Chapter 7
Design Patterns Support for RIAs Development

ABSTRACT

In Chapter 7, a review of Design Patterns proposed by GOF classification is presented. GOF classifies Design Patterns in Creational Patterns, Structural Patterns, and Behavioral Patterns. Some implementation examples of GoF Design Patterns are presented by using both JavaScript-based frameworks and non-JavaScript-based frameworks for RIAs development. Additionally, the chapter also presents a comparative analysis that summarizes the review of the capabilities of the RIA frameworks in the context of GoF Design Patterns support between JavaScript-based RIA frameworks and non-JavaScript-based RIA frameworks. Finally, this chapter also mentions how to solve different programming problems by using Design Patterns. Throughout the chapter, GOF Design Patterns and their classification are used.

1. INTRODUCTION

Design Patterns generally arise by the need of solving problems of software development in a systematic way. A Design Pattern is a formal description of a problem and its solution. Design Patterns must have a simple and descriptive name that can be readily used when referring to the pattern. A pattern should document the problem, its solution and the consequences of using it. Design Patterns can be used to assist in solving related problems to software development. From this perspective, a Design Pattern is a reusable solution that can be applied to common problems in software design.

A Design Pattern systematically names, motivates, and explains a general design that addresses a recurring problem in the design of object-oriented systems. It describes the problem, the solution, in the moment of applying the solution and its consequences. The solution is a general agreement of the objects and classes that solve the problem. The solution is customized to solve the problem in a particular context. (Gamma et al., 1994)

A well-known Design Pattern on Web applications development is MVC (Model-View-Controller). The MVC pattern separates the domain modeling, the presentation and the actions based on the user input into three classes: 1) Model, 2) View and 3) Controller. MVC pattern is a funda-
mental Design Pattern for the separation of user interface logic from business logic. Fortunately, the emergence of Web applications has helped to solve some of the ambiguity because the separation between the view and the controller is apparent.

In Software Engineering field, the Design Patterns were classified by the GoF in three types: 1) Creational patterns, 2) Structural patterns, 3) Behavioral patterns.

Creational Design Patterns are Design Patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. All the creational patterns define the best possible way in which an object can be created considering reuse and changeability. These describe the best way to handle instantiation (Christiansson et al., 2008).

Structural Design Patterns are Design Patterns that ease the design by identifying a simple way to realize relationships among entities. These patterns describe how objects and classes can be combined to form structures. It is important to distinguish between object patterns and class patterns. The difference is that class patterns describe relationships and structures with the help of inheritance. While object patterns describe how objects can be associated and aggregated to form larger, more complex structures (Christiansson et al., 2008).

Behavioral Design Patterns are Design Patterns that identify common communication patterns between objects and realize these patterns. These patterns are focused on the interactions among cooperating objects. The interactions between cooperating objects should be such that they are communicating while maintaining as loose coupling as possible. The loose coupling is the key to n-tier architectures. From this context, the implementation and the client should be loosely coupled in order to avoid hard-coding and dependencies (Christiansson et al., 2008).

The Design Patterns are very important on RIAs (Rich Internet Applications) development. In particular, Design Patterns are employed on frameworks for RIA development. In this chapter, a review and analysis about the Design Patterns support on JavaScript-based frameworks and non-JavaScript-based frameworks for RIAs development are presented. It should be noted that the GoF classification was used for this review and analysis.

2. DESIGN PATTERNS

Design Patterns are classified into 3 categories according to GoF, Creational, Structural and Behavioral Patterns. In the following subsections, the description and the UML class diagram representing the structure of each Design Pattern are presented.

2.1. Creational Patterns

Examples of Design Patterns located in this category are Abstract Factory, Builder, Factory Method, Prototype and Singleton patterns. These design patterns are described below in detail.

Abstract Factory

The abstract factory pattern is a software Design Pattern that provides a way to encapsulate a group of individual factories that have a common theme. This pattern separates the details of implementation of a set of objects from their general usage. Factories and products are the key elements into Abstract Factory pattern. This pattern captures how to create families of related objects without carry out the instantiation process of classes. It is most appropriate when the number and general kinds of product objects stay constant, and there are differences in specific product families (Gamma et al., 1994).

A factory is the location or a concrete class in the code at which objects are constructed. The intent in employing the pattern is to insulate the creation of objects from their usage. This allows