Pattern-Oriented Architecture for Web Applications

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

A. Seffah
Human-Centred Software Engineering Group Concordia University, Montreal, Quebec, Canada
Phone: +1 (514) 848 2424 ext 3024
Fax: +1 (514) 848-3028
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

Abstract

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

Keywords
Design Patterns, Pattern-Oriented Architecture, Software architecture, Web applications.

1. Introduction

The Internet and its languages offer major opportunities to develop a new generation of Web software systems architecture. These new Web software systems are highly interactive, platform-independent and run on the client Web browser across a network. This paper aims to provide a pool of proven solutions to many recurring Web design problems. Examples of such problems include: (1) decoupling the different aspects of Web applications such as the business logic, the user interface, the navigation and information architecture, (2) isolating platform specifics from the common concerns to all Web applications.

Within this paper, the definition of software architecture from [1] 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 architecture elements (for instance: subsystems, components, and connectors), how to represent them by means of a set of different views and to represent both functional and non-functional requirements.

To address these issues, a proposed architecture and the related list of patterns aim to provide a pool of proven solutions. This paper proposes a list of patterns for a pattern-oriented architecture

for Web application. These individual patterns could then be combined at different levels of abstraction into heterogeneous structures that can be used as building blocks in the development of Web applications.

This paper is divided as follows: section 2 introduces related work on pattern-oriented architecture in general such as MVC model (3-tier architecture), model Core J2EE patterns (5-tier architecture) and Zachman model (multi-tier architecture); section 3, based on Zachman work, describes primarily the proposed pattern-oriented architecture and some patterns which we identified and formalized; finally, section 4 presents a summary and future work.

2. Related Work

2.1. MVC model

The basic architecture we considered as a starting point is the MVC pattern. MVC pattern 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 applications architecture whereas MVC is seen as 3-tier architecture that often is used by Web applications designers to maintain multiple views of the same data. At the design level, the MVC pattern hinges on 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).

This MVC pattern is illustrated in the UML class diagram in Figure 1. Other patterns may apply in the construction of these components. For example, in MVC the views are tightly coupled with the control. Some authors suggested using C*ommand Action* pattern to ensure the separation between views and controls.

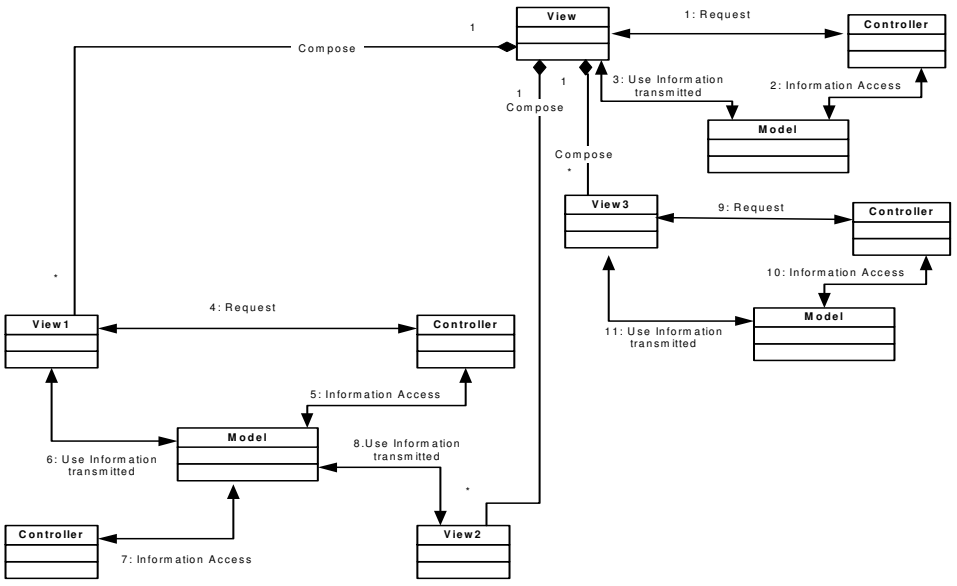

Figure 1: Class Diagram of Model-View-Controller architectural pattern [18]

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

- Encourage the designer to consider other aspects of the dialogue which are very important for the user, such as assistance or error management;
- Facilitate the use for the interface descriptions whereas they are of great importance to the designer [14, 15, 16, 17].

2.2. More Advanced Architecture: Model Core J2EE patterns

Building on the MVC pattern, the Java Sun team proposed a 5-tier architecture [4] to model Core J2EE Patterns Architecture [18] - illustrated in table 1. Java also provides support for the implementation of the Model-View-Controller architecture using the Observer Interface and the Observable classes that together implement the observer pattern. The class Observable 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 interface Observer.

Architectural Level	Description
Browser	This part is very often non-representative of the architecture but it is not excluded that this one contains applicative parts commonly called "Tests of first level". The test of first level consists mainly in the checking of the contents of the capture forms. They enable to ensure that all mandatory fields was indicated well for example. However, this series of tests MUST form part of the layer of presentation. Indeed, it is not excluded that the end-user decides to de-activate the JavaScript functionalities of his browser. Another use of this layer is the representation of the dynamic pages, among others, with DHTML format.
Presentation	This level deals with the logic of the navigation. It often implements JSP/Servlets technologies.
Logical Subject	Implemented in the form of Java Beans or EJB, it is in this layer that we find the whole of the treatments of an application.
Middleware	This part of the architecture covers connections with the other patterns of the same level or the composed patterns of different levels of patterns.
Persistence	It is often composed of one or several types of patterns.

Table 1: Summary of Patterns Architecture, adapted from [4]

An example of a UML class diagram of J2EE Patterns is presented in Figure 2

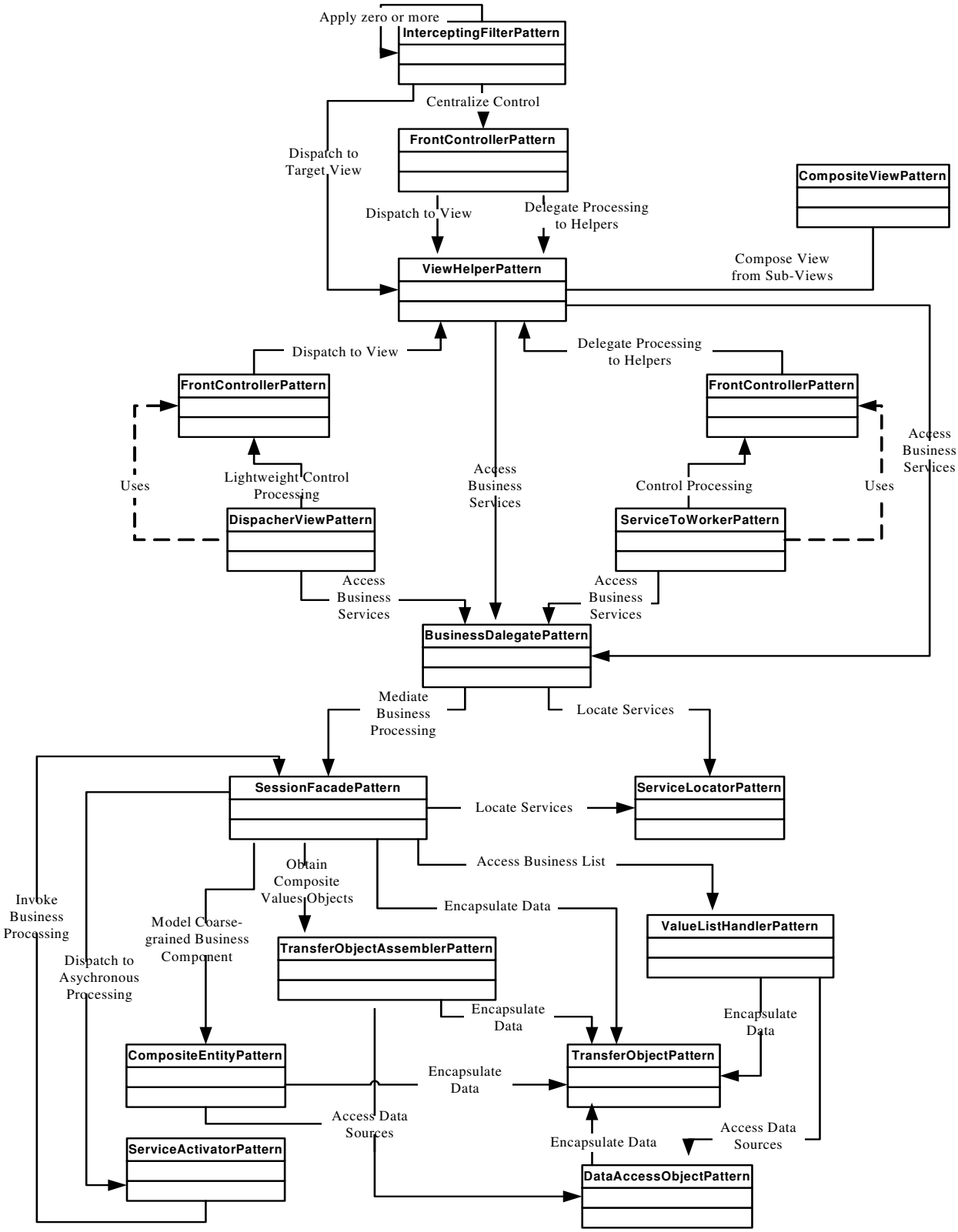

Figure 2: Core J2EE Patterns Architecture [18]

Table 2 summarises the core J2EE patterns-oriented Web software architecture proposed in [18].

Pattern Name	Function
Business Delegate	Reduce coupling between Web and Enterprise Java Beans tiers
Composite Entity	Model a network of related business entities
Composite View	Separately manage layout and content of multiple composed views
Data Access Object (DAO)	Abstract and encapsulate data access mechanisms
Fast Lane Reader	Improve read performance of tabular data
Front Controller	Centralize application request processing
Intercepting Filter	Pre- and post-process application requests
Model-View-Controller	Decouple data representation, application behaviour, and presentation
Service Locator	Simplify client access to enterprise business services
Session Facade	Coordinate operations between multiple business objects in a workflow
Transfer Object	Transfer business data between tiers
Value List Handler	Efficiently iterate a virtual list
View Helper	Simplify access to model state and data access logic

Table 2: Summary of Core J2EE Patterns Architecture [18]

It can be observed that Web architecture need to represent at six different levels, which are listed in Table 3.

Architectural Level	Function
1. Navigation	Provides proven techniques for navigation.
2. Interaction	Provides dialog styles to achieve tasks.
3. Presentation	Provides solutions for how the contents or the related services are visually organized into working areas, the effective layout of multiple informations and the relationship between them.
4. Visualization	Provides different visual representations / metaphors for grouping and displaying large set of information into cognitively accessible chunks.
5. Interoperability	Provides mechanisms to decouple the different layers of a Web application in particular information (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.

Table 3: Six Architectural levels of a Web Architecture

To understand and define in more details these different levels, the Zachman model is used next, that is an architectural multi-layer framework.

2.3. Zachman model as a basis for a multi-layers architecture

Zachman [2, 3] proposed a Multi-tier architecture. He proposed an Enterprise Architecture schema in which he depicted 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' 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 - Table 4.

WHAT (Data)	Each of the rows in this column addresses the understanding of, and dealing with, an organization's data.
HOW (Function)	The rows in the function column describe the process of translating the mission of the organization into successively more detailed definitions of its operations.
WHERE (Network)	This column is concerned with the geographical distribution of the organization's activities.
WHO (People)	This column describes who is involved in the business and in the introduction of new technology.
WHEN (Time)	This column describes the effects of time on the organization.
WHY (Motivation)	As Zachman describes it, this is concerned with the translation of business goals and strategies into specific ends and means.

Table 4: Set of concepts in Zachman Framework

3. The proposed architecture

3.1. Overview

To tackle some of the weaknesses identified in the related work, the Zachman theory or set of concepts is used to propose a 6-tier architecture of a Patterns-Oriented generic classification schema for a Web Software Architecture. Therefore, we use the matrix classification proposed by Zachman where the columns are the questions and the rows represent the six levels defined in Table 5.

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

Table 5: Patterns-Oriented generic classification schema for Web Software Architecture.

3.2. Patterns 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 different architectural levels. The list of patterns is not exhaustive: there is no doubt that more patterns are needed, and that a number of others have yet to be discovered.

A number of Web pattern languages have been suggested; for example, Van Duyne's "The Design of Sites" [5], Welie's Interaction Design Patterns [6], and Tidwell's UI Patterns and Techniques [7] play an important role. In addition, specific languages such as Laakso's User Interface Design Patterns and the UPADE Web Language [8, 9] have been proposed as well. Different specific pattern collections have been published including patterns for Web page layout design [7, 10, 6], for navigation in 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 HSCE Research Group of Concordia University identified at least six types of Web patterns that can be used to create a Pattern-Oriented Web Software Architecture. Table 6 illustrates these levels, as well as examples of patterns.

Architectural Level	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 - Bread Crumb 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, 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 page, 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 Pattern** This category of patterns aims to decouple the different 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, command actions 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

Table 6: Patterns-Oriented taxonomy schema for Web Software Architecture

Some examples of proposed patterns are presented next.

3.3. Information Patterns

The following examples show 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 a number of others have yet to be discovered. Table 7 provides examples of Information patterns.

Pattern Name	**Description**
Sequence pattern	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.
Hierarchy pattern	Are particularly well suited to Web application content, because Web sites should always be organized as offshoots of a single Home Page [11].
Grid pattern	Organizes the best way to present the content of information of a complex Web application.

Table 7: Information Patterns

3.4. Navigation Patterns

The 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 [12, 11]. See also [7, 6, 8, 13] for an exhaustive list of navigation patterns. Table 8 presents a list of navigation patterns.

Name of Pattern	**Description**
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 since the user accessed 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. For every item that is presented to the user, a navigation widget allows the user to choose the next or previous item in the list. The ordering criterion should be visible (and be user-configurable). To support orientation, the current item number and total number of items should be clearly visible,

Table 8: Navigation Patterns

3.5. Interaction Patterns

A critical design issue for resource-constrained (small) devices is how long does it take to determine if a document has 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. Table 9 presents a list of interaction patterns.

Pattern Name	Description
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
Search Pattern	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. Make your search box at least 25 characters wide, so it can accommodate multiple words without obscuring parts of the user's query

Table 9: Interaction Patterns

3.6. Visualization Patterns

Information overload is another fundamental issue to tackle through a 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 a large Web site 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.

These patterns are some of the information visualization patterns for solving another complex design problem. These patterns are generally composed to provide a comprehensive map to a large amount of content that cannot be reasonably presented in a single view. The underlying content can be organized into distinct conceptual spaces or working surfaces, which are semantically linked to each other. Table 10 presents a list of visualization patterns

Pattern Name	Description
Collections Favorites pattern	This pattern saves all desired Web sites visited by the users on Microsoft Internet Explorer.
Bookmark pattern	This pattern saves all desired Web sites visited by the users on Netscape browser.
Frequently Visited Page pattern	Users will often remember good articles, products, or promotions. However, when these are not featured on the home page, the users do not know how to find them. To help users locate key items, keep a short list of recent features on the home site, and supplement it with a link to a permanent archive.
Navigable Spaces Map pattern	This information visualization pattern summarizes the structure of the underlying content architecture.

Table 10: Visualization Patterns

3.7. Presentation Patterns

The presentation patterns define the appearance and the form of presentation on the Web pages of the applications. All 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, and tables. Table 11 presents a list of Presentation patterns.

Name of pattern	Description
Home Page pattern	This pattern defines the layout of home page Web site. This pattern is very important because the home page is the Web site face to the world and the starting point for most user visits.
List pattern	This pattern displays the information using forms.
Table pattern	This pattern displays the information in tables.

Table 11: Presentation Patterns

3.2.6. Interoperability Patterns

The communication and interoperability 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 all Web patterns of Interoperability Patterns. Table 12 presents some examples of Interoperability Patterns.

Pattern Name	Description
Adapter	Converts the interface of a class into another interface clients expect. An adapter lets classes work together that could not otherwise because of incompatible interfaces.
Bridge	Decouples an abstraction from its implementation so that the two can vary independently.
Builder	Separate the construction of a complex object from its representation so that the same construction process can create different representations.
Decorator	Attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub-classing for extending functionality.
façade	Provides a unified interface to a set of interfaces in a subsystem. Façade defines a higher-level interface that makes the subsystem easier to use.
Factory	Provides an interface for creating families of related or dependent objects without specifying their concrete classes.
Method	Defines an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.
Mediator	Defines an object that encapsulates how a set of objects interacts. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
Memento	Without violating encapsulation, captures and externalizes an object's internal state so that the object can be restored to this state later.
Prototype	Specifies the kind of objects to create using a prototypical instance, and create new objects by copying this prototype.
Proxy	Provides a surrogate or placeholder for another object to control access to it.
Singleton	Ensures a class has only one instance and provide a global point of access to it.
State	Allows an object to alter its behaviours when its internal state changes. The object will appear to change its class.
Strategy	Defines a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.
Visitor	Represents an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.

Table 12: Interoperability Patterns

4. Summary and Future Work

In this paper, we have identified and proposed six categories of patterns, together with examples, for a pattern-oriented architecture for Web applications to resolve many recurring Web design problems. Examples of such problems include: (1) decoupling the different aspects of Web applications such business logic, the user interface, the navigation, and information architecture, (2) isolating platform specifics from the common concerns to all Web applications. Therefore, our discussion focused on the way to specify a Pattern-Oriented Architecture using particularly patterns.

Future work requires the classification of each pattern and the illustration of each one of them in UML class and sequence diagrams. Next, some relationships must be defined between patterns in order to compose them together to create some models based on composed patterns.

5. References

[1] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996). A System of Patterns: Pattern-Oriented Software Architecture. West Sussex, England, John Wiley & Sons.

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

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

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

[5] 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.

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

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

[8] 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.

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

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

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

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

[13] Garrido, A., Rossi, G., and Schwabe, D. Pattern Systems for Hypermedia," Pattern Language of Programming Conference, 1997.

[14] Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

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

[16] B. A. Myers & W. Buxton, Creating highly-interactive and graphical user interfaces by demonstration.

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

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