends on the presentation [LH99]. The patterns supporting this characteristic are Behavioral Grouping, Behavioral Anticipation, Information on Demand, Information-Interaction Coupling.
3. System of Patterns

3.1 System of Patterns

According to Buschmann et al., a pattern system for software architecture is a collection of patterns, together with guidelines for their implementation, combination, and practical use in software development [BMR++96].

Patterns do not exist in isolation. There are many interdependencies among them. A plain catalog-like list of all patterns does not reflect these relationships. A pattern system ties individual patterns together. It describes how its constituent patterns are connected with other patterns in the system, how they complement each other, how these patterns can be implemented, and how application development with patterns is supported [BMR++96].

A pattern system is a powerful vehicle for expressing and constructing the architecture of an application. The main objective of a pattern system is to support the development of a high quality application, fulfilling both functional and non-functional requirements. To achieve this objective, a pattern system must meet the following requirements [BMR++96]:

- It should comprise a sufficient base of patterns. It needs patterns that support specification of the basic architecture of a system, patterns that help with refining this basic architecture, and patterns that help with implementing the architecture in a specific programming language.

- It should describe all its patterns uniformly. The form of description must capture both the essence of a pattern and a precise depiction of its details. The form must further support the comparison of a pattern with other patterns.

- It should expose the various relationships between patterns. The pattern system must identify which other patterns a pattern refines, which other patterns it exposes, with which patterns it can be combined, and what alternatives are available.

- It should organize its constituent patterns. Users should be able to find a pattern
quickly that helps them solve their concrete design problem, and they should be able to explore alternative solutions that are addressed by different patterns.

- It should support the construction of applications. A pattern system should show how to apply and implement its constituent patterns.

- It should support its own evolution. With evolving technology, a pattern system will evolve as well. Existing patterns will change, their description will improve, new or missing patterns will be added and existing ones may even 'die'.

3.2 A System of Patterns for Web Navigation

Web Navigation: Navigation is one of the most essential ingredients for any successful Web site. Visitors must be able to find the information they are looking for. If its navigation is confusing or convoluted, the usability of the Web site will be diminished. This might lead to a major usability problem: "lost in hyperspace", which is disorientation or loss of context that arises from an unfamiliar conceptual structure and organization of the site. A good navigation design:

- makes information easy to find or explore,

- helps readers find the desired content quickly,

- provides multiple navigation paths for different readers,

- allows users to know the current position in navigation,

- is consistent, and

- provides feedback and support in navigation.

The system of patterns for Web Navigation contains a collection of Navigational patterns to design the navigational structure of Web applications especially Web Information systems. It includes the most frequently used navigational design patterns: Set Based Navigation [ACM03, G. 99], Landmark [RSL99], Active Reference [ACM03, GRS97, LRS98b,
RSG97], Visited Objects, Guided Tour [ACM03], News [ACM03, G. 99, LRS98a], Selectable Search spaces [ACM03], Simple Search Interface [ACM03], and Shopping Basket [ACM03, G. 99]. There is a behavioral relationship among the patterns such that they can be combined into a framework. Here the Active Reference will provide a road map of the site. The Selectable Search Space, Simple Search Interface and Set Based Navigation will help to find the desired information quickly. The framework can provide multiple navigation paths using the Guided Tour, Active Reference, and Visited Objects pattern. The Active Reference pattern lets the users know the current position in navigation. The Set Based Navigation pattern provides consistency and support in information navigation by organizing the information space into meaningful groups. All these patterns combinely fulfill all the basic requirements of navigation design. It can be extended by using the other navigational patterns. The Dreamweaver templates [Dre04] are very good for general purpose web application development. But the patterns included in this pattern system are more suitable for eCommerce applications or web information systems. In fact, some of the templates have instantiated the Landmark and Opportunistic Linking pattern.

3.2.1 Base Patterns

The number of design pattern is increasing. It is now becoming difficult to keep track of them and to determine whether there is a pattern that solves a particular problem or finding the pattern that solves the problem [GC00]. All these patterns should be collected to form a catalog. Appendix A contains a list of 96 patterns. This list includes only the hypermedia design patterns. The Hillside.net pattern library is a good source for information about all aspects of software patterns and pattern languages [Hil04][Pat04]. All the patterns are collected by tracing different publications. Most of the patterns are listed in [GC00] and [ACM03]. The list is ordered by name and includes a brief description and the publication in which it is appeared. The list arguably forms a sufficient base of patterns to create a system of patterns like system of patterns for Web navigation.
3.2.2 Uniform Description

In this thesis, it is decided to describe the patterns of the system in Gamma notation (GoF format). This notation provides good separation of concern, lends a uniform structure to the information contained in the pattern, and makes them easier to learn, compare, and use. It represents the pattern using a graphical structure which increases the understandability. From the structure, participants and collaborations, one can identify the classes/objects and their relationships, which will facilitate the implementation or use of the pattern in practical applications. This notation explains the situations where the pattern can be used. Hence, Software or Hypermedia design patterns should be described using Gamma notation.

The patterns included in the system of patterns will be described later in Chapter 4.

3.2.3 Relationships among the patterns

There is a behavioral relationships among the patterns and that relationship is depicted later in chapter 4.

3.2.4 Support for the construction of hypermedia systems

All the patterns of this pattern system are implemented. Chapter 5 explains an approach for the reusable implementation of the patterns and it is described how hypermedia applications can be developed by integrating the patterns.

3.2.5 The refinement of the hypermedia design patterns

As explained in [GC00], the existing hypermedia patterns can be refined in the following ways: (1) Unifying, (2) Renaming and (3) Rewriting.

Unifying: The patterns can be unified in the following ways [GC00]:

- Some patterns are the same under different names: The intent of Bread Crumbs pattern is: the users need to know where they are in hierarchical structure
[Mar03]. Again the pattern Active Reference provides a perceivable and permanent reference of current position in navigation. These two patterns are same under the different names.

The pattern Split Navigation [Mar03] is the same as Hierarchical Structure Through Navigation Side bars [st99].

- Some patterns are special cases of another. For example, Shopping Basket describes how to keep track of user selections during navigation and making these selections persistent to process them, and the pattern Visited Objects is to keep track of the records of all the items visited. Shopping Basket is a special case of Visited Objects.

Double Tab describes the users need to navigate a hierarchical structure [Mar03]. The instance of the pattern is shown in the figure 3.1.

![Amazon.com screenshot](image)

**Figure 3.1: Double Tab in www.amazon.com**

This pattern is a special case of Landmark pattern which provides direct access to different sub-systems of the application. One will get the same effect applying the Landmark pattern in two levels.

One can also get the same effect by combining the Set Based Navigation pattern with the Landmark pattern. The first level which shows the different subsystems will be represented by the Landmark and the second level which shows the different sets in reality of a particular sub-system can be represented by the Set Based Navigation pattern. In fact, the second level may not contain the sets
of a sub-system of the first level every time. It also may contain different types of distinct sub-system like the first level from a different perspective instead. In this case, one can achieve the same effect by applying the Landmark pattern in multiple levels.

![Double Tab in www.apple.com](image)

**Figure 3.2: Double Tab in www.apple.com**

The Double Tab in figure 3.2 can be represented by applying Landmark in two levels.

- Some patterns are extended versions of other patterns. For example, the pattern Simple Search Interface is an extended version of Selectable Search Spaces.

- Some patterns provide insight that can enhance similar patterns. Some patterns provide different views of the same problem and they should be unified into a single super pattern including all the insights provided by the patterns individually. For example, the patterns Simple Search Interface and Selectable Search Space can be unified into a single pattern, Search which will allow to search both structured and unstructured information space. The unified pattern, Search will allow to choose a particular category of items in which the search will be performed in case of structured information space and all items in case of unstructured information space. Figure 3.3 shows an instance of this unified pattern.

**Renaming:** A pattern name should be precise, concise, carefully chosen and clearly convey its purpose. Some patterns should be renamed in order to express their purposes more clearly. The pattern News shows the new releases. WIS usually contains different types of item and News shows the new releases of a particular type of item at a time. So it can be renamed as Selectable News in order to stress the fact that it will
allow the users to select a particular category and a time span and then navigate the new releases of that category in that time span.

**Rewriting:** Some patterns should be rewritten entirely in order to express their purposes in a more understandable way. For example, the description of *Set Based Navigation* [RSL99] in its original publication states:

The Problem

WISs usually involve dealing with collections of objects (e.g., Books, CDs etc.). These collections may be explored in different ways, according to the task the user is performing. In an electronic shop we may want to explore all Rolling Stones CDs. We could also want to navigate to all CDs in which Mick Jagger participated (including Rolling Stones ones). Notice that the same situation appears in different domains, such as institutional sites. We should help the user by identifying meaningful sets for the targeted tasks, and ease the traversal of these sets easily by providing fast access to different members of the set.

Solution

The usual naive strategy followed by designers consists in providing an index to set members, users must then go back to the index to navigate to the next member of the set. As a consequence, the usual simplicity of the hypertext navigation metaphor is lost. Even if we closely follow a systematic design method, like OOHDM or RMM, we may obtain the same solution. The reason is that there are no additional semantic relationships among itself, so one may
not realize that they should be linked explicitly (at least in the node-and-link hypertext model of the WWW).

Beginner designers tend to worry about how the reader reaches a given node, leading to an index. But these designers fail to consider what the reader should do next (once the node has been reached) typically, the only option left is to back up to the index. This leads to a purely hierarchical organization of the information, failing to exploit the richness of full hypertext.

The expert solution is to consider set-based navigation as a first-class navigation strategy. We group nodes in meaningful sets and provide inter and intra-set navigation facilities, such as indexes and links for letting the user navigate to the next and previous elements of the current one in his traversal. Notice that a node may belong to more than one set, so it is often useful to allow the user to switch sets. We call such sets Navigational contexts in OOHDM; they are also referred to as Collections in HDM2.

Navigation inside contexts complements conventional semantic links, such as, for example, those connecting a node about a CD with the node about artist himself or CDs to comments about them. In other words, the reader can browse through the set or leave it to explore other nodes (or eventually other sets).

**Known uses**

Sets appear in almost all web-sites; for example; every index (menu) automatically designs a set, the nodes it points to. This is also true of every Web site where we can perform searches. In spite of this, set-based navigation itself is not as widely provided. As an exception, many sites provide guided tours through collections; see for example http://www.nga.gov.

**Related Patterns**

Nodes in Context is a pattern that complements Set-based Navigation showing how to solve the problem that appears when a Node appears in more than one Set (Context). Shopping basket is a special kind of set that is built dynamically,
as the result of users selections during navigation [LRS98a].

This description is rewritten in Gamma notation using UWE in Chapter 4.

### 3.3 Evolution

With the development of new technologies over time, the existing technologies are enhanced or become outdated. Therefore, new patterns will emerge, and existing patterns may become outdated. The new patterns should be integrated into the pattern system to keep it up-to-date and outdated patterns should be removed if they are no longer used.

Within the context of pattern system evolution, the following issues must be considered [BMR+96]:

- **The evolution of pattern descriptions:** It is important to improve and stabilize the description of every pattern in a pattern system continuously for the system to remain useful. The number of hypermedia design patterns or navigational patterns are increasing and different authors are writing them in different notations. To be more effective, all the patterns should be written in a standard notation.

  The patterns of this system are described in Gamma notation which is the most accepted notation. The description of those patterns can be reviewed by using the experience gained from its application. Such review may identify the additional benefits, further potential liabilities and limitations provided by the patterns. The description of the patterns can be extended by extending the list of known uses whenever a pattern is applied successfully. The more known uses listed, the greater is the chance that users will identify a similar design situation.

- **Pattern mining:** It is useful to mine a new pattern for a particular problem, if there is no suitable pattern for that problem. The number of hypermedia design patterns or navigational patterns is increasing and our pattern system for Web navigation can be extended by mining more new patterns.

- **The integration of new patterns:** A pattern system can be extended by integrating
new patterns or other existing patterns into it. It can be done by identifying the relationships of the pattern to be included with the already included patterns. For example, the pattern system for Web navigation can be extended by including the pattern *Hierarchical structure through navigation side bar.* One can create a hierarchical structure of the sets created by *Set Based Navigation* by using the pattern *Hierarchical structure through navigation side bar.*

- **Removing outdated patterns:** With evolving technology, patterns can become outdated for the following reasons:

  1. *Disappearance of the problem:* A problem that in the past had to be explicitly addressed might now be handled by the programming languages or system environments in use.

  2. *Better alternatives:* A new solution to a particular design problem might become available which is preferable to existing patterns that address the same problem.

  3. *Technology evolution:* The evolution of programming languages and styles or a change in the kinds of system that are developed can cause existing patterns to become outdated.
4. Catalog of Patterns

4.1 Introduction

This chapter includes a catalog of navigational design patterns described in Gamma notation. It includes the following frequently used navigational design patterns:

- Set Based Navigation: Organize the information in sets of related information items. Provide inter- and intra-set navigation capabilities [RSL99, ACM03].

- Landmark: Provide direct access to critical sub-systems in Web Information Systems [RSL99].

- Active Reference: Provide a perceivable and permanent reference about the current status of the navigation, providing an orientation tool and an easy way to navigate to a set of related nodes, at the same or higher hierarchical level [ACM03, GRS97, LRS98b, RSG97].

- Visited Objects: Keep a record of items that the user has visited in navigating different types of items in a Web Information Systems.

- News: Allow easy access to new information items as the Web Information System grows [ACM03, G. 99, LRS98a].

- Guided Tour: Provide an “easy-to-use” access to a small group of objects, assuming that user has no reason or is not able to select one of them [ACM03].

- Selectable Search Space: Specify a category within which the search should be made or restricted to [ACM03].

- Simple Search Interface: Provide users with powerful, yet simple, search mechanisms [ACM03].
- Shopping Basket: It keeps track of items that the user has selected making these selections persistent and ready to be processed when the user wishes. [G. 99]

UML-based Web Engineering (UWE) is used to describe the structure of each pattern in addition to standard UML, which represents the pattern graphically. The modeling elements of UWE, which are used in this catalog to describe the structure of the patterns, are described in the following section.

4.1.1 The UWE Methodology

According to Koch et al. [KK02], UML is not powerful enough to cover all the requirements that arise when modeling Web applications. For most of these requirements, it is possible to use the notation and diagrammatic techniques provided by UML. To cover the special aspects of Web application design, Koch et al. have defined UWE-UML-based Web Engineering.

As explained in [KK02], UML-based Web Engineering (UWE) is a development process for Web applications with a focus on systematic design, personalization and semi-automatic generation. UWE describes a systematic design methodology using exclusively UML techniques, the UML notation and the UML extension mechanisms. UWE prescribes how to build navigation and presentation models for Web applications defining special UML stereotyped modeling elements and tagged values.

4.1.1.1 Modeling Elements

The following modeling elements are defined in UWE [KK02, HK01, KKH01, GHJV96]:

- Navigational class: A navigation class models a class whose instances are visited by the user during navigation, and it is depicted by the diagram in figure 4.1. This modeling element has been used to represent an information space to be explored in describing the patterns in this thesis. This information space will contain different types of items to navigate. This class can be instantiated using the attributes, which are representing a particular type of collection of items.
Figure 4.1: Navigational Class

- Instantiation: A dashed arrowhead line indicates a class that instantiates objects of another class. The arrow points to the class of the instantiated objects.

```
Instantiator ----> Instantiatee
```

- Association: Associations in the navigation space model are interpreted as representing direct navigability from the source navigation class to the target navigation class. It is represented by a solid arrowhead line.

- Presentational class: A presentation class is a structural unit which allows to partition a user interface view into groups of user interface elements which is represented by the diagram depicted in figure 4.2. Instances of a presentation class are containers which comprise modeling elements like texts, images, anchors, video sequences, audio sequences etc. In this thesis, its instances are the actual pages or part of the pages created by different patterns to be shown in browser. The exact content of those pages, in other words, instances of this class is implementation and application dependent.

1. Text: A text is a sequence of characters.

2. Anchor: An anchor is a clickable piece of text which is the starting point of a navigation, establishing the relationship to other nodes.

3. Button: A button is a clickable area which has an associated action. Examples of actions are playVideo, displayImage, stopAudio, and runApplet.
4. Image, audio, and video: Image, audio and video are multimedia objects. An image can be displayed, audio and video can be started, stopped, rewinded and forwarded.

5. Form: A form is used to request information from the user who supplies information in one or more input fields or selects options from a browser or checkbox.

6. Collection and anchored collection: Collection and anchored collection are model elements, such as lists of text elements and lists of anchors introduced to provide a convenient representation of composites.

4.2 Relationships among the patterns

The patterns in the system were chosen in a way that they work together. Figure 4.3 depicts the relationships among the patterns and how the patterns will interact with one another when they are used together. The diagram shows the flow of information from one pattern to another. Set-Based Navigation plays a central part in this system, as many other patterns rely on it to display specific information to the user.
4.3 Set Based Navigation

**Name:** Set Based Navigation.

**Intent:** Organize the information in sets of related information items. Provide inter- and intra-set navigation capabilities [RSL99].

**Motivation:** Different Web Information Systems involve dealing with collections of objects. Those collections can be explored in many ways based on the tasks users will perform. For example, in a bookstore, users may want to explore all “Database” books, “Software engineering” books, etc. In a movie store, users may want to explore all “Action” movies, “Comedy” movies, “Horror” movies etc. Again, users may want to explore the “Classical” movies, “General” movies etc. of “Action” movies. Usually it is done by providing an index to each element and users must then go back to the index to navigate to the next element. But this backtracking will incur significant navigational overhead and a list of items will not provide any guidance in information navigation. It is possible to help the user in exploring the collections of objects by identifying meaningful sets and ease the traversal of those sets by providing fast access to different members of those sets.
**Applicability:** This pattern is applicable in exploring different collection of items. Most Web Information Systems organize their information into collections of objects. These collections can be navigated using set-based navigation.

**Structure:** Figure 4.4 shows the structure of the pattern.

![Diagram of the structure of set-based navigation](image)

**Figure 4.4: The Structure of Set Based Navigation**

**Participants:** The participants of this pattern are:

1. **Navigation class:** It represents the collection of items to be explored like Book, Movie, DVD.

2. **SetBased class:** It identifies the sets of objects to navigate and provides access to the members of those sets. For instance, it may create a table of contents of sets, a table of contents of the members of a particular set, and provides access to previous and next element of the current element of a particular set. In other words, it does the set based navigation on navigational classes like Bookstore, Moviestore. It creates the presentation classes to be shown on View.

3. **View:** It is the user interface to show the different components like table of contents of the created sets, content of a member of a particular set etc. which are represented by the presentation class.

**Collaborations:**

1. **Navigation class:** It provides the collection of items to SetBased class.
2. *SetBased class*: It takes a navigational class and sends the Presentation classes to View.

3. *View*: It receives the Presentation classes generated by the *SetBased class*.

**Consequences:** Users can navigate through collection of items with less cognitive overhead.

**Known uses:** Almost any hypermedia application uses this pattern to show sets of related objects. For example, Figure 4.5 shows an example of the pattern, where it is used to display a set of books at www.chapters.ca. This pattern is also used in www.amazon.com to navigate different collections of items like book, music, DVD.

![Figure 4.5: Set Based Navigation in http://www.chapters.indigo.ca](image)

**Related Patterns:** This pattern can be used in combination with *Active Reference* which will show the different positions in intra-set navigation of *Set Based Navigation*.

**Other Related Patterns:** The pattern *Nodes in Context* [ACM03] complements *Set Based Navigation*. It shows how to solve the problem that appears when a node appears in more than one set [RSL99].

### 4.4 Landmark

**Name:** Landmark
**Intent:** Provide direct access to critical sub-systems in the Web Information Systems [RSL99].

**Motivation:** A large Web site can be divided into a number of sections. These sections may be presenting different types of items (e.g., book, movie) or important pages (e.g., home, help). Each section is a landmark and a navigational schema like a menu, can be built with all the landmarks. This schema is visible in the home page of Web site and all the other important pages. Users can move from one section to another section using a single click.

**Applicability:** This pattern is applicable in large Web site that is divided into a number of sections.

**Structure:** Figure 4.6 shows the structure of the pattern.

![Diagram](image)

**Figure 4.6: The Structure of Landmark**

**Participants:**

1. *List of Hyperlinks:* It contains the addresses of all the sections which will be presented as individual landmark.

2. *Controller:* It builds a menu of the landmarks taking addresses of all the sections and makes the landmarks accessible from every node of that menu using a single click.

3. *View:* It is the user interface to show the menu generated by the Controller.
Collaborations:

1. List of Hyperlinks: It provides the addresses of the sections using parameters to Controller.

2. Controller: It receives the addresses from 'List of Hyperlinks' and provides a menu to View.

3. View: It receives the menu from the Controller.

Consequences: It provides a fast way to move from one section to another section of the Web site.

Known uses: Almost all hypermedia applications have an instance of this pattern. For example, Figure 4.7 shows an instance of the pattern in www.chapters.ca. This pattern is also used in www.nytimes.com to show the different sections of the Web site such as 'International', 'National', 'Job Market', 'Real Estate', 'Weather'.

![Figure 4.7: Landmark in http://www.chapters.ca](image)

Related Pattern: It will be more effective to use this pattern in combination with Set Based Navigation and Active Reference patterns. When a landmark represents a collection of items, then the Set Based Navigation pattern will be used to explore those items and the Active Reference pattern will be used to show the current position in navigation hierarchy.

4.5 Active Reference

Name: Active Reference

Also Known as: Bread Cumbs.
**Intent:** Provide a perceivable and permanent reference about the current status of navigation, providing an orientation tool and an easy way to navigate to a set of related nodes, at the same or higher level in the hierarchy [ACM03].

**Motivation:** The information space can be represented by hierarchical structure. For example, in most Web Information Systems, it is necessary to navigate different types of collections of items. This information space may have a hierarchical structure representing the category of items, sets and subsets of items of that category. The hierarchical structure of the information space can be represented by a tree structure as shown in the figure 4.8. The user should be provided a way to understand the current position in the hierarchy of navigation starting from the root of the hierarchy to leaf. It will help the user understand where he is.

![Diagram of hierarchical structure](image)

**Figure 4.8:** The hierarchical structure in an example information space

**Applicability:** This pattern is applicable where there is a hierarchical structure in the information space.

**Structure:** Figure 4.9 shows the structure of active reference pattern.

**Participants:** The participants of this pattern are:
**Figure 4.9: The Structure of Active Reference**

1. **Controller:** This keeps track of the user's current position in navigating the hierarchy of information.

2. **ActiveReferencer:** This keeps track of the user's navigation path starting from the root of the hierarchy to current position and generates a presentational class showing the whole navigation path. It will also allow the users to backtrack to any of the previous position.

3. **View:** It is the user interface to show the presentation classes.

**Collaborations:**

1. **Controller:** It provides the user's current position to ActiveReferencer.

2. **ActiveReferencer:** It receives the user's current position from the Controller and provides a presentational class showing the whole navigation path containing all the positions to View. It also provides the presentational classes to View based on user's intention in backtracking to the previous positions.

3. **View:** It receives the presentation class from the ActiveReferencer to show in the interface.

**Consequences:** Using this pattern, users will have a better understanding of their position in the navigational space.

**Known Uses:** The Figure 4.10 shows an instance of the pattern in www.chapter.ca to
show the hierarchy of navigation like Books, then Entertainment, then Drama etc. This pattern is also instantiated in www.imdb.com to show the photo galleries of different celebrities.

![Books > Entertainment > Drama](image)

We found 6,068 matching items.

Sort By: Availability | Title | Price | Release Date

Page 1 of 494 1 2 3 4 5 6 7 8 9 10  

1. Wozz Albert
   Author: Percy Mtwa
   Trade Paperback
   ISBN: 0413530000
   Published: September 2003

Figure 4.10: Active Reference in www.chapters.ca

Related Pattern: This pattern can be used in combination with the Set Based Navigation pattern which will be used to create the hierarchical structure of the information space.

Other Related Patterns: The patterns Group Location Awareness [SS99] and Here I am [ACM03] also show similar functionality. It can be used with Hierarchical structure through navigation side bar which will be used to show the navigation hierarchy.

4.6 Visited Objects

Name: Visited Objects.

Intent: To keep a record of items that the user has visited recently.

Motivation: In different Web Information Systems, users may explore different items randomly and users may want to revisit some of these items. It will be time consuming to navigate the site again looking for these items, trying to remember how one
got there in the first place. Visited Objects pattern keeps a list of those items, and provides a link to their corresponding address.

Applicability: This pattern is applicable in different online stores where users are interested in items that are distinguishable from the rest. For example, in an electronic store, each item for sale would be added to the Visited objects when it is browsed.

Structure: Figure 4.11 shows the structure of Visited Objects pattern.

![Diagram of Visited Objects pattern]

Figure 4.11: The Structure of Visited Objects

Participants: The participants of this pattern are:

1. Set Based Navigation Pattern: The different navigational classes like Book, Movie etc. are explored using this pattern.

2. Controller: It processes all visited items. It creates the presentational class to show the list of items on interface and allows to go back to the details of the items.

3. View: It is the user interface to show the pages, which are represented as presentational classes in UWE.

Collaborations:

1. Set Based Navigation pattern: It sends each of the items visited by the user to Controller.
2. **Controller**: It receives the items visited from Set Based Navigation pattern and sends the presentational class to View.

3. **View**: It receives the presentation classes from the Controller to show them in the interface.

**Consequences**: It helps the user in revisiting the items previously visited, without having to retrace the whole information space.

![RECENTLY VIEWED ITEMS](image)

**Figure 4.12**: Visited Objects in www.amazon.com

**Known Uses**: Figure 4.12 shows an instance of the pattern from Amazon which shows all the recently visited items. Ebay also uses this pattern to show previously visited items.

**Related Patterns**: This pattern is similar to the *Shopping Basket* pattern. The difference is that this pattern puts all the items visited in the list and *Shopping Basket* puts only the items selected by the users.

**Other related pattern**: The problem solved by this pattern can also be solved using the *History* pattern. But the *History* pattern keeps the complete navigational history of the user who is browsing the application, not only for different types of items visited.

### 4.7 News

**Name**: News.
**Intent:** Allow easy access to new information items as the Web Information System grows [ACM03].

**Motivation:** One of the most important design issue of WIS is how to maintain its constant growth over time. The navigable space of WIS is not fixed and it evolves as new contents are added over time. For example, an electronic shop may contain books, CDs, and Videos etc. and it must be ensured that users will know about new books, CDs etc. Presenting the new items is a vital selling opportunity for ecommerce applications. Therefore, the users should be provided with instant feedback of any recent changes or additions to the information available.

**Applicability:** This pattern can be used in any WIS that keeps changing. For example, in an electronic store, it can show the newly available items; in an electronic magazine, the latest articles; in a teaching WIS, the latest lectures of a class etc.

**Structure:** Figure 4.13 shows the structure of the News pattern.

![Diagram of the News pattern](image)

**Figure 4.13: The structure of News**

**Participants:** The participants of this pattern are:

1. *Navigation class:* It represents the collection of items to be explored.
2. **News class**: It identifies the new items from the Navigation class.

3. **Set Based Navigation pattern**: It is used to navigate the new items and to create the presentational classes to be shown on interface.

4. **View**: It is the user interface to show the presentational classes.

**Collaborations:**

1. **Navigation class**: It provides the collection of items to be explored to News class.

2. **News class**: It receives all the items from Navigation class and provides the new items to Set Based Navigation pattern.

3. **Set Based Navigation pattern**: It receives all the new items from the News class and sends the presentational classes to View.

4. **View**: It receives the Presentational classes from Set Based Navigation pattern to show them on browser.

**Consequences**: It helps in finding the new items easily.

**Known Uses**: Figure 4.14 shows an instance of the pattern in www.chapters.ca to show the DVDs coming soon, the DVDs within the current month, and the new releases. There is another instance of the pattern in www.imdb.com to show the new pictures of the celebrities.

<table>
<thead>
<tr>
<th>Performing Arts</th>
<th>2. Woza Albert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Television</td>
<td>Author: Percy Mtwa</td>
</tr>
<tr>
<td>Theatre</td>
<td>ISBN: 0413530000</td>
</tr>
<tr>
<td><strong>By New Releases</strong></td>
<td>Published: September 2003</td>
</tr>
<tr>
<td>Coming Soon</td>
<td>Not yet available – PreOrder Today</td>
</tr>
<tr>
<td>New This Month</td>
<td>Our Price: $22.50</td>
</tr>
<tr>
<td>New Releases</td>
<td>Add to Shopping Bag  Add to Wish List</td>
</tr>
</tbody>
</table>

**Figure 4.14: News in www.chapters.ca**
Related Patterns: It will be more effective to use this pattern in combination with Set Based Navigation, and Active Reference. The Set Based Navigation pattern will be used to create sets of new items and the Active Reference pattern will be used to show the current position in the hierarchy of navigation in navigating the new items of a particular category.

4.8 Guided Tour

Name: Guided Tour.

Intent: To provide sequential access to a small group of objects.

Motivation: In a large Web Information System, naive or first-time users will have a little knowledge of the application domain and they need to be acquainted with the content of the application. Some users may prefer to get brief information about the application rather than undergo an exclusive navigation of the whole application. Designer may want the user to navigate a number of same type of objects sequentially.

Applicability: This pattern will be applicable in commercial Web sites to provide brief information about their products. For example, an ecommerce application may sell different types of products and an introduction about those products may be provided instantiating this pattern. Again, a company may produce different types of products, (say a car company produces different model of car) and a short navigation through these products containing brief summary of those products, can be provided to the users instantiating this pattern.

Another applicability of this pattern is in big applications which usually contain a number of sections, and users can be provided with a brief summary of those sections by instantiating this pattern. For example, the Web site of internet movie database (www.imdb.com) contains a number of sections like “Now Playing”, “Movie/TV news”, “My Movies” etc. and the Guided Tour pattern is instantiated to provide an introduction of those sections.
In essence, this pattern is applied either to provide some information about the elements of a particular set created by Set Based Navigation pattern navigating them sequentially or about the different sections of a large application which are represented by presentation class in UWE.

![Figure 4.15: The navigation structure using Guided Tour](image)

**Structure:** Figure 4.16 shows the structure of the guided tour pattern.

![Figure 4.16: The structure of Guided Tour](image)

**Participants:** The participants of this pattern are:

1. **List of Hyperlinks:** It provides a list of hyperlinks of pages of a Web site containing brief introduction about the important sections of the Web site.

2. **Set Based Navigation Pattern:** It does the set based navigation on different types of items.

3. **GuidedTour class:** This class creates a sequential link among the elements of a set created by Set Based Navigation Pattern in navigating different types of
items or the pages containing a brief introduction of the important sections of the Web site. It determines the object to be shown in the interface and also keeps track of the user intention: prev, next. This linking may have different variants like one-way, two-ways or circular. In case of a circular guided tour, it links the last member to the first one. It also may provide several variants in returning to the center: establishing a link from every member, from the last member or from the first member to the start.

4. **View**: It is the user interface to show the presentation classes which are the actual pages to be shown on the browser and generated by GuidedTour class.

**Collaborations:**

1. **List of Hyperlinks**: It provides links of different pages to GuidedTour class.

2. **Set Based Navigation**: It provides the elements of a set to GuidedTour class.

3. **GuidedTour class**: Depending on the applicability of the pattern, this class will collaborate with the other participants in two ways. If the pattern is instantiated to provide a summary about the products or different types of items, it receives the individual elements of a set from the **Set Based Navigation** pattern. If it is instantiated to provide a summary about the important sections of a large application, it receives the links of the pages containing the summaries from the 'List of Hyperlinks'.

4. **View**: It receives the presentation classes from the GuidedTour class to show them in the interface.

**Consequences**: This pattern helps commercial applications in advertising their products or in introducing their products by creating a sequential link among the product descriptions. The circular variant of this pattern avoids the need to scan the whole collection backwards up to the first member. As a result, this variant of this pattern promotes usability.
The main disadvantage of this pattern is that its members must be traversed sequentially. It will not be possible to access a particular member randomly and it will incur a significant overhead, if there are too many members [ACM03].

**Known uses:** This pattern is very common in most of the commercial Web sites or e-commerce applications. Most Web sites have something like “Virtual Showroom” instantiating this pattern. Figure 4.17 shows the instance of the pattern in www.globeandmail.com to show its best daily photographs using this pattern. There is another instance of the pattern in www.nga.gov to navigate different art works of a particular category sequentially.

![Figure 4.17: Guided Tour in http://www.globeandmail.com](image)

**Related Patterns:** This pattern is similar to *Set Based Navigation* pattern. Both patterns are used to navigate a set of elements.

### 4.9 Simple Search Interface

**Name:** Simple Search Interface.

**Intent:** Provide users with powerful, yet simple, search mechanisms [ACM03].

**Motivation:** In a large application, one of the most important concerns is how user will find the desired information. If the information space is very big, it requires an
effective search mechanism. A simple query may satisfy the user’s need in finding the required information instead of complicated search mechanism which may require sophisticated query formulations. For example, in an online store, to find a particular item (e.g., book), a search using a keyword (e.g., name, author, publisher etc.) may be enough instead of browsing all the items which will incur significant overhead. Similarly, to find information about a celebrity, about a movie in an entertainment Web site, a simple keyword search may be enough. Furthermore, users usually want to find the desired information with as little an effort as possible. In this case, a simple search may be more useful than navigation.

The main rationales of this pattern are [ACM03]:

- The information space is unstructured or users don’t have any idea about the category of the expected information.
- The information is structured, but some information relevant to a given topic may appear in more than one group/category.
- Novice users typically do simple searches.
- Simple searches are often enough for the needs of the user.
- Advanced search capabilities require more expertise from the user.
- Formulating advanced queries takes more time, and requires a more sophisticated interface that is justified only when it is really needed.

**Applicability:** Most of the Web sites require search facilities. This pattern is applicable in several situations: when the information space is unstructured, i.e., the user does not understand the organization and classification of the information space; when it is desired to present a very simple search interface.

**Structure:** Figure 4.18 shows the structure of simple search interface pattern.

**Participants:** The participants of this pattern are:

1. **Navigation class:** It represents the collection of items to be explored.
Figure 4.18: The Structure of Simple Search Interface

2. **SimpleSearch class**: This class provides a very simple and straightforward search interface with one field and no operator. This class processes the search string and identifies the navigational classes in which the search string may appear. It does the search and sends the search result to View or to Set Based Navigation Pattern.

3. **Set Based Navigation Pattern**: It is used to explore search result.

4. **View**: It is the user interface to show the pages containing the search result in browser.

**Collaborations:**

1. **Navigation class**: It provides the information space to SimpleSearch class.

2. **SimpleSearch class**: It receives the information space from Navigation class and search string from user; and sends the search result to View or to Set Based Navigation pattern.

3. **Set Based Navigation pattern**: It receives the search result from SimpleSearch class and sends them to View creating sets of related items in the search result.

4. **View**: It receives the pages generated either by the SimpleSearch or by the Set Based Navigation pattern.

**Consequences:** This pattern has the following advantages and disadvantages [ACM03]:

**Advantages:**
- High recall. The goal of any search activity is to provide search results with high precision and high recall and it is possible to trade off precision against recall. This pattern applies the search through the whole information space providing high recall, i.e., as many documents as possible.

- Less navigational overhead for simple queries.

Disadvantages:

- It may decrease the precision. For example, in a university Web site, one is looking for “John”. This pattern will identify all possible “John”s (all professors, students, stuff, and others having the word “John” as part of their name), without any classification or without restricting the search in any specific category.

- Adds an extra element to the user interface, which in turn may result in more complex interfaces for novel users.

**Known uses:** Figure 4.19 shows the instance of the pattern in the University of Victoria. There is another instance of the pattern in CNN. Google provides a simple way to add an instance of this pattern to almost any web site.

![Image](image.png)

**Figure 4.19:** Simple Search Interface in the Web site of the University of Victoria

**Related Patterns:** It is similar to **Selectable Search Space**. The **Simple Search Interface** does the search in the whole information space and the **Selective Search Space** does the search in a particular category. **Set Based Navigation** is used to explore the search result.
4.10 Selectable Search Space

**Name:** Selectable Search Space.

**Intent:** Specify a category within which the search should be made or restricted to [ACM03].

**Motivation:** Most of the Web sites require search facilities and in most of the cases, it is possible to satisfy the user requirements using simple search mechanism that is provided by *Simple Search Interface* pattern. It is possible to create this simple search mechanism by tuning the search result based on how the information space is organized. In most of the cases, information space is structured, i.e., the information space is organized into several categories or is grouped based on semantic relationships. In this case, it will be more effective to restrict the search on a particular category of items rather than applying the search on the whole information space. For example, online stores present their items organizing them in several categories (e.g., book, music, movie). When a user is looking for something concrete, then it will be meaningless to search the item in product categories different from the one with which it belongs to. It would make sense to allow the user to select a category and perform the search only in that category.

The goal of any search activity is to provide search result with high precision, providing only the relevant documents and high recall, providing as many documents as possible and it is possible to trade off precision against recall. When the information space is structured, it is possible to provide more precision by selecting a category and restricting the search only in that category.

**Applicability:** This pattern is applicable with structured information, where it is possible to group semantically related information. This pattern is applicable in the following situations:

- The information space is organized into different categories.
- The user knows the category where the result is expected to be.
- Higher precision is desired.

![Figure 4.20: Structured and Unstructured Information](image)

**Figure 4.20: Structured and Unstructured Information**

**Structure:** Figure 4.21 shows the structure of the selectable search space pattern.

![Figure 4.21: The Structure of the Selectable Search Space](image)

**Figure 4.21: The Structure of the Selectable Search Space**

**Participants:** The participants of this pattern are:

1. **Navigation class:** It represents the collection of items to be explored.

2. **SelectableSearch class:** It provides the users with a mechanism to select a category in which the search will be performed and only one category at a time. It also takes a search string from the user. It sends the search result to View or to **Set Based Navigation pattern**.

3. **Set Based Navigation pattern:** It is used to explore the search result.
4. View: It is the user interface to show the presentation classes, which are the different pages generated either by the SelectableSearch class or the Set Based Navigation pattern.

Collaborations:

1. Navigation class: It provides the items of selected category to SelectableSearch class.

2. SelectableSearch class: It receives the items from navigation class and provides the search result either to View or to Set Based Navigation pattern.

3. Set Based Navigation pattern: It receives the search result from SelectableSearch class and sends them to View creating sets of the search result.

4. View: It receives the presentation classes generated either by the SelectableSearch class or by the Set Based Navigation pattern.

Consequences: This pattern has the following advantages and disadvantages [ACM03]:

Advantages:

- It provides higher precision than that of Simple Search Interface by restricting the search within a particular category.

- Using this pattern, fewer documents are retrieved than that of Simple Search Interface and increases user's satisfaction.

Disadvantages:

- Categories usually must be determined manually.

- Having too many categories effects its usability. Therefore, a certain balance among the granularity of the search spaces and their quantity must be achieved.

Known uses: Figure 4.22 shows the instance of the pattern in www.chapters.ca. There is another instance of the pattern in www.amazon.com.
Related Patterns: It is advanced version of Simple Search Interface pattern and Set Based Navigation pattern is used with it to explore the search result.

Other Related Patterns: It can be used in combination with Node In Context to automatically set the searching category [ACM03].

4.11 Shopping Basket

Name: Shopping Basket

Intent: Keep track of user selections during navigation, making these selections persistent so they can be processed when the user decides to. Decouple product selection from product consumption and/or processing [RSL99].

Motivation: In an ecommerce application, users navigate through the information space and decide what and when they will buy. For example, in http://www.amazon.com, a user can browse through hundreds of books or CDs and choose a subset of them to buy. Usually it can be done in two ways: (1) the user will buy the product in the moment it is found or (2) bookmark all the desired products and buy them in different navigation sessions. But these approaches are not suitable in cases when users want to buy several products. It will be impractical for both the user and the shop to require one transaction for each product. It will also require the user to navigate to the check-out page many times which will be time consuming.

It is possible to provide a better solution by allowing the users to select the products
to buy as they are traversed and to provide a persistent store for those items such that they can be accessed as another navigation object and associate processing operations such as removing an item, showing details etc [ACM03].

**Applicability:** This pattern is applicable in all ecommerce applications, where selected items need to be stored for further processing.

**Structure:** Figure 4.23 shows the structure of the shopping basket pattern.

![Diagram of the shopping basket pattern](image)

**Figure 4.23:** The structure of Shopping Basket pattern

**Participants:** The participants of this pattern are:

1. *Set Based Navigation pattern:* Objects that can be placed into the shopping basket are navigated using this pattern.

2. *SelectedItems:* It is a navigation class containing items selected by the user.

3. *ShoppingBasket class:* This class does all the processing like removing items, showing details etc. on the items selected by the user, i.e., the items in the SelectedItems.

4. *View:* It is the user interface to show the different pages on browser that are represented as a presentational class in UWE.
Collaborations:

1. *Set Based Navigation pattern:* It provides the items navigated and selected by the user to SelectedItems for persistent storage.

2. *SelectedItems:* It creates a persistent storage of the items selected by the user, which can be considered as a Navigational class and sends them to ShoppingBasket class for further processing.

3. *ShoppingBasket class:* It gets the items selected by the user from SelectedItems and creates the pages to be shown on the View doing some processing on those items like removing items, showing the details etc..

4. *View:* It gets the pages from ShoppingBasket class and shows them on browser.

Consequences: This pattern will simplify the buying process in online store.

Known Uses: Figure 4.24 shows the instance of the pattern in www.chapters.ca. There is another instance of the pattern in www.amazon.com.

Figure 4.24: Shopping Basket in www.chapters.ca

Related Pattern: It is also similar to the Visited Objects pattern.
Acknowledgments: The original authors of the patterns Set-Based Navigation, Landmark, Shopping Basket, Simple Search Interface, Selectable Search Space and News were D. Schwabe, G. Rossi, and F. Lyardet. The original authors of Active Reference were D. Schwabe, G. Rossi and A. Garrido. The original authors of the pattern Guided Tour were P. Paolini, F. Garzotto, S. Valenti and D. Bolchini.
5. Java Implementation

5.1 Introduction

The nine navigational design patterns, which have been described in the catalog in Chapter 4, are implemented in Java using the Apache Tomcat server. Each pattern is implemented as a component which is instantiable in different applications. A class library is developed implementing those patterns.

5.2 Component:

According to Clemens Szyperski [Szy99], “A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.” The most desirable attributes of components are:

- Reuse: One of the main objectives of a component is to provide reuse. Design and develop something and use it over and over again in different contexts, thereby realizing large productivity gains, taking advantage of best-in-class solutions, the consequent improved quality, and so forth [CD01].

- Replaceable: The major challenge for the component to system development is managing change. It is very obvious that a system will evolve, the requirements may change and hence, one of the primary objectives of a component is that it must be easily replaceable- either by a completely different implementation of the same functions or by an upgraded version of the current implementation [CD01].

- Units of deployment: A component is a unit of independent deployment. A software is what is actually deployed - as an isolatable part of a system. For a component to be independently deployable, the component needs to be well separated from its environment and from other components [Szy99].
- Interface: A component needs to come with clear specifications of what it requires and provides. In other words, a component needs to encapsulate its implementation and interact with its environment through well defined interfaces [Szy99].

The patterns included in this pattern system are implemented in the form of components. Each pattern is implemented containing one or more classes and each pattern can be treated as one component. The developed components support reusability and it is the main objective of this implementation. The developed components can be replaced by a different implementation as long as they have the same interfaces and those interfaces are described in AppendixB. The components have an interface and can interact with their environments through interfaces. The developed components are units of deployment. In other words, the developed components have the desirable attributes of a component.

5.2.1 Navigational Framework

The implemented patterns have behavioral relationships among them and they are integrated into a framework. Each pattern provides some assistance in information navigation and all together, they form a framework of information navigation. It is a component oriented navigational framework implementing the navigational design patterns for information navigation on Web and it is reusable and extensible.

5.3 Parameterization

The main objective of the framework is to navigate different types of objects in Web information systems. The implemented patterns can be instantiated in different applications to navigate different types of objects or collections of items using Parameterization. Any design pattern can be parameterized via two configuration files, one for navigation: Navigation Parameters and another for presentation: Presentation Parameters.

1. Navigation Parameters: These parameters describe the type of navigational object that the pattern is going to be used upon. They describe how the pattern instance should retrieve the data from a given database.
These parameters include the elements of the database of the collection like database name, table name, different fields etc. These parameters are used in forming sets, and subsets of objects. All these parameters are read from text files. For example, to do Set Based Navigation in a collection of book, the text file containing the parameters of this collection is read. To do the same on a collection of movies, the text file containing the parameters of movie collection is read. In this way by reading the appropriate text file, one can navigate any collection.

2. Presentation Parameters: This parameterization determines how the pattern will look when it is instantiated. For instance, for set-based navigation, it specifies how the table of contents of the set is created and displayed, for each element (e.g. a book), it specifies which of its attributes are shown (title, authors, year of publication) and its typeset (e.g. font type, size, color, etc). This parameterization is done via text file, XML documents and XSL and CSS style-sheets.

In this implementation, Tomcat server is used. Some of the presentation parameters which are less likely to be changed very frequently, are put in the server start-up file, server.xml or Web deployment descriptor, web.xml file. The others are kept in text files. For example, the captions of the fields to be shown are also parameterized. Suppose, for example, that for a collection of books, one wants to present the name, author etc. For name, the caption may be "The Name of the Book", "Name", "Book Name" etc. The captions of all the fields are put in a text file. To manipulate the color, font and size of a particular field, stylesheet is used and the particular values of those colors, fonts, sizes of text etc. will come from parameters.

The parameter setup is the responsibility of the designer or developer. This parameterization is totally transparent from the normal users of the application.

5.4 Objectives of the framework

The patterns are implemented and combined considering the following issues:
1. **Information Navigation:** Information navigation is the most important activity of any Web application especially for Web Information Systems. According to Lowe and Hall [LH99], the single most evident characteristic of any hypermedia application, and the one which is the most likely to affect the ability of an application to achieve its goal, is the usability of the application and the functionality provided to support general navigation and browsing within the information space will be critical to the usability of an application.

It assists the users to understand the overall information space and the user’s location within this space. Navigation also helps users to remember where the user has been and allows the users to return to these places. It aids in identifying the information and provides searching and indexing mechanisms to help the users in finding information.

In this pattern system, the patterns are selected focusing on information navigation. The patterns are implemented to organize the information such that the users can navigate them easily. An effective navigation mechanism is provided by structuring the information space (e.g., individual landmarks), identifying the information (e.g., forming meaningful sets of related items), suitable linking mechanism among the pages, and providing simple search mechanism. An effective information navigation is provided by the approach *Parameterization*. For example, one can navigate any collection using parameters. The information is presented forming sets of related items and in forming set, one can use any attribute to form the sets using parameters, which provides more diversity in information navigation.

2. **Information Presentation:** A suitable presentation is critical to identify appropriate information. It is not possible to use the information if one can’t sense it and information presentation should support information interpretation. It focuses on what will be the look and feel of the information being presented.

In this pattern system, the patterns are implemented using *parameterization* and it is possible to provide a flexible information presentation. For example, one can choose
any attributes of a navigational class and for different navigational classes different attributes suitable for that navigational class for information presentation. One can choose different formats of information presentation (e.g., stylesheet, font, size). One can also choose the placement of different components (e.g., button, menu, text) using parameters.

3. Reusability: It has been described that the patterns are the problems and solutions of those problems that occur over and over again in our environment. The main goal of the patterns is to provide reuse of knowledge. Patterns are abstract concept and provide design reuse. The goal of this work is to embed those concepts into code and provide code reuse.

The main objective of this framework is to provide reusability and it is done by using two different approaches: parameterization and inheritance.

4. Extensibility: Another consideration of this implementation is to support extensibility. Developed components are extensible. Developed classes can be inherited to add special behavior.

The framework is developed first by implementing the Set Based Navigation pattern. Then it is extended by implementing the other patterns and integrating those into the framework. The framework can be extended by integrating the other patterns too. In this case, the following issues have to be considered:

- Navigation parameters: How the implemented pattern will agree with the navigation parameters.

- Presentation Parameter: What are the presentation parameters needed for the new components provided by the new patterns, and how they agree with the presentation parameters of existing components provided by the already instantiated patterns.

- Behavioral Relationship: How the already instantiated patterns will refer this pattern and vice versa. In other words, the relationship of this pattern with the
5.5 N-tier Architecture

An N-tier architecture is a system with at least three logical tiers or layers [d-198]:

- Presentation-tier: It is the front end layer for users usually Web browsers or specific purpose applications with graphical user interfaces. It is responsible for the presentation of data, receiving user events and controlling the user interface.

- Application-tier: It is the middleware layer for business logic, such as Web servers and application servers. Business-objects that implement the business rules exist here. This tier protects the data from direct access by the clients. Java servlets are server side components and all those components reside in this tier.

- Data-tier: This is the back-end tier responsible for data storage. It contains the relational databases.

If a system has more than three layers, then the additional layers would be application tier or data tier. Figure 5.1 demonstrates the N-tier approach and it is ideal for dynamic content generation [Van03]. In this pattern system, the developed components and the framework

![Diagram of N-tier Architecture](image)

**Figure 5.1: N-tier Architecture**
follow the 3-tier architecture, where-

Client(Presentation-tier): contains the HTML pages that are sent to the client.

Server(middle-tier): contains all the components or the servlet classes. It processes the request from the client, and it is responsible for calling the corresponding Java class that implements the design pattern.

Data-tier: contains the application database and the parameters of the patterns.

The architecture of the framework is depicted in Figure 5.2.

![Architecture Diagram](image)

**Figure 5.2: The architecture of the framework**

### 5.6 Using the patterns

Two applications are developed integrating all the components of the those patterns to show how to use them. The applications are developed to navigate different types of
collection with different *navigation parameter* and *presentation parameter* using the same components of the implemented patterns.

### 5.6.1 Integration of the patterns

The main purpose of the developed components is to provide reusability. One must be able to integrate the patterns in different applications. To do so, one has to write a wrapper class in the form of Java servlet. There are three ways to integrate the patterns in that template:

- **Inheritance**: The classes of the developed components can be extended and by calling the methods of these classes, one can get the behavior of a pattern. One can add the necessary code in the derived class to add additional behavior.

- **RequestDispatcher**: One can integrate the components in that template by forwarding a request and including a response from the component using the RequestDispatcher, which is an interface of the Java servlet. One can use the forward and include methods of RequestDispatcher to share data with the template, in broader sense, among the applications.

- **Object**: One can instantiate the classes to create an object and then calling the methods, one can include the behavior of the pattern.

This is how to integrate the components in different applications. In other words, how the components are reusable. Here, it is possible to argue how the components are different from standard classes. The developed components will be run in a server and one can send request to those components and include the response using an interface without knowing the details of those components, which is never possible in the case of a standard class.

### 5.6.2 Case Study-1:

An application is developed, the purpose of which is to navigate a collection of book, movie and employee on the Web. Figure 5.3 shows a screenshot of the navigation of a
collection of books and Figure 5.4 shows the navigation of a collection of employees.

The member of Java

Name of the Book: Advanced Java

Author of the Book: O'Reilly

Book price: 40.9

Description: This is an advanced Java book

you are here: Bookstore > Programming > Java >

Figure 5.3: Case Study-1

Here, the same classes of the implemented patterns have been used in both the cases. The Navigation Parameter identifies it as a collection of books or employees and Presentation parameter identifies what attributes to be presented (e.g., for a collection of book it may be name, author, price and description or any other), the caption of individual attributes presented (e.g., Name, Name of the book), and the stylesheet (e.g., font, color.).

5.6.3 Case Study-2:

Another application is developed to navigate a collection of dvd and video on the Web using different Navigation parameter and Presentation parameter. The roles of Navigation Parameter and Presentation Parameter are same as described in Case Study-1. Figure 5.5 shows a screenshot of the navigation of a collection of DVDs and Figure 5.6 shows the navigation of a collection of videos.
5.7 Restrictions

The developed components have the following restrictions:

- The information must be represented in a relational database.
- To use the components, one has to write a wrapper class in the form of Java servlet.
- The active reference pattern shows only the intra-set navigation of set based navigation.
- The guided tour pattern will produce a circular sequential link among the elements.
Figure 5.5: Case Study-2
Figure 5.6: Case Study-2
6. Conclusions

There is a good number of hypermedia design patterns and they appear in different journal papers, conference papers and online documentation. The patterns are scattered in different publications and they are written in different notations. There are also some duplicate patterns. Some patterns are extended versions of others. Users are required to scan a lot of publications to find a pattern if there is one. In this thesis, all the publications are scanned and a list of patterns is provided collecting all the published patterns upto 2003. The list includes the name and intent of each pattern and also the publications in which it has been appeared.

In this thesis, nine navigational patterns are selected based on their behavioral relationships. The list included in this thesis contains 96 patterns and about one fourth of them are navigational patterns. The selected patterns are refined and rewritten. The relationship among the selected patterns is depicted.

The selected patterns are written in Gamma notation. This notation provides good separation of concerns. The structure of the pattern in this notation shows the participants of the pattern, and helps in identifying the classes or objects to implement the pattern. The structure diagrams of the selected patterns are drawn using UWE and UML notations. These structure diagrams don’t appear in the other publications.

A framework is created for the implementation of the patterns. The selected patterns are implemented in Java using the Apache Tomcat server so that they can be instantiated in different applications to get the behavior of the patterns. The implemented patterns are integrated to provide a navigational framework for information navigation on Web.

Two applications were developed instantiating the implemented patterns. It was shown that it is possible to get the behavior of the patterns in different applications of same domain using the same implementation.
6.1 Observations

The Gamma notation has a substantially different structure from the Alexandrian notation. The Gamma notation lends a uniform structure to the information contained in the pattern, and makes them easier to learn, compare, and use. It also represents the pattern using a graphical structure, which shows what are components present within a pattern. From the structure, participants and collaborations, one can identify the classes/objects and their relationships, which will facilitate the implementation or use of the patterns in practical applications [Joh03].

The existing eight patterns included in this system were originally written in the Alexandrian notation. These descriptions were something like prose and one could argue difficult to read. They provided a brief textual description for the solutions of the patterns. Gamma notation breaks up the solution into a number of sections. In this format, the solution is described by identifying the participants and collaborations among them and presenting them using some design diagrams or structure diagrams, which is expected to increase the understandability and assist in the implementation of the patterns.

One important contribution of this work is the structure diagram of the patterns, which represents the patterns graphically. None of the previous versions of these patterns contains a structure diagram of the patterns. It can be argued that the patterns described herein using this notation are more explanatory, easier to understand, and to use in practical applications.

Patterns are abstract concepts and independent of any implementation. The patterns in the Gamma notation follow this appropriately. The sample methods included in the structure diagram are just to explain the activities or the responsibilities of the different components or the participants contained in the structure diagram and to provide some hints about its potential implementation. Moreover, this notation supports the well-known software engineering concept: Separation of concern. This notation splits the description in different sections, each showing a different perspective of the pattern, like what are the motivations of this pattern, what are situations for this pattern to be used, how this pattern solves the problem, etc. Again, the solution in this description separates the different partic-
ipants including their responsibilities and collaborations. For navigational design patterns, in many cases, this description follows the N-tier or 3-tier architecture. For instance, the structure diagram of *Set-based navigation* identifies three parts: the Navigation class, the SetBased class and the Presentation class, where the Navigation class belongs to *Data tier*, SetBased class belongs to *Middle tier* and presentation class belongs to *Presentation tier* of well-known 3-tier architecture.

The depicted relationships among the patterns shows how the patterns of this system are connected together or how they will interact with one another when they are being instantiated in application. This will help the users in choosing the required pattern(s) from the patterns included in this system for an ecommerce application or an Web Information Systems.

Both UML and UWE modeling elements are used in drawing the structure diagrams and were enough for the patterns included in this pattern system. Combined UML and UWE contain enough modeling elements to design software or hypermedia application. But none contains modeling elements to represent the use of a pattern. In this thesis, the component diagram has been used to represent a pattern. The patterns are the micro-architectures and the isolatable parts in the architecture of an application. Components are also the isolatable parts of a developed application. So, it can be argued that it is reasonable to implement a pattern in the form of a component and a component diagram is appropriate to represent a pattern as a whole.

The list of 96 published patterns collected in this thesis arguably forms sufficient base patterns to create system of patterns. The selected patterns have been described in uniform notation. In this description, the *Related patterns* section shows what are alternatives of a pattern within the patterns included in this system and the *Other related patterns* section shows what are the alternatives from the patterns not included in this system. All these related patterns described in both sections are the constituent patterns of a pattern. The relationships among the selected patterns has been identified. They are implemented. Their possible evolution also has been described. In essence, the system of patterns created in this thesis, satisfies all the requirements of Buschmann *et al.*
6.2 Contributions

In this thesis, a system of patterns has been created containing nine patterns. The pattern Visited Objects is new while the other eight patterns are already exist in the literature. The system of patterns has been created by following this process [AG03]:

- provided a comprehensive list of published patterns,
- selected nine navigational design patterns, refined them, and described their evolution,
- described the selected nine patterns in Gamma notation,
- created a framework for the implementation of the patterns,
- implemented the selected patterns and developed two applications using them.

In essence, a system of patterns has been created for navigation of information on Web. This pattern system fulfills all the required conditions to be a pattern system.

6.3 Future Work

It is possible to extend this pattern system by adding more patterns such as Opportunistic Linking, Hierarchical Structure Through Navigation Side Bar, History. It has been stated that different authors are writing the patterns in different notations and in this thesis, the patterns are written in the Gamma notation. The next step is to collect feedback from different researchers and practitioners about the notation used in this thesis and based on their opinions, it is possible to come up with a universally accepted pattern language.

In this thesis, only a particular type of patterns, navigational patterns, is considered. It should be evaluated whether it is possible to write all types of patterns in the same notation or not or how the other patterns of all categories can also be written in the same notation used in this thesis.

One of the factors that affect the usability and success of any complex hypermedia application depends on its architecture and it is already stated that patterns are the microarchitectures. It should be evaluated how a pattern oriented hypermedia architecture can be
designed or the impact of patterns in designing hypermedia architecture. It is possible to concentrate on the whole application or part of it like pattern-oriented navigation design.

It is often unrealistic to start from scratch in developing a new product or migrating to a modern architecture. A realistic approach can start by analyzing legacy systems to understand the current architecture and developing a strategy for mining and reusing existing assets. Mining involves rehabilitating parts of an old system for use in a new system. The technical approaches focus on two aspects of mining assets for a product: (1) mining existing systems at an architectural level through Mining architectures for product (MAP) line evaluation and (2) mining components for a product line through Options Analysis for Reengineering (OAR) [Car03].

Patterns play a significant role in the architecture of an application and an application usually contains a number of patterns. Therefore patterns can play a significant role in Reengineering hypermedia applications. Especially for MAP approach, patterns may be very valuable. Patterns are the isolatable parts in the architecture of the application and may be the most potential candidates to be extracted from an existing legacy system in migrating into a new system. An application may become a legacy system, but patterns are not supposed to, as they are tested and proven techniques. At least, the patterns should have a longer life cycle than the whole application unless the patterns themselves become obsolete or outdated through evolution over time. It may be possible to develop a framework in pattern oriented Reengineering of hypermedia applications.
Bibliography


[GHJV96] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley, Reading, 1996.


APPENDIX A

A Compendium of Patterns

A.1 List of Patterns

Active Reference: Provides a perceivable and permanent reference about the current status of navigation [ACM03, GRS97, LRS98b, RSG97]

Advising: Help the user find a product in the store. Assist him according to his wishes [ACM03].

Analyse Organise Synthesize: Provides a more systematic approach to multimedia development, which results in high quality products and in the increase in productivity [ACM03]

Anchor: How to activate links from within a node? [GRS97]

Avatar: How can a self-representation of users be provided in an intuitive way? [SS99]

Behavioral Anticipation: How do you indicate the effect or consequence of activating an interface object? [ACM03, GRS97]

Behavioral Grouping: How to organize the different types of controls in the interface so the user can easily understand them? [ACM03, GRS97]

Bread crumbs: The users need to know where they are in a hierarchical structure [Mar03].

Carrot and a Stick: How do you get end users to provide information that they are reluctant to share? [PK02]

Clustering: Avoid the presentation of more than 7 items simultaneously. [NN98]

Collector: How to make a set of elements behave in the same way depending on one element. [DM99]
Collection Center: To make a collection well understood and usable. [ACM03]

Communication Channel: How can information be exchanged that is not directly related to the document content? [SS99]

Complex Entity: To design an information intensive object, wishing to provide a complete, hypertextual, not annoying data consumption for the user. [ACM03]

Component Layout: Several artifacts need to be arranged in respect to their audio-visual properties. [CL98]

Compound: How to describe the resulting behavior when two elements are joined to work together. [DM99]

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. [NN98]

Contour: Cycles overlap on each other, allowing free movement from one cycle to another. [Ber98]

Counterpoint: Two “voices” alternate, interleaving, giving the reader the option to either follow one or to jump from one to the other. [Ber98]

Cycle: The reader returns to a previously visited node and departs along a new path. [Ber98]

Decorator: Provides a flexible alternative to subclassing for extended functionality. [GLWG99]

Doormat Navigation: Users need to be directed to the right section of the Web site [Mar03].

Double Tab Navigation: The users need to navigate a hierarchical structure. [Mar03]

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. [LRS98a]

Easy Undo: Provide safe undoing capabilities in a complex process. [ACM03]
Explicit Process: Help the user understand the buying process when it is not atomic. [RLS00]

Faceted Navigation: The users need to select one item out of many items [Mar03].

Fly-out Menu: Users need to select a submenu item [Mar03].

Glue: Joins a number of multimedia artifacts into a single composite artifact. [CL98]

Group Location Awareness: How can we provide a permanent reference about the user's current locations in the collaborative hypermedia space? [SS99]

Guided Tour: To provide an “easy-to-use” access to a small group of objects, assuming that user has no reason (or is not able) to select one of them. [ACM03]

Headerless Menu: Users need to access the main sections of the site [Mar03].

Here I am: How do we keep the user informed about the actions that can be carried out within a complex and/or non-standard interface? [ACM03]

Hybrid Collection: To provide an easy-to-use access to a small group of objects, allowing both a complete scan and the search for a specific member. [ACM03]

Index Navigation: To provide a fast access to a group of objects, for users who are interested to one or more of them, and are able to make a choice. [ACM03]

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. [st99]


Hyper-Map: Provides an interface to geographical information. [D. 97]

Icon Menu: Users need so make a selection out of a limited set of items [Mar03].
Image Browser: Users need to view and/or select images out of a set of images [Mar03].

Information Factoring: Presents information needed by the reader to understand a given topic/information unit. [LRS98b, ACM03]

Information on Demand: Lets users decide which items they want further described in the context of the same node. [GRS97, LRS98b, RSG97, ACM03]

Information-Interaction Coupling: How do we make clear what is the object affected by a control in a node’s interface? [GRS97, ACM03]

Information-Interaction Decoupling: How do you differentiate contents and various types of controls in the interface? [GRS97, ACM03]

Landmark: Provide direct access to critical sub-systems in the WIS. [G. 99]

Link as a Relationship View: How to represent the relationships among the components of an application in a hypermedia view of those components, and allow to navigate those relationships? [GRS97]

Link Creation Method: When is it better to create static links, and when is it preferable to create links through computations? [GRS97]

Link Destination announcement: Avoids unnecessary link firing by providing information about the destination. [NN98]

Logical Glue: Small information sets need to express meaningful structure to avoid being perceived as an arbitrary grouping. [NN98]

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. [NN98]

Meta Navigation: Users want to know who they are dealing with [Mar03]

Minesweeping: Users need to be stimulated to interact with navigational elements [Mar03].

Mirrorworld: Provides two or more views of the same information. [Ber98]
**Missing Link:** Suggests a link that does not exist. [Ber98]

**Montage:** Several distinct writing spaces appear simultaneously, maintaining their separate identities. [Ber98]

**Navigation Strategy:** How to allow static as well as dynamic acquisition of a link destination? [ACM03]

**Navigational Context:** Provides the user with closed navigational subspaces containing context-related guidelines and relationships. [GRS97, LRS98b, RSG97, ACM03]

**Navigational Feint:** Establishes the existence of a navigational opportunity that is not meant to be followed immediately. [Ber98]

**Navigational Observer:** Decouples the navigation process from the perceivable record of the process. [RSG97]

**Neighborhood:** Establishes an association among nodes through proximity, shared ornament, or common navigational landmarks. [Ber98]

**News:** Allows easy access to new information items as the WIS grows. [ACM03, G. 99, LRS98a]

**Node Creation Method:** When is it better to create nodes statically, and when is it preferable to create nodes dynamically? [GRS97]

**Node as a Navigational View:** How to add navigation capabilities to the components of an existing application (for example a DBMS one), therefore adding hypermedia functionality to it? How to combine conventional transactional processing with navigation? [ACM03]

**Node as a Single Unit:** How do you decide the extent of a node? [GRS97, LRS98b]

**Node Class, Link Class:** How can node types be defined, according to the classes of an OO application that is extended with hypermedia functionality, or according to types defined by a database, thus automating or easing node creation? [GRS97]
Opportunistic Linking: Keep the user interested in the site. Seduce him to navigate in 
the site even when he has already found what he was looking for. [ACM03]

Outgoing Links: Users need to know when they leave the current context [Mar03].

Partitioned Incremental Development: Provides the basis for development of hyperme-
dia in an incremental manner, supporting progressive integration and delivery of 
components.[Low99]

Plan B: How do you support users who are unable to perceive web page graphics? [PK02]

Policy Statement: How do you establish sufficient trust with users so that they will pro-
vide personal information? [PK02]

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? [ACM03, GRS97]

Push Communication: Simplify the searching process for customer-selected areas or prod-
ucts. [RLS00]

Repeated Menu: Users need to access the main navigation [Mar03]

Required Field Markers: How do you ensure that the end user provides information es-
sential to the use of a web application? [PK02]

Selectable Keywords: Suggest a set of keywords that are relevant to the results of previous 
searches, in order to improve the quality of subsequent search results. [ACM03]

Selectable Search Engine: To enhance a website search capabilities by delegating the ac-
tual search service to more sophisticated engines, which may be able to produce a 
Structured Answer with more detailed or better formatted information. [ACM03]

Selectable Search Space: Specify a category within which the search should be made or 
restricted to. [ACM03]

Session: How can we structure collaboration between users and groups of users? [SS99]
Set-based navigation: Organizes the information in sets of related information items. Provide intra-set navigation capabilities. [ACM03, G. 99]

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. [G. 99]

Sieve: Sorts readers through one or more layers of choice in order to direct them to a given section. [Ber98]

Simple Search Interface: Provides users with powerful, yet simple, search mechanisms. [ACM03]

Shortcut Box: Users want to access specific functionality in a direct way [Mar03].

Split/Join: Knits two or more sequences together. [Ber98]

Split Navigation: The users need to navigate a hierarchical structure [Mar03].

Structured Answer: Provides the user with an organized information report as the result of a query. [ACM03]

Scrolling Menu: The user needs to select an image out of a set of images [Mar03].

Tangle: Confronts the reader with a variety of links without providing clues to guide the reader’s choice. [Ber98]

Teaser Menu: User has to select a link [Mar03].

Template: A need exists to produce a collection of composite artifacts similar in structure and contents. [CL98]

Trail Menu: User need to find information in a hierarchical structure [Mar03].

User Role: How to represent the different behaviors a user shows, depending on the collaborative context? [SS99]
Virtual Product: Displays a product as part of an electronic catalog. [D. 97]

Virtual Room: How can we structure collaboration between users and groups of users in a natural and intuitive way? [SS99]

What They See is All They Get: How do you make sure an end user sees everything on a web form that the user needs to see? [PK02]

Wrapper Node: How can we add navigational behavior to existing GUIs? [GRS97]
APPENDIX B

Interfaces

B.1 Interfaces of the Components

Set Based Navigation

public interface SetBased{
    public String[] createSet(Collection info);
    public String[] getSetMembers(String set);
    public String[] getElementContent(String et);
    public Page createMenu(String[] set);
}

Landmark

public interface Landmark{
    public void createLandmark(URL []url);
    public Page display(URL url);
}

Active Reference

public interface ActiveReference{
    public String getCurrentPos();
    public URL combinePos(URL url, String currPos);
    public Page display(URL url);
    public Page backtrack(URL url);
}

Visited Objects

public interface ListVisited{

public Page showVisitedItems(String item);
public Page showDetails(String item);
}

Simple Search Interface

public interface SimpleSearch{
  public String getSearchString();
  public String[] doSearch(String srchString, Collection info);
  public Page showResult(String []result);
}

Selectable Search Space

public interface SelectSearch{
  public String getCategory();
  public String getSearchString();
  public String[] doSearch(String srchString, Collection info);
  public Page showResult(String []result);
}

Guided Tour

public interface GuidedTour{
  public Page nextRequest(URL url);
  public Page prevRequest(URL url);
  public Page showCurrentElement(URL url);
  public void returnToStart(URL startUrl);
}
News

    public interface News{
            public String[] getNews(Time tLimit, Collection info);
    }

Shopping Basket

    public interface ShopBasket{
            public void addElement(String elem);
            public String[] removeElement(String elem);
            public Page showEList();
            public Page showDetail(String elem);
            public Boolean buyElements();
    }