Pattern-Oriented Design Composition and Mapping for Cross-Platform Web Applications

M. Taleb[1], H. Javahery[2], A. Seffah[3]

[1]Human-Centered Software Engineering Group, Concordia University
Montreal, Quebec, Canada
Telephone: +1 514 848 2424 ext. 7165

mtaleb@encs.concordia.ca

[2]Human-Centered Software Engineering Group, Concordia University
Montreal, Quebec, Canada
Telephone: +1 514 848 2424 ext. 7165

h_javahe@encs.concordia.ca

[3]Human-Centered Software Engineering Group, Concordia University
Montreal, Quebec, Canada
Telephone: +1 514 848 2424 ext. 3024

seffah@encs.concordia.ca

ABSTRACT

In the context of cross-platform Web applications, Pattern-Oriented Design (POD) proposes that developers use proven solutions emerging from best design practices in order to solve common design problems. In addition, it requires composing patterns to create a platform-independent design and then mapping these pattern-oriented designs to specific platforms. This prevents the designer from reinventing the wheel and can have positive implications on system performance, scalability and usability. In this paper, we introduce different types of Web design patterns, as well as different composition and mapping rules to design a multi-platform Web application. We discuss why patterns are a suitable means for creating and mapping a Web design to new platforms while maintaining usability.

Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: User Interfaces. D.2.11 [**Software Architecture**] Patterns

General Terms

Design, Usability, Human Factors, Standardization, Languages, Web Engineering

Keywords

Design Patterns, Usability, Pattern-Oriented Design, Web Applications, Mapping, Pattern Composition

Number of pages: 14

DSVIS2006, July 26-28, 2006, Trinity College Dublin, Ireland.
The XIII International Workshop

1. Introduction

In recent years, the Web has matured from offering simple static-page functionality to providing intricate processes such as end-to-end financial transactions. Users have been given more sophisticated techniques to interact with available services and information using different types of computers. Different kinds of computers and devices (including, but not limited to, traditional office desktops, laptops, palmtops, PDAs with and without keyboards, mobile telephones, interactive televisions) are used for interacting with such applications. One of the major characteristics of such cross-platform Web applications is that they allow a user to interact with the server-side services and contents in various ways. Web applications for small and mobile devices are resource constrained and cannot support the full range of Web application features and interactivity because of the lack of screen space or low bandwidth**.**

The mosaic of Web applications and platforms has led to the emergence of Web engineering as a sub-discipline of software engineering with some specific challenges. One important question is how to develop and deploy the same application for different platforms – without "architecturing" and specifically writing code for each platform, or learning different languages, not to mention the many Web design guidelines that are available for each platform.

Even if other definitions are possible; we will use the following terminology. A pattern is a proven design solution to a common problem which occurs in different contexts of use. An instance of a pattern is defined in the POD approach as a specific implementation of pattern in a certain context. An example is the two implementations of the "*Convenient Toolbar pattern*" for a PDA and Desktop. Within this paper, we are considering only Web Sites even if the approach may apply to other forms of sites. Example of another platform, at least when we define the relationships. Consider some patterns from the GUI, Web and mobile patterns catalog and Welie patterns [4]. We are adapting and refining Web patterns and our pattern-oriented design approach to address some of the challenges of designing cross-platform Web applications. Some of the reasons are listed here.

First, POD can help with decoupling the different aspects of Web application architectures and isolate platform specifics from remaining concerns that are common to all platforms. As with other multi-tiered schemes such as client-server architectures, POD proposes a common information repository, which is at the core of multi-layer architectures. Services should be accessed strictly through an adaptable presentation layer, which provides decoupling of the data from the device-specific interfaces. In this way, developers need only worry about the standardized middleware interface rather than having to concern themselves with the multitude of toolkits put forth by database repository manufacturers. Segmenting the architecture and reducing coupling to stringent specifications allows designers to quickly understand how changes made to a particular component affect the remaining system.

Second, the diversity of platforms exhibits drastically different capabilities that designers need to take into account. For example, PDAs use a pen-based input mechanism and have average screen sizes in the range of 3 inches. On the other hand, the typical PC uses a full size keyboard, a mouse and has an average screen size of 17 inches. Coping with such

drastic variations implies much more than mere layout changes at the presentation level. Pen-based input mechanisms are slower than traditional keyboards and thus are inappropriate for applications such as word processing that require intensive user input. Similarly, the small screens available on many PDAs only provide coarse graphic capabilities and would be ill-suited for photo editing applications. We will show how the mapping rules can be applied to adapt a design to the capabilities of a platform while providing replacement and alternative navigation and interaction patterns.

Third, many system manufacturers and researchers have issued design guidelines to Web application designers [1]. Recently, Palm Inc. has put forth design guidelines to address navigation issues, widget selection, and the use of specialized input mechanisms such as handwriting recognition. Microsoft, IBM and Sun Microsystems have also published their own guidelines to assist developers with programming applications targeted at the Pocket PC/Windows CE platform. However, these guidelines are different from one platform or device to another. When designing a multi-device Web application, this can be a source of many inconsistencies. The Java "look-and-feel" developed by Sun has been introduced as a set of cross-platform guidelines that can solve some of these inconsistencies. However, these guidelines do not take into account the particularities of a specific device, such as the platform constraints and capabilities. This can be problematic for a user interacting with different kinds of devices to access server-side services and information of a Web application. Furthermore, for novice designers, it is hard to remember this mosaic of design guidelines let alone use them effectively. Like many others who have been experimenting with POD and patterns, we are proposing to use patterns as an alternative design tool to guidelines. Patterns are considered as a generic building block that can be instantiated for each platform.

The proposed POD-Web approach consists of composing different architectural, navigational and interaction patterns to create high level and platform independent designs. A sub-objective of our research is to define rules to compose and map these designs. The foundation of our approach is a multi-layered and pattern-oriented design architecture.

2. Background Work

Introduced by the architect Christopher Alexander in 1977 [8], design patterns can be viewed as building blocks that can be composed to create designs. A single pattern describes a problem that appears constantly in our environment, with a corresponding solution to this problem, and in a way which allows designers to reuse this solution for different platforms [8]. For cross-platform Web applications, patterns are interesting for three reasons. See [9] for a more general discussion on patterns and their benefits:

- They come from experiments on good know-how and were not created artificially;
- They are a means of documenting architectures;
- In the case of cross-platform development, they make it possible for the team to have a common vision.

Every pattern has three essential elements, which are: A context, a problem, and a solution. The context describes a recurring set of situations in which the pattern can be

applied. The problem refers to a set of forces, i.e., goals and constraints, which occur in the context. Generally, the problem describes when to apply the pattern. The solution refers to a design form or a design rule that can be applied to resolve the forces. The solution describes the elements that constitute a pattern, relationships among these elements, as well as responsibilities and collaboration.

Similar to the entire Software Engineering community, Web engineers and the user interface design community has been a forum for vigorous discussion on pattern languages for user interface design and usability. The goals of HCI patterns are to share successful user interface design solutions among HCI professionals, and to provide a common ground for anyone involved in the design, development, usability testing, or the use of highly interactive systems including the Web. A number of pattern languages have been suggested. For example, Van Duyne's "The Design of Sites" [6], Welie's Interaction Design Patterns [4], and Tidwell's UI Patterns and Techniques [3] play an important role. In addition, specific languages such as Laakso's User Interface Design Patterns and the UPADE Web Language [8, 18] have been proposed as well. Different pattern collections have been published including patterns for Web page layout design [3, 4, 5], for navigation in large information architectures, as well as for visualizing and presenting information.

In our work, we investigate how these collections of proven Web design patterns are "composed" into reliable, robust and large-scale Web interactive systems. However, the development of web applications using design patterns as design components requires a careful look at composition techniques. Several methods have been proposed for composition, within approaches such as Pattern-Oriented Design (POD). For example, Yacoub and Ammar [7] proposed two composition techniques:

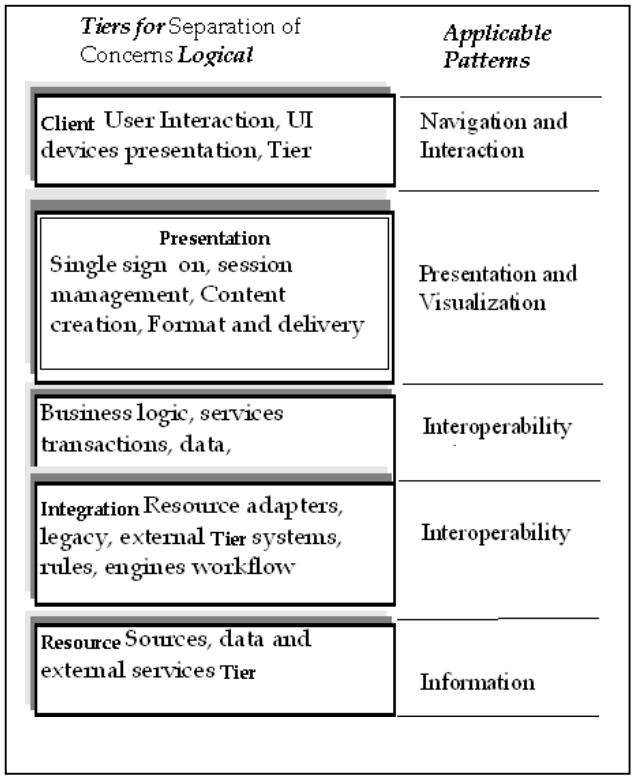
- **Behavioral composition techniques** that are based on object interaction specifications to show how instantiations of patterns can be composed.
- **Structural composition techniques,** which are based on the static architectural specifications of composed instantiated patterns using class diagrams.

The Pattern-Supported Approach (PSA) [10] to the user interface design process suggests a wider scope for the use of patterns by looking at the overall design process. Based on the fact that the usability of a system emerges as the product of the user, the task and the context of use, PSA integrates this knowledge in most of its patterns, dividing the forces in the pattern description correspondingly (i.e., describing Task, User, and Context forces). PSA provides a double-linked chain of patterns (parts of an emerging pattern language) that *support* each step of the design process.

Building on PSA, our proposed approach highlights another important aspect of Pattern-Oriented Design: *Pattern combination*. By combining different patterns, developers can use pattern relationships and combine them in order to produce an effective design solution. We will consider this principle in section 4. As a result, patterns become a more effective vehicle that supports design reuse.

3. The Proposed Web Design Patterns

We conducted a survey in 2004, which informed us that there are at least six types of design patterns that can be used in Web application engineering (Figure 1).



Navigation patterns implement proven techniques for navigating within and/or between a set of pages and chunks of information.

Interaction Patterns focus on the interaction mechanisms that can be used to achieve tasks and the visual effects they have on the scene, as such they relate primarily to graphical and rendering transforms.

Presentation patterns provide solutions for how the contents or the related services are visually organized into working surfaces, the effective layout of multiple information spaces and the relationship between them. These patterns define the physical and logical layout suitable for specific Web pages such as home page, lists, tables, and forms.

Visualization Patterns suggest different visual representations/metaphors for grouping and displaying information in cognitively accessible chunks. They define mainly the form and content of the visualization i.e. the graphical scene, and as such relate primarily to data and mapping transforms.

Interoperability patterns aim to decouple the different layers of a Web application. In particular, between the content, the dialog and the views or presentation layers as outlined in Figure 1. These patterns are generally extensions of the Gamma design patterns such as MVC (Model, View and Controller) observer, and command action patterns.

Information patterns describe different conceptual models and architectures for organizing the underlying content across multiple pages, servers and computers. Such patterns provide solutions to questions such as

Figure 1: A Pattern-Oriented Architecture of a Web Application

In what follows, we introduce some concrete examples of the patterns that we have been using. These examples also show the need to combine several types of patterns to provide solutions to complex problems. The list of patterns is not exhaustive. There is no doubt that more patterns still need to be documented, and that an endless number have yet to be discovered.

To organize the content of a Web application, the simplest pattern is the *sequence pattern* which organizes a set of interrelated pages in a linear narrative. This pattern applies to information that naturally flows as a narrative, time line, or in a logical sequential order. Hierarchical organization schemes, as with the *hierarchy pattern*, are particularly well suited to Web application content, because Web sites should always be organized as offshoots of a single Home Page [1]. Many procedural manuals, lists of university courses, or medical case descriptions are best organized with the *grid pattern*. As illustrated in Figure 2, several patterns need to be combined to organize the entire information content of a complex Web application.

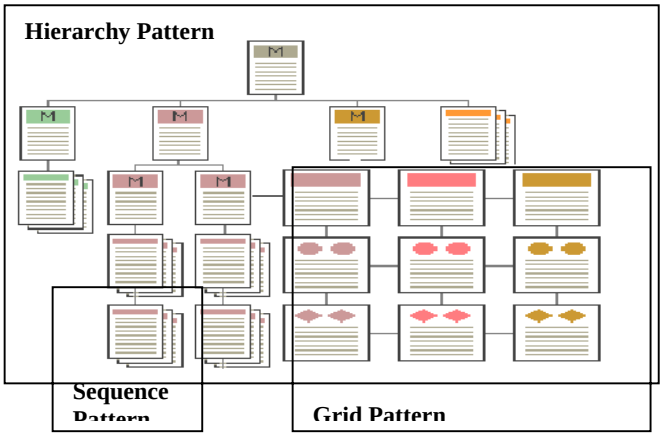


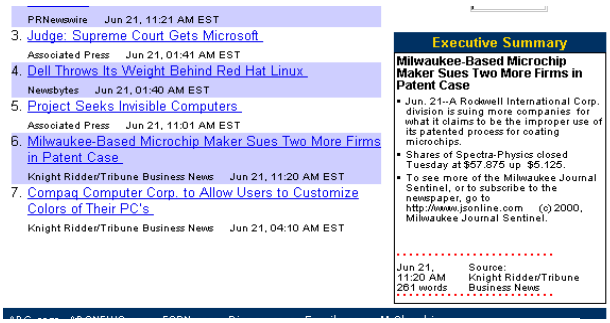
Figure 2: A Combination of Information Architecture Patterns

Navigation patterns are fundamental in Web design since they help the user navigate easily and clearly between information chunks and pages. They can obviously reduce the user's memory load [1, 7]. See also [3, 5, 8, 9] for an exhaustive list of navigation design patterns. The following are some of the basic patterns:

Shortcut Pattern. Lists the frequently visited pages or used services. They are generally embedded in the home page and help experienced users find their favorite information and services with one mouse click.

Dynamic Path Pattern (or Bread Crumb) is a very useful pattern that indicates the entire path taken by the user since accessing the Web application.

Index Browsing Pattern allows a user to navigate directly from one item to the next and back. The ordering can be based on a ranking.



The results of the *Search Pattern* (described below), with the complicity of the *Executive Summary Pattern* (a page layout pattern), provides users a preview of underlying information before spending time downloading, browsing and reading large amounts of information included in subsequent pages (Figure 3).

Figure 3: Example of structural patterns

Information overload is another fundamental problem in Web engineering. Web applications, especially large Web portals, provide access to millions of documents. The designer must consider how best to map the contents into a graphical representation that

conveys information to the user while facilitating the exploration of content from a large Website. In addition, the designer must provide dynamic actions that limit the amount of information the user receives while at the same time keeping the user informed about the content as a whole.

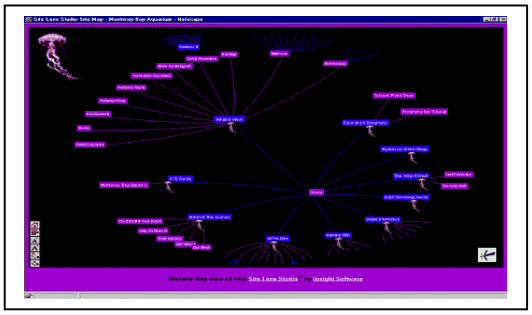


Figure 4: The Navigation Spaces Map Patterns

Favorites Collection, *Bookmark*, *Frequently Visited Pages*, *Preferences* and *Navigable Spaces Map* patterns are some of the information visualization patterns for solving another complex design problem. As presented in Figure 4, these patterns are generally composed to provide a comprehensive map to a large amount of content, which cannot be reasonably presented in a single view. The underlying content can be organized using the combination of patterns presented in Figure 2; into distinct conceptual spaces or working surfaces that are semantically linked to each other.

Communication and interoperability design patterns are useful patterns to facilitate the mapping of design between platforms. Examples of patterns that can be considered to ensure the interoperability of Web applications include *Adapter*, *Bridge*, *Builder*, *Decorator*, *Facade*, *Factory Method*, *Mediator*, *Memento*, *Prototype*, *Proxy*, *Singleton*, *State*, *Strategy*, and *Visitor* [2].

4. Pattern Composition Rules

A platform-independent pattern-oriented design exploits several relationships between patterns. Gamma et al. emphasize in their book “Design Patterns” that defining, as part of the description of a pattern, the list of related patterns is a key notion in the composition of patterns and their usages. Zimmer [11] implements this idea by dividing the relations between the patterns of the Gamma catalog itself in 3 types: “X is similar to Y”, “X uses Y”, and “Variants of X uses Y”. They are in fact relationships between patterns in a certain context; in other terms they are relationships between instances of patterns. Based on Zimmer’s work, we define five types of relationships between patterns.

1. **Similar** (X, Y) if and only if X and Y can be replaced by each other in a certain composition. This means that X and Y are patterns of the same category and they provide different solutions to the same problem in the same context. As illustrated in Figure 5, *Index Browsing* and *Menu Bar* patterns are similar. They both provide navigational support in the context of a medium size Web site.

Figure 5: Similar Pattern extracted from OMG site

2. **Competitor** (X, Y) if X and Y cannot be used at the same time for designing the same artifact relationship that applies to two patterns of the same pattern category. Two patterns are competitors if and only if they are similar and interchangeable. For example, the *Web convenient toolbar* and *Index Browsing* patterns are competitors. The *Index Browsing* pattern can be used as a shortcut toolbar that allows a user to directly access a set of common services from any Web page. The *Convenient Toolbar* that provides the same solution is generally considered more appropriate.
3. **Super-ordinate** (X, Y) is the basic relationship to compose several patterns of different categories. A pattern X is a super-ordinate of pattern Y means that pattern Y is used as a building block to create pattern X. An example is the *Home Page* pattern which is generally composed of several other patterns (see Figure 6)
4. **Sub-ordinate** (X, Y) if and only if X is embeddable in Y. Y is also called super-ordinate of X. This relationship is important in the mapping process of pattern-oriented design from one platform to another. For example, the *Convenient Toolbar* pattern is a sub-ordinate of the *Home Page* pattern for either a PDA or Desktop Web application. Implementations of this pattern are different for different devices, as will be discussed in the next section.
5. **Neighboring** (X, Y) if X and Y belong to the same pattern category (family). For example, neighboring patterns may include the set of patterns to design a specific page such as a home page.

5. An illustrative Example

To illustrate the use of the relationships described in the previous section, we discuss the redesign of the home page of a popular Web site for biomedical experts, the National Center for Biotechnology Information (http://www.ncbi.nlm.nih.gov). This Web site is "extra-large" in size [5], providing information and a large set of services. The home page is the starting point for most user visits, and is similar to a portal. Improving the home page multiplies the entire Web site's usability and increases the accessibility and visibility to the 10, 000 pages included in this site.

Based on an empirical study we conducted [5] and recommendations [9, 10], a usable and easy to maintain home page for this large Web site can be developed by combining the following patterns (Figure 6):

1. *Tangline Pattern*. The home page must explain what the site does and what makes it unique among competitors.
2. *Web Convenient Toolbar Pattern*. This navigation design pattern provides access to the most common services such as help, feedback, etc.
3. *Search Pattern* is important for any big Web site. When users want to search, they typically scan the home page looking for "the little box where I can type," so your search should be a box.
4. *Frequently visited pages Pattern*. Users will often remember good articles, products, or promotions.

5. *Site Map Pattern*. This information visualization design pattern summarizes the structure of the underlying content architecture.
6. *About Pattern*. Looking for specific information is rarely a user's first task, but sometimes people do need details about who you are. An "About <company-name>" pattern is the best way to link users to more in-depth information than can be presented on the home page.
7. *Executive Summary Pattern*. Don't just provide a link to some pages behind the home page. Show some of your best or most recent content. This can be achieved via the use of different instances of the executive summary pattern.
8. *Index Browsing Pattern*. Your home page should offer users a clear starting point for the main pages that users can undertake when visiting your site.
9. *Disclaimer Pattern*.
10. *Maintainer Info Pattern*. Sometimes pages are wrong; even the best proofreaders make mistakes. Therefore, a home page provides a link to the page's maintainer.
11. *Go Safe Place Pattern*. This pattern allows the user to go back to the home page from any page. To familiarize the user with it, this pattern should be visible on the home page.

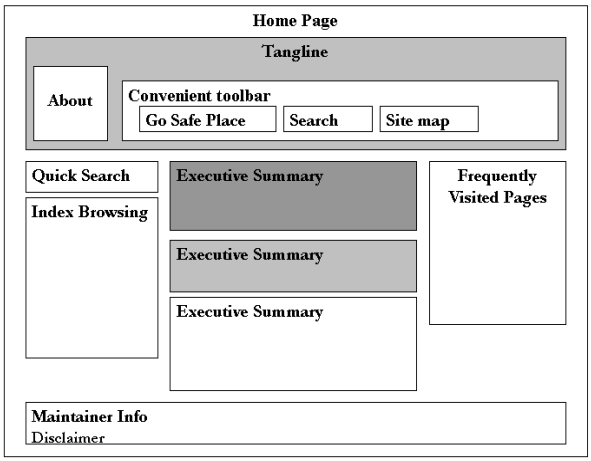


Figure 6: A Home Page Design using Patterns

6. Pattern-Oriented Design Mapping

Using a pattern-oriented design as a starting point, it is possible to redesign a Web application for other platforms, by using what we call *pattern mapping*. The original set of patterns used in the Web application are transformed or replaced in order to redesign and re-implement the application, and in particular the user interface, for mobile or PDA applications. Since patterns hold information about design solutions and context of use, platform capabilities and constraints are implicitly addressed in the transformed patterns.

Figure 7: The Web Convenient Toolbar Pattern Implementations and Look & Feels for Different Platforms

Figure 7 illustrates different mappings of the *Quick Access pattern* for three different platforms. For a web browser on a desktop, it is implemented as an *index browsing toolbar*. For a PDA, the Quick Access pattern can be implemented as a *combo box*. For a mobile phone, the Quick Access pattern is implemented as a *selection* [4]. Pattern descriptions should provide advice to pattern users for selecting the most suitable implementation for a given platform.

To illustrate the mapping-based design approach, in what follows, we describe the effect of screen size on the usage and selection of patterns. Different platforms use different screen sizes, and these different screen sizes afford different types and variants of patterns. The question when mapping a POD is how the change in screen size between two platforms affects redesign at the pattern level. The amount of information that can be displayed on a given platform screen is determined by a combination of area and number of pixels. The total difference in information capacity between platforms will be somewhere between these two measures of 20 times the area and 10 times the pixels. We can conclude that to map the desktop display architecture to a PDA display architecture, the options are as follows:

1. To reduce architecture size, it is necessary to significantly reduce both the number of pages and the quantity of information per page.
2. To hold constant the architecture size (i.e. topics or pages), it is necessary to significantly reduce the quantity of information per page (by a factor of about 10 to 20).
3. To retain the full amount of information in the desktop architecture, it is necessary to significantly increase the size of the architecture, since the PDA can hold less information per page.

The choice of mapping strategy will depend on the size of the larger architecture and the value of the information:

- For small desktop architectures, the strategy can be weighted either towards reducing information if the information is not important, or towards increasing the number of pages if the information is important.
- For medium or large desktop architectures, it is necessary to weigh the design strategy heavily towards reducing the quantity of information, since otherwise the architecture size and number of levels would rapidly explode out of control.

Finally, we can consider mapping of patterns and graphical user objects in the context of the amount of change that must be applied to the desktop design or architecture to fit it into a PDA format. The following is the list of mapping rules we suggest:

1. **Identical**. For example, drop-down menu patterns can usually be copied without transformation from a desktop to a PDA.
2. **Scalable** changes to the size of the original design or to the number of items in

the original design. For example, a long horizontal menu pattern can be adapted to a PDA by reducing the number of menu elements.

3. **Multiple** of the original design, either simultaneously or sequentially in time. For example, a single long menu can be transformed into a series of shorter menus.
4. **Fundamental** change to the nature of the original design pattern while replacing it generally by another one. For example, permanent left-hand vertical menu patterns are useful on desktop displays but are not practical on most PDAs. In the transformation to a PDA, left-hand menus normally need to be replaced with an alternative solution such as drop-down menu patterns.

These mapping rules can be used by designers in the selection of patterns especially when different patterns apply for a platform and not another one, when the cost of adapting or purchasing a pattern is high, or when validity of a pattern is questionable.

This taxonomy of mapping types is especially relevant to the automation of cross-platform designs using patterns since the designs that are easiest to transform are those that require the least transformation. The taxonomy therefore identifies where human intervention will be needed for design decisions in the transformation process. In addition, when building a desktop design for which a PDA version is also planned, the taxonomy indicates which patterns to use in the desktop design to allow easy transformation to the PDA design.

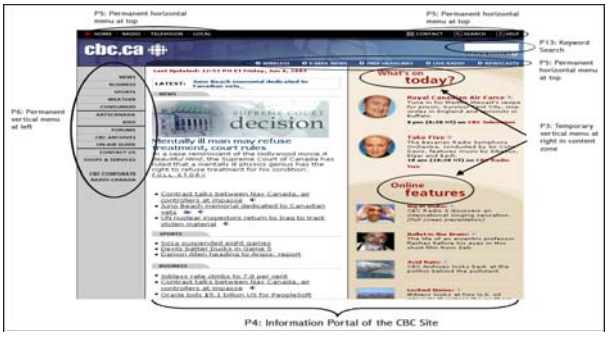


Figure 8: Patterns Extracted from the CBC News

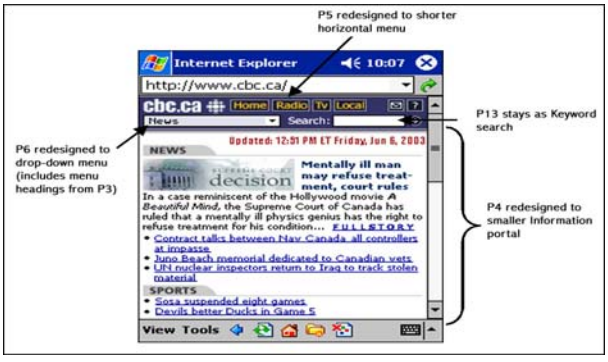


Figure 9: Migration of the CBC site to a PDA Platform using Pattern Mapping

Figure 8 illustrates some of the navigation design patterns as used in the home page of a desktop-based Web application. Once these patterns are identified in the desktop-based Web application, they can be transformed or replaced by others in a PDA version.

Figure 9 demonstrates the redesigned interface of the CBC (Canadian Brodcasting Corporation, www.cbc.com) site for migrating to a PDA platform. The permanent horizontal menu pattern at the top (P5) in the original desktop UI was replaced with a shorter horizontal menu pattern (P5s). In order to accommodate this change on the small PDA screen, the three different horizontal menus had to be shortened, and only important navigation item was used. The keyword search pattern (P13) remains as a keyword search. The permanent vertical menu at the left (P6) is redesigned to a drop-down menu (P15). The drop-down menu in the PDA design also includes the menu headings, "***What's on today***?" and "***Online features***" from the temporary vertical menu (P3) in the original desktop design. Finally, the information portal (P4), which is the first thing that captures the user's attention, is redesigned to a smaller information portal (P4s).

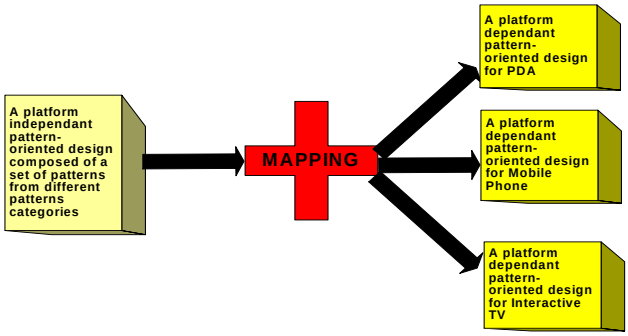


Figure 10: Pattern-Oriented Design Mapping Architecture

In short, what we have just illustrated in this section and the examples of Figures 7, 8 and 9 can be characterized in the form of architecture of composed pattern-oriented design mappings (see Figure 10).

7. Discussion

Table 1 describes the types of cross-platform mappings that are recommended for some of the most popular Web patterns, and which can be used to redesign a Web site for the PDA display. These mappings offer the closest and simplest equivalent platform-dependent designs for the corresponding platform. In the third column, the suffix "s" after a pattern indicates, "scaled (down)", and the suffix "m" indicates "multiple (sequence)". In this paper, we have introduced a pattern-oriented design method that essentially exploits pattern composition mappings. This approach is a significant improvement over non-structured migration methods currently in use, for the following reasons:

HCI pattern in desktop display	Type of transformation	Replacement pattern in small PDA display
P.1 Bread crumbs	Scalable or fundamental	P.1s - Shorter bread crumb trail; P.15 - Drop-down "History" menu.
P.2 Temporary horizontal menu	Scalable or fundamental	P.2s - Shorter menu; P.6 - Link to full-page display of menu options ordered vertically
P.3 Temporary vertical menu in content zone	Identical, scalable or fundamental	P.6 - Temporary vertical menu in content zone; P.6s – Shorter temporary vertical menu; or P.15 - Drop-down menu
P.4 Information portal	Scalable	P.4s - Smaller information portal
P.5 Permanent horizontal menu at top	Scalable or fundamental	P.5s – Shorter horizontal menu at top; P.6 - Link to full-page display of menu options ordered vertically
P.6 Permanent vertical menu at left	Fundamental	P.15 - Drop-down menu
P.7 Progressive filtering	Identical	P.7 - Progressive filtering
P.8 Shallow embedded vertical menus	Identical or fundamental	P.6m – Sequence of temporary vertical menus in content zone; P.8 - Shallow embedded vertical menus
P.9 Sub-site	Scalable or fundamental	P.6 – Temporary vertical menu in content zone; P.9s – Smaller sub-site;
P.10 Container navigation (3 containers)	Scalable or fundamental	P.10s - Container navigation (2 containers); P.7 - Progressive filtering
P.11 Deeply embedded vertical menus	Multiple or fundamental	P.6m – Sequence of single-level menus; P.8m - Sequence of shallow embedded menus
P.12 Alphabetical index	Scalable	P.12s - Alphabetical index (less items per page, or smaller index)
P.13 Key-word search	Identical	P.13 - Key-word search
P.14 Intelligent agents	Identical	P.14 - Intelligent agents
P.15 Drop-down menu	Identical, scalable or fundamental	P.15 - Drop-down menu; P.15s – Shorter drop-down menu; Hyperlink to P.6 – Temporary vertical menu in content zone
P.16 Hybrid navigation: Key-word search	Identical or scalable	P.16s - Hybrid approach with smaller or less deeply embedded menus

Table 1: Examples of Patterns Mapping

- The method provides a standardized table of pattern mappings, thereby reducing the redesign effort and ensuring consistency in redesign.
- The standardized rules for composition and mapping formalize best practices in design, thereby ensuring optimal quality of the migrated user interface.
- The method helps designers in design choices associated with (1) the size of the source architecture and target architecture and (2) the amount of information to maintain in migrating from the source platform to the target platform.
- The method is simple enough to be used easily by novice designers, as compared to reengineering which currently requires a considerable degree of expertise and abstract reasoning ability.

Pattern-Oriented Design offers the very useful ability of easily extracting multiple platform-specific designs from a single generic (platform-independent) UI model. However, the current state of the art in patterns and cross-platform research is not yet mature enough to handle all the requirements of pattern-assisted design techniques. Before generic UI pattern-based models can be defined, more research must be addressed to define the multiple levels of abstraction of patterns and to create a clear, well-structured taxonomy of patterns. The simplified taxonomy presented in section 3 is a starting point. Thus, within a pattern-based framework, the simplified "redesign and design" method proposed here is currently the most practical approach for migration of a Web application between platforms.

8. A Concluding Remark

In this paper, our discussion focused on a way to combine and map several types of patterns to create a pattern-oriented design for cross-platform web applications. The proposed POD approach aims to demonstrate when a pattern is applicable or required in a design process, how it can be used, as well as how and why it can or cannot be combined with other related patterns. In POD, cross-platform Web application developers can exploit the composition and mapping relationships and the underlying best practices to derive a concrete design from a pattern-oriented design.

Our investigations are based on several years of Web application development, ethnographic interviews with Web developers, as well as expert recommendations. Such recommendations include reported best practices for using patterns as a bridge over the gaps between design practices and software tools [11, 7]. Our experiences have also highlighted that in order to map patterns, especially by novice designers and software engineers who are unfamiliar with Web engineering, patterns should be presented to developers using a flexible structure to represent patterns, to make it easy for both the pattern authors, reviewers and users.

One of the major problems we found is that mastering and applying large collections and different types of patterns requires in-depth knowledge of both the problems and forces at play. As such, it is inconceivable that pattern hierarchies, composition and mapping rules will evolve strictly from theoretical considerations. Practical research and industry feedback are crucial in determining how successful a pattern-oriented design framework is at solving the cross-platform design problems listed in section 1. It is therefore essential to build an "academia-industry bridge" by establishing formal communication channels between industrial specialists on Web patterns, software design patterns, information architecture patterns as well as software pattern researchers. It is hoped that such collaboration will lead to a common POD framework that is essential in making the large diversity of patterns accessible to common Web engineering designers.

At this time the POD approach, including the list of patterns and relationships, has been defined and illustrated for Web Sites. We can explore other forms of Web applications such as transactional applications, e-commerce, etc. Further work needs to be done for exploring: (1) the scalability of the approach to given multiple patterns, platforms and platform design guidelines, and (2) the strategies for automating the POD approach as well as the reuse of patterns.

9. References

[1] Lynch, P.J, and Horton, S. Web Style Guide: Basic Design Principles for Creating Web Sites. New Haven and London: Yale University Press. 1999.

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

[3] Tidwell, J. Common Ground: A Pattern Language for Human-Computer Interface Design, 1997. http://www.mit.edu/~jtidwell/common_ground.html

[4] Welie, M.V. The Amsterdam Collection of Patterns in User Interface Design 1999- http://www.cs.vu.nl/~martijn/patterns/index.html

[5] Engelberg, D., and Seffah, A. Design Patterns for the Navigation of Large Information Architectures. 11th Annual Usability Professional Association Conference Orlando, Florida, July 8-12, 2002.

[6] Duyne D. K. van, Landay, J. A, and Hong J. I. The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Addison-Wesley, 2003.

[7] Sherif Yacoub, Hany Ammar, 2003, *Composition of Design Patterns*, Addison Wesley Professional, ISBN 0-201-77640-5, 416 pages, Hardcover, Germany.

[8] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiskdahl-King, S. Angel. A Pattern Language, Oxford University Press, New York, 1977.

[9] F. Buschmann, What is a pattern? Object Expert vol. 1(3), pp17-18, Mars/April 1996.

[10] Åsa Granlund, Daniel Lafrenière, David A. Carr, 'A Pattern-Supported Approach to the User Interface Design Process', Proceedings of HCI International 2001 9th International Conference on Human-Computer Interaction, August 5-10, 2001, New Orleans, US.

[11] Walter Zimmer. Relationships between design patterns. New York, NY, USA, 1995. ACM Press / Addison-Wesley Publishing, pp 345–364.