# A Component-Oriented Framework for the Implementation of Navigational Design Patterns

Mohammed Abul Khayes Akanda and Daniel M. German

Department of Computer Science,
University of Victoria,
{makanda,dmgerman}@uvic.ca

**Abstract.** In this paper, we describe a framework for the implementation of Web applications based on navigational design patterns. This framework is being implemented as a collection of Java classes that can be easily parameterized or inherited in order to instantiate a given design pattern.

**Keywords:** Design Patterns, navigational design, reuse, components.

## 1 Introduction

A design pattern "describes a problem which occurs over and over again and then describes a solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [1]. A design pattern attempts to collect experience from the expert to pass on to other experts or novices in the field, hence avoiding reinvention by others. The navigational structure of a Web application can be improved using navigational design patterns. For example, the pattern *Set-Based Navigation* appears in almost any application. A generalized implementation of this pattern can therefore be instantiated over and over again.

## 2 A Framework for the Implementation of Navigational Design Patterns

We are in the process of building a framework for the implementation of a set of navigational design patterns in Java, using the Apache Tomcat server. Our framework implements the following design patterns, which we considered to be some of the most commonly used: *Active Reference, Set-Based Navigation, Landmark, News, Shopping Basket, History, Guided Tour, Selectable Search Space* [2]. The architecture of the framework is depicted in figure 1. The three main components are: 1) Client (presentation tier) which is responsible of creating the HTML pages that are sent to the client; 2) Server (middle tier) that processes the request from the client, and it is responsible of calling the corresponding Java class that implements the design pattern; and 3) data tier which contains the application database and the parameterization for the instanstiation of the patterns.

J.M. Cueva Lovelle et al. (Eds.): ICWE 2003, LNCS 2722, pp. 449–450, 2003.

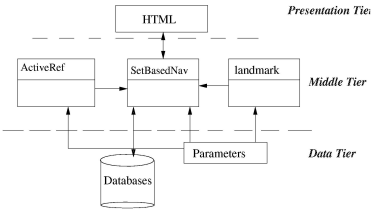


**Fig. 1.** The architecture of the framework

## 3 Reuse

The main objective of our framework is to achieve Alexander's goal of using "this solution a million times over, without ever doing it the same way twice" [1]. We do it by using two different approaches: parameterization, and inheritance.

**Parameterization.** Any design pattern can be parameterized via a two configuration files, one for navigation: *Navigation Parameters*, and another for presentation: *Presentation Parameters*. The *Navigation Parameters* describe the type of navigational object that the pattern is going to be used upon. In its simplest instance, it describes how the pattern instance should retrieve the data from a given database. The *Presentation Parameters* determine how the pattern will look when it is instantiated. For instance, for set-based navigation, it specifies how the table of contents of the set is created and displayed, for each element (e.g. a book) , which of its attributes are shown (title, authors, year of publication) and typeset (e.g. font type, size, color, etc). This parameterization is done via XML documents and XSL and CSS style-sheets.

**Inheritance**. Further configuration can be achieved by creating a subclass of the design pattern, in order to achieve any special behavior.

## 4 Current and Further Work

We are actively developing our framework. Our goal is to provide a library of java servlets that is ready to be used, and, if necessary, further extended (using inheritance and parameterization) to implement features not available in our implementation. We are also interested in creating actual Web applications so we can evaluate, first, their effectiveness (how useful they are), and second, their practicality (how easy to implement they are). Patterns are gaining popularity for hypermedia applications and a reusable implementation of those patterns will increase their use.

## References

1. Christopher Alexander, Sara Ishakawa, and Murray Silverstein. *A Pattern Language*. Oxford University Press, New York, 1977.
2. D. M. German and D. D. Cowan. Towards a unified catalog of hypermedia design patterns. In *Proceedings of the 33th Hawaii International Conference on System Sciences*, Jan. 2000.