PATTERN-ORIENTED ARCHITECTURE FOR WEB APPLICATIONS

M. Taleb, A. Seffah
Human-Centred Software Engineering Group Concordia University, Montreal, Quebec, Canada
Phone: +1 (514) 848 2424 ext 7165 and/or ext 3024
Fax: +1 (514) 848- 3028
mtaleb@encs.concordia.ca, Seffah@encs.concordia.ca

A. Abran
Software Engineering Department & Information Technology,
École de Technologie Supérieure (ÉTS), Montréal, Québec, Canada
Phone: +1 (514) 396-8632
Fax: +1 (541) 396-8405
aabran@ele.etsmtl.ca

Keywords: Design patterns, Pattern-oriented architecture, Software architecture and Web applications.

Abstract: A number of Web design problems continue to arise, such as: (1) decoupling the various aspects of Web applications (for example, business logic, the user interface, navigation and information architecture; and (2) isolating platform specifics from the concerns common to all Web applications. In the context of a proposal for a pattern-oriented architecture for Web applications, this paper identifies an extensive list of patterns aimed at providing a pool of proven solutions to these problems. The patterns span several levels of abstraction, from information architecture and interoperability patterns to navigation, interaction, visualization and presentation patterns. The proposed architecture will show how several individual patterns can be combined at different levels of abstraction into heterogeneous structures, which can be used as building blocks in the development of Web applications.

1. INTRODUCTION

The Internet and its languages offer major opportunities for developing a new generation architecture for Web software systems, the latest of which are highly interactive, platform-independent and run on the client Web browser across a network. This paper is aimed at providing a pool of proven solutions to many recurring Web design problems. Examples of such problems include: (1) decoupling the various aspects of Web applications such as business logic, the user interface, navigation and information architecture; and (2) isolating platform-specific issues from the concerns common to all Web applications.

In this paper, the definition of software architecture from (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996) is adopted: "*the structure of the subsystems and components of a software system and the relationships between them typically represented in different views to show the relevant functional and non functional properties*." This definition introduces both the main architectural elements (for instance, subsystems, components and connectors), and covers the ways in which to represent them, including both functional and non-functional requirements, by means of a set of views.

A pool of proven solutions is proposed here in the form of an architecture and the related patterns for a pattern-oriented architecture for Web applications to address solving these problems. These individual patterns can then be combined at different levels of abstraction into heterogeneous structures, which can be used as building blocks in the development of these applications.

This paper is organized as follows: section 2 introduces related work on pattern-oriented architectures in general, such as the Model-View-

Controller model (3-tier architecture), the Core J2EE pattern model (5-tier architecture) and the Zachman model (multi-tier architecture); section 3, based on Zachman's work, primarily describes the pattern-oriented architecture proposed here and some patterns which we have identified and formalized; finally, section 4 presents a summary and directions for future work.

2. RELATED WORK

2.1. MVC Model

The basic architecture we considered as a starting point is the Model-View-Controller (MVC) pattern, which is commonly used to structure Web applications that have significant processing requirements. This makes them easier to code and maintain. MVC is used here to describe the core components of Web application architectures, as it is a 3-tier architecture that is often used by Web application designers to maintain multiple views of the same data. At the design level, the MVC pattern features a clean separation of three types of objects:

- **Model:** for maintaining data;
- **View:** for displaying all or a portion of the data;
- **Controller:** for handling events that affect the model or view(s).

Other patterns may apply in the construction of these components. For example, in MVC, the views are tightly coupled with the control. Some authors have suggested using the "Command Action pattern" to ensure the separation of views and controls.

In Web applications design, several aspects need to be considered separately, including dialogs, persistence, site management and error handling. By itself alone on its own, the MVC architectural pattern is does not a sufficient solution that fully addresses these issues. Other patterns are also required to:

- Encourage the designer to consider other aspects of the dialog which are very important to the user, such as assistance or error management;
- Facilitate the use for the interface descriptions, which are highly important to the designer (Booch, Rumbaugh and Jacobson, 1999), (Myers, 1986), (Myers and Buxton, 1986) and (Meyer, 1990).

2.2. More advanced architecture: Modeling Core J2EE patterns

Building on the MVC pattern, the Java Sun team has proposed a 5-tier architecture (from Web site www.developpez.com) to model the Core J2EE Pattern Architecture (from Web site www.sun.com). Java also provides support for the implementation of the MVC architecture using the Observer Interface and the Observable Classes that together implement the observer pattern. The Observable Class represents an observable object, or "data" in the model-view paradigm. It can be "sub-classed" to represent an object that the application wants to have observed. An observable object can have one or more observers. An observer may be any object that implements the Observer Interface. The core J2EE Patterns-oriented Web software architecture proposed in Web site www.sun.com.

It can be observed that Web architecture needs to operate at six different levels, which are listed in Table 1.

Table 1: Six architectural levels of a Web architecture

Architectural Level	Function
1. Navigation	Provides proven techniques for navigation
2. Interaction	Provides dialog styles to perform tasks
3. Presentation	Provides solutions for how to visually organize the contents or the related services into working areas, the effective layout of multiple data and the relationship between them
4. Visualization	Provides different visual representations/metaphors for grouping and displaying a large set of data into cognitively accessible chunks
5. Interoperability	Provides mechanisms for decoupling the various layers of a Web application into particular information categories (content) and within the four higher levels listed above

6. Information	Provides conceptual models and architectures for organizing the underlying content across multiple pages, servers, databases and computers

To understand and define these levels in greater detail, we use the Zachman model, which is a multi-layer architectural framework.

2.3. Zachman Model as the basis for a multi-layer architecture

(Zachman, 1987) and (Sowa and Zachman., 1992) proposed a multi-tier architecture which was an Enterprise Architecture schema depicting two distinct dimensions in a matrix. The columns classify answers to questions such as What (Data), How (Function), Where (Network), Who (People), When (Time) and Why (Motivation). The rows classify the audience's perspectives: scope, owner, designers, builder, trades and functioning organization. This gives 36 cells that uniquely classify portions of the organization. The columns in the Zachman framework represent different areas of interest for each perspective and describe the dimensions of the systems development effort.

3. THE PROPOSED ARCHITECTURE

3.1. Overview

To tackle some of the weaknesses identified in related work, the Zachman theory, or set of concepts, proposes a 6-tier architecture of a pattern-oriented generic classification schema for Web software architecture. We use the matrix classification proposed by Zachman, according to which the columns constitute the questions and the rows represent the six levels defined in Table 2.

Table 2: Pattern-oriented generic classification schema for a Web software architecture

	WHAT (Data)	HOW (Function)	WHERE (Network)	WHO (People)	WHEN (Time)	WHY (Motivatio
Navigation	✓	✓		✓		✓
Interaction		✓	✓	✓	✓	✓
Presentation	✓	✓				✓
Visualization	✓	✓		✓		✓
Interoperability	✓	✓	✓	✓	✓	✓
Information	✓	✓	✓	✓	✓	✓

3.2. Pattern taxonomy

A taxonomy of patterns is proposed next. Examples of patterns are also presented to illustrate the need to combine several types of patterns to provide solutions to complex problems at the six architectural levels. This list is not exhaustive: there is no doubt that more patterns are needed, and that others have yet to be discovered.

A number of Web pattern languages have been suggested; for example, Van Duyne's The Design of Sites (Duyne, Landay, and Hong, 2003), Welie's Interaction Design Patterns (Welie, 1999) and Tidwell's UI Patterns and Techniques (Tidwell, 1997) play an important role, and specific languages, such as Laakso's User Interface Design Patterns (Laakso, 2003) and the UPADE Web Language (Engelberg and Seffah, 2002), have been proposed as well. Various specific pattern collections have been published, including patterns for Web page layout design (Tidwell, 1997), (Coram and Lee, 1998) and (Welie, 1999), for navigation around large information architectures, as well as for visualizing and presenting information.

In our work, we investigate how these existing collections of patterns can be used as building blocks within the context of the proposed six-layer architecture. Which patterns at which level solve which problem is the question we try to answer.

An informal survey conducted in 2004 by the HSCE Research Group at Concordia University identified at least six types of Web patterns that can be used to create a pattern-oriented Web software architecture. Table 3 illustrates these levels, and gives examples of patterns.

Table 3: Pattern-oriented taxonomy schema for a Web software architecture

Architectural Level and Category of Patterns	Examples of Patterns
Navigation **Navigation Patterns** This category of patterns implements proven techniques for navigating within and/or between a set of pages and chunks of information.	- Shortcut pattern - Breadcrumb pattern - Index Browsing pattern
Interaction **Interaction Patterns** This category of patterns focuses on the interaction mechanisms that can be used to achieve tasks and the visual effects they have on the scene, and, as such they relate primarily to graphical and rendering transforms.	- Search pattern - Executive Summary pattern
Presentation **Presentation Patterns** This category of patterns provides 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 pages, lists and tables	- Home Page pattern - List pattern - Table pattern
Visualization **Visualization Patterns** This category of patterns suggests different visual representations and metaphors for grouping and displaying information in cognitively accessible chunks. They mainly define the format and content of the visualization, i.e. the graphical scene, and, as such, relate primarily to data and mapping transforms.	- Favourite Collection pattern - Bookmark pattern - Frequently Visited Page pattern - Navigation Space Map pattern
Interoperability **Interoperability Patterns** This category of patterns is aimed at decoupling the layers of a Web application; in particular, between the content, the dialog and the views or presentation layers. These patterns are generally extensions of the Gamma design patterns, such as MVC (Model, View and Controller) observer and command action patterns. Communication and interoperability patterns are useful patterns to facilitate the mapping of design between platforms.	- Adapter pattern - Bridge pattern - Builder pattern - Decorator pattern - Façade pattern - Factory pattern - Method pattern - Mediator pattern - Memento pattern - Prototype pattern - Proxy pattern - Singleton pattern - State pattern - Strategy pattern - Visitor pattern
Information **Information 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 as which information can be or should be presented on which device	- Sequence pattern - Hierarchy pattern - Grid pattern

Some examples of proposed categories of patterns are presented below.

3.3. Information patterns

This category shows the need to combine several types of patterns to provide solutions to complex problems. Here again, the list of patterns is not exhaustive: there is no doubt that more patterns still need to be documented, and that others have yet to be discovered.

3.4. Navigation patterns

Navigation patterns are fundamental in Web design, since they help the user move easily and in a straightforward manner between information chunks and pages. They can obviously reduce the user's memory load (Nielsen, 1999) and (Lynch and Horton, 1999). See also (Tidwell, 1997), (Welie, 1999), (Engelberg and Seffah, 2002) and (Garrido., Rossi, and Schwabe, 1997) for an exhaustive list of navigation patterns.

3.5. Interaction patterns

A critical design issue for resource-constrained (small) devices is how long it takes to determine whether or not a document contains relevant information. The *search pattern* with the complicity of the "*Executive Summary pattern*" (a page-layout pattern), provides users with a preview of underlying information before spending time downloading, browsing and reading large amounts of information included in subsequent pages.

3.6. Visualization patterns

Information overload is another fundamental issue to tackle through Web software architecture. Web applications, especially large Web portals, can 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 the content of a large site. 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.

Information visualization patterns can be used to solve another complex design problem, which is to provide a comprehensive map for a large amount of content that cannot be reasonably presented in a single view. They are generally combined in such a way that the underlying content can be organized into distinct conceptual spaces or working surfaces which are semantically linked to one another.

3.7. Presentation patterns

The presentation patterns define the appearance and the form of presentation of the application on the Web page. These patterns provide solutions for how the contents or the related services can be visually organized into working surfaces, the effective layout of multiple information spaces and the relationship between them. They define the physical and logical layout suitable for specific Web pages such as home pages, lists and tables.

3.2.6. Interoperability pattern

The communication and interoperability patterns are useful for facilitating the mapping of a design between platforms. Examples of patterns that can be considered to ensure the interoperability of Web applications include all Web patterns of Interoperability patterns.

4. SUMMARIES AND FUTURE WORK

In this paper, we have identified and proposed six categories of patterns, providing examples, for a pattern-oriented architecture for Web applications to resolve many recurring Web design problems, examples of which include: (1) decoupling the various aspects of Web applications such business logic, the user interface, navigation and information architecture; (2) isolating platform-specific problems from the concerns common to all Web applications. Our discussion has focused on the way to specify a pattern-oriented architecture using particular patterns.

Future work will require the classification of each pattern and the illustration of each of them in UML class and sequence diagrams. Next, some relationships will have to be defined between patterns so that they can be combined to create models based on the resulting patterns.

5. REFERENCES

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, I., 1996. A System of Patterns: Pattern-Oriented Software Architecture. West Sussex, England, John Wiley & Sons

Zachman, John A., 1987. A Framework for Information Systems Architecture. IBM Systems Journal, vol. 26, no. 3, IBM Publication G321-5298

Sowa, J.F. and Zachman, John A., 1992. Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, vol. 31, no. 3. IBM Publication G321-5488

Architecture multi-tiers. Retrieved 2006, [Online] available at: http://java.developpez.com/archi_multi-tiers.pdf

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

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

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

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

Laakso, Sari A., 2003. Collection of User Interface Design Patterns University of Helsinki, Dept. of Computer Science, September 16, 2003. http://www.cs.helsinki.fi/u/salaakso/patterns/

Coram, T. and Lee, J., 1998. Experiences – A Pattern Language for User Interface Design, at http://www.maplefish.com/todd/papers/experiences

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

Nielsen, J., 1999. Designing Web Usability: The Practice of Simplicity. New Riders

Garrido, A., Rossi, G. and Schwabe, D., 1997. 'Pattern Systems for Hypermedia', Pattern Language of Programming Conference

Booch, G., Rumbaugh, J. and Jacobson, I., 1999. *The Unified Modeling Language User Guide*, Addison-Wesley

Myers, B. A., 1986. Visual programming, programming by example, and program visualization: A taxonomy. In Proceedings of the ACM CHI'86 Conference on Human Factors in Computing Systems; ACM New York, pp. 271-278; April 1986

Myers, B. A. and Buxton, W., 1986. Creating highly-interactive and graphical user interfaces by demonstration, International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 13th annual conference on Computer graphics and interactive techniques, Pages: 249 – 258

Meyer, B., 1990. Conception et programmation par objets pour du logiciel de qualité, Inter-Éditions, Paris

Core J2EE Patterns, Retrieved 2006, [Online] available at:http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html