Patterns-Oriented Design Applied to Cross-Platform Web-based Interactive Systems

M. Taleb
Human-Centred Software Engineering, Group Concordia University, Montreal, Quebec, Canada
Telephone: +1 514 848 2424 ext 3024 mtaleb@encs.concordia.ca

A. Seffah
Human-Centred Software Engineering Group Concordia University Montreal, Quebec, Canada
Telephone: +1 514 848 2424 ext.3024 seffah@cse.concordia.ca

A. Abran
Software Engineering Department & Information Technology, École de Technologie Supérieure (ÉTS),
Montreal, Quebec, Canada
Telephone: +1 514 396 8800 ext. 8632 aabran@ele.etsmtl.ca

Abstract

Design pattern suggests that developers must be able to reuse proven solutions emerging from the best design practices to solve common design problems while composing patterns to create reusable designs that can be mapped to different platforms. Without this, cross-platform Web applications designers are not properly applying design solutions or take full benefit of the power of patterns as reuse blocks, resulting in poor performance, poor scalability, and poor usability. Furthermore, the designer may “reinvent the wheel” when attempting to implement the same design for different platform. In this paper, we first introduce different types of Web design patterns as a vehicle for capturing and reusing good designs while detailing a motivating exemplar on how Web design patterns can be composed to create a generic Web application design. Then, we discuss why patterns are a suitable means for mapping a Web design to new platforms while maintaining its intrinsic quality attributes.

Keywords
Design Patterns, Patterns-Oriented Design, Web Applications, Mapping, Patterns Composition

1. Introduction

In recent years, the Web has matured from offering just simple functionality like displaying static pages to being able to cater to intricate processes such as end-to-end financial transactions. The user has been given also more sophisticated techniques to interact with the information. Furthermore, the convergence of the Internet, mobile telephony, and personal digital assistants (PDAs) technologies has led to the emergence of a new generation of Web applications. One of the major characteristics of such applications is that they allow a user to interact with the server-side services and contents in a different of ways. Different kinds of computers and devices (including, but not limited to traditional office desktop, laptops, palmtop, PDA with and without keyboards, mobile telephone, interactive television are used for interacting with such applications.

In this technological context, we are moving away from a print paradigm of a mere “brochureware” to Web sites as highly interactive information systems. Such kind of applications includes also mobile telephone embedded Internet applications (read e-mail, browse mobile portal sites, etc.). These resource constrained Web applications for small and mobile devices cannot support the full range of Web applications features because of the lack of screen space or low bandwidth.

This mosaic of Web applications has lead to the emergence of Web engineering as a sub-discipline of software engineering for creating high-quality Web interactive systems. One of the fundamental questions addressed by Web engineering is how can we develop and deploy the same application for different platform available today without architecturing and writing specific for each platform, learning different languages and the hundreds of Web design guidelines that are available for each platform.

Here, we are borrowing, adapting and refining the so popular and powerful patterns-assisted development to
cross-platform Web applications. The following are some of cross-platform Web engineering challenges that we are addressing specifically while adapting the pattern-oriented design approach.

Firstly, the POD should help in decoupling the different aspects of Web application architecture and isolate platform specifics from remaining concerns. The vast majority of current industry Web applications have adopted a layered architecture. As with other multi-tiered schemes such as client-server architecture, a common information repository is at the core of the multi-layers architecture. The services should be accessed strictly through the an adaptable presentation layer, which provides decoupling of the data from the device specific interfaces. In this way, device application 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 designer to quickly understand how changes made to a particular component affect the remaining system. That is because achieving these goals requires a consistent approach to applying both cognitive and social factors to user interface design, and that would require independent developers to coordinate their activities. Unfortunately, conspiring at this level may be beyond the abilities of the industry.

Secondly, 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 mechanism are slower than traditional keyboard 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 thus would be ill suited for photo editing applications.

Finally, many system manufacturers and researchers have issued design guidelines to capture and reuse the best Web design practices [1]. Recently, Palm Inc. has put forth design guidelines to address navigation issues, widget selection, and 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-devices 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. However, cross-platform guidelines do not take into account the particularities of a specific device, in particular the platform constraints and capabilities. This can be a source of problems for a user using different kinds of devices to interact with the server side services and information of a Web application. Furthermore, for a novice designer or a software engineer who is not familiar with this mosaic of guidelines, it is hard to remember all design guidelines, let alone using them effectively.

In this paper, we introduced pattern-oriented design as a method to overcome such challenges. The proposed approach consists: (1) to compose different architectural, navigational and interaction patterns to create a high level and reusable designs, (2) to adapt this composing design to a particular platform using the mapping, i.e., make a transformation of this multiplatform architecture to obtain one specific and particular platform. To understand this point, refer to the section 5, which illustrates it with example. A sub-objective of our research is to define a systematic methodology, supported by a Web CASE tool, to compose and map a design using multi-layer pattern-oriented design architecture.

2. Background work

Introduced by the architect Christopher Alexander in 1977 [14], design pattern can viewed as a building block that we compose to create a design. A single pattern describes a problem, which appears constantly in our environment, and thus described the hart of the solution to this problem, in a way such as one can reuse this solution for different platform, without ever doing it twice in same manner [14]. For Web cross-platform application, patterns are interesting for three reasons; see also [15] for a more general discussion on patterns benefits:

- They come from experiments on good know-how and were not created artificially;
- They are a means of documenting architectures (out of building or software);
- They make it possible in the case of a cross-platform development in team to have a common vision.

Similar to the entire Software Engineering community, the HCI/UCD community has been a forum for vigorous discussion on pattern languages for user interface design and usability engineering. The goals of HCI patterns is 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 interactive systems. Several HCI practitioners and interactive application designers have become interested in formulating HCI patterns (pattern languages). A number of pattern languages for interaction design have been suggested. For example, Van Duyne’s "The Design of
Sites” [10], Welie’s Interaction Design Patterns [5], and Tidwell’s UI Patterns [3] and Techniques play an important role. In addition, specific languages such as Laakso’s User Interface Design Patterns and the UPADE Web Language have been proposed as well.

The idea of using patterns for Web engineering is not new. Different pattern collections have been published including patterns for Web pages layout design [3, 4, 5], for navigation in a large information architecture as well as for visualizing and presenting information. In our work, we investigate Web Patterns and Patterns Oriented Design (POD) as a solution for cross-platform application and in particular, to solve the following design challenges.

Pattern-Oriented Design introduces a methodology for "composing" proven design patterns into reliable, robust large-scale software 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. For example, Yacoub and Ammar [12] proposed two composition techniques:

- **Behavioural 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) 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 [16]. Building on PSA, the proposed approach highlighted another important aspect of pattern-oriented design: **pattern combination**. By combining different patterns, developers can utilize 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 taxonomy

We propose at least six types of design patterns that can be used in Web-based applications engineering. Together these patterns provide an integrative solution to address the multi-faces of Web design:

1. **Information Architecture Design Patterns.** This category of patterns describes different conceptual models and architectures for organizing the underlying content across multiple pages, servers and computers. Such patterns provide solutions to questions such which information can be or should be presented on which device?

2. **User Interface and Page Layout Design Patterns.** These 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.

3. **Navigation Design Patterns.** These patterns implement proven techniques for navigating within and/or between a set of pages and chunks of information.

4. **Interaction Design 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.

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

6. **Interoperability and System Design Patterns.** These patterns aim to decouple the different layers of a Web application in particular between the content, the dialog and the views or presentations layers. These patterns are generally extensions of the Gamma design patterns such as MVC (Model, View and Controller) observer, command actions patterns.

In what follows, we have introduced some concrete examples of this mosaic of patterns that we have been using. These examples have shown also 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 are still to be discovered, and that an endless number have yet to be invented.

The simplest pattern to organize the content of a Web application is in the sequence pattern that 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 are particularly well suited to Web applications 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 the grid patterns. Several patterns need to be combined to organize the content of a complex Web application.

Navigation Patterns are fundamental in Web design in the sense that they help the user to recognize the document's allocation and navigate between chunks of information and pages easily and clearly. They can obviously reduce the user's memory load [7, 1]. See also [3, 5, 8, 9] for an exhaustive list of navigation design patterns. The following are some of the basic patterns: Shortcuts pattern, Dynamic Path Pattern or Bread Crumb, and Index Browsing pattern.

A critical design issue in particular for resource constrained (small) devices is how long does it take to determine if a document has relevant information? The index browsing patterns – A typical navigation design pattern, with the complicity of Executive Summary pattern, a page layout pattern, provide users a preview of underlying information before spending time downloading, browsing and reading large amounts of information.

Information overload is another of the fundamentals Web engineering problems. 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 a large Website content. 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.

The following are some of the information visualization patterns for displaying such solving this complex design problem: collections Favourites, Bookmark, Frequently Visited Page, Preferences and Navigable Spaces Map patterns. These patterns need to be composed to provide a comprehensive map to a large amount of content that can be too much to be reasonably presented in a single view. The underlying content can be organized, using the combination of patterns, into distinct conceptual spaces or working surfaces, which are semantically linked to each other.

Communication and interoperability design patterns are fundamental patterns to facilitate the mapping of design between platforms. Example 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, Visitor [2].

4. Pattern composition rules

A creation of a platform independent pattern oriented design exploits several relationships between patterns. Based on previous work [13], we identify four types of relationships.

1. Similar is a relationship, which applies to the same category of patterns.
2. Competitor is a relationship that applies to two patterns of the same patterns category.
3. Super-ordinate is the basic relationship to compose several patterns of different categories.
4. Subordinate. If pattern X is super-ordinate of Y and Z then Y and Z are sub-ordinate of X. This relationship is important in the mapping process of pattern-oriented design from a platform to another one.

5. An illustrative example

To illustrate the use of the relationships described in the previous section, we discuss here the redesign of the home of one of the most popular Web site for bio-informaticians, the National Center for Bioinformatics Information http://www.ncbi.nlm.nih.gov/. This extra large Web site provides information and offers a large set of very powerful, yet difficult to use, services for biologists. The home page is its facade to the world and the starting point for most user visits. Improving the home page multiplies the entire Web site's usability and increase the accessibility and visibility to the 10,000 pages included in this site.

Based on an empirical study we conducted [8] and recommendations from others [9, 10], a usable and easy to maintain home page for this large Web site can be developed by combining the following patterns [8]:

Tagline Pattern, Web Convenient Toolbar Pattern, Frequently- or Most Visited Pages Pattern, Site Map Pattern, About Pattern, Executive Summary Pattern, Index Browsing Pattern, Disclaimer Pattern, Maintainer Info Pattern, Go Back To Safe Place Pattern.

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 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.

The different mappings of the Quick Access pattern for three different platforms. This navigation design pattern helps the user reach specific pages, which reflect important website content, from any location on the site. For a news Web application, it can provide direct and quick access to central pages such as Top Stories, News, Sports, and Business. 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 [5]. 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 pattern. 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 choice of mapping strategy will depend on the size of the larger architecture and the value information.

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 list is ordered from the most direct to the least mapping rules:

1. **Identical**. No changes.
2. **Scalable** changes to the size of the original design or to the number of items in the original design.
3. **Multiple** of the original design, either simultaneously or sequentially in time.
4. **Fundamental** change to the nature of the original design pattern while replacing it generally by another one.

This taxonomy of mapping types is especially relevant to the automation of cross-platform design 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.

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.

The redesigned interface of the CBC (Canadian Brodcasting Corporation) site for migrating to a PDA platform. The permanent horizontal menus pattern at the top (P5) in the original desktop UI were replaced to 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 items were 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).

### 7. Discussion

The types of cross-platform mapping that are recommended for some the most popular Web patterns and which can be used to redesign of a Web site for PDA. These transformations offer the closest and simplest equivalent in the corresponding platform. 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.

In this paper, we have introduced a pattern-oriented design method that essentially exploits pattern composition mapping. This approach is a significant improvement over non-structured migration methods currently in use, for the following reasons:

- The method provides a standardized table of pattern mapping, 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 by novice designers, as compared to reengineering which currently requires a considerable degree of expertise and abstract reasoning ability.

Pattern-assisted 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. 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 Web application between platforms.
8. A concluding remark

In this paper, we proposed and illustrated the POD approach, which aims to create a design for cross-platform application. Our discussion focused on the way to combine several types of patterns to create a reusable pattern-oriented design as well as mapping rules to translate this generic design into platforms-dependant Web applications. The proposed pattern taxonomy and the supporting composition and mapping rules aims to demonstrate when a pattern is applicable or required during the design process, how it can be reused, as well as how and why it can or cannot be combined with other related patterns and how it can be mapped. Using POD and the proposed taxonomy of patterns, cross-platform Web applications developers are provided with an effective tool: composition and mapping relationships and the underlying best practices to come up with reusable design solutions.

Our investigations are based on several years of Web applications development, validation and ethnographic interviews with Web developers, as well as suggestions from others. Such suggestions include reported best practices for using patterns as a bridge over the gaps between the design practices and software tools [11, 7]. Our experiences highlighted also that in order to render the patterns understandable 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 several types of patterns and a large collection of patterns require in-depth knowledge of both the problems and forces at play and most importantly must ultimately put forth battle-tested solutions. As such, it is inconceivable that pattern hierarchies 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 real-world problems. It is therefore essential to build an “academia-industry bridge” by establishing formal communication channels between industrial specialists in patterns, software design patterns, information architecture patterns as well as software pattern researchers. Such collaboration will lead, at to a common terminology which essential making the large diversity of patterns accessible to common Web engineering designers.

9. References


