Using Design Pattern for Micro Level Software Design
S. A. M. Rizvi, Syed Imtiyaz Hassan
Department of Computer Science, Jamia Millia Islamia University, New Delhi.
Department of Computer Science, Hamdard University, New Delhi.

Abstract: Several problems can arise while developing an interactive or web based systems containing mixture of data access, business logic, and presentation code. Such applications are difficult to maintain, because interdependencies between all of the components cause strong ripple effects whenever a change is made anywhere. High coupling makes classes difficult or impossible to reuse because they depend on so many other classes. While developing such system, one should think beyond traditional, structured or object oriented methods as they offer limited means of reasoning about the validity of specifications. Finding the right abstractions of classes and defining the collaboration between them will be easier if design pattern is integrated with object oriented method of software engineering. With the help of pattern a system can be designed for change. Even though the use of a design pattern increases the reusability of the software, it is not practiced to its strength, largely because of the unfamiliarity of the benefits a design pattern offers. The present paper demonstrates how an interactive or web based system (including Web2.0) can be developed with the help of a well accepted design pattern, Model-View-Controller (MVC), that has been used in popular frameworks like Java Swing, Struts, spring and .Net.

Keywords: Design Pattern, Model-View-Controller (MVC), Web Application, User Interface Design.

1. Introduction

The modern world relies more and more on software. Customer requirements for new software products get more demanding all the time. On the other hand software is used in critical areas, where the quality has the highest priority. The competitive market causes pressure to produce software faster, cheaper and better. That can be done by improving the efficiency of software production and increasing the amount of reuse. Object oriented design and programming solve some problems of efficiency and reuse, but achieving best results requires planned and purposeful adoption of reuse in software development processes. The increasing complexity and size of software systems require sound engineering principles and design pattern [1] to formally structure the design process into multiple but consistent perspectives. In the absence of an explicit architectural framework and sound design pattern, the system is difficult to maintain and cannot evolve adequately. A design pattern is not all about program proving, rather, it works largely by making one think very hard about the system one propose to build [2]. A design pattern can be viewed as micro-architecture within a broad architectural picture which describes a particular recurring design problem, proposes a predefined scheme for its solution, and includes heuristic rules for how and when to use it [3]. In other words, a pattern is a form of solution intended to address problem and the purpose of the pattern is to reduce the complexity of user interfaces for a large and complex information system [4]. The ability to compose user interface representations by composing component specification is an essential requirement for design pattern. In nutshell, design patterns are a key concept in the field of software architecture: they offer well-established solutions to architectural problems, help to document the architectural design decisions, facilitate communication between stakeholders through a common vocabulary, and describe the quality
attributes of a software system as forces. To mitigate the risk associated with creating a robust client tier, developers have produced several design patterns with varying degrees of success. The Model-View-Controller (MVC) remains one of the more enduring patterns [5]. The following sections discuss the origin and timeline of Object Oriented paradigm along with the MVC [6] design pattern.

2. Background

The idea of object-oriented software originated in Norway in the mid 1960s with Simula, an extension to the ALGOL programming language. The inventors of object-oriented programming conceived objects as representations of the entities that model a chosen domain, with each object encapsulating the state of that entity (i.e. its attributes, including any relationships to other objects) together with the behaviours associated with that entity and hence each object was seen as being self-contained and the attributes of an object were encapsulated with all the necessary behaviours [7]. One of the languages that embraced the concepts of Simula was Smalltalk. A project commenced at Xerox PARC in the early 70s with the aim of creating the quintessential dynamic object oriented language. This is one which allows new classes, objects and behaviour to be appended on the fly by representing the class hierarchy, objects and methods of a program as meta-data at run-time [8]. Simula was the first language which introduced the concept of Graphical User Interface (GUI) with object oriented systems. Design patterns are a recent software engineering problem-solving discipline that emerged from the object-oriented community. It has roots in many disciplines, including software engineering. Concept of software design patterns was first introduced by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, also known as Gang of Four (GoF) in the year 1995 with the book [1].

3. The MVC

The MVC pattern is one of the oldest design patterns and is also perhaps the best known. MVC was first introduced by Trygve Reenskaug, a Smalltalk developer at the Xerox Palo Alto Research Center in 1979, and helps to decouple data access and business logic from the manner in which it is displayed to the user. More precisely, MVC (refer Figure1) can be broken down into three elements:

- **A Model** contains the underlying data and methods of the application. This is often referred to as the business logic of the application.
- **A View** contains a representation of the data in the Model and concerned with output to the user. The View also displays the user interface components that receive input from users. If the Model data changes, the View must update its presentation as needed.
- **A Controller** connects the Model and View and coordinates activities between them. Based on user inputs, it determines which methods on the Model should be invoked and which View should display the updated data in the Model. The Controller is responsible for processing input from the user.

A major advantage of adopting the MVC pattern is flexibility in user interface. Look and feel of the user interface can be easily modified. Change or upgrade of the client side technology such as Flash and mobile devices can be implemented with minimal effort. Another advantage is maintainability of the system. In addition, MVC increases testability because components are
encapsulated into model, view, and controller.

Figure 1: MVC (Adapted From [6])

4. Role of MVC in Web Development

Since its development, MVC has proven to be an important design pattern for facilitating the development, debugging, and maintenance of systems. While it was originally intended to more easily develop GUI systems, it can be successfully applied to the development of client-server and web based systems as well. It has enjoyed considerable popularity in Java programs as Sun Microsystems has used it extensively for the Swing GUI toolkit, Servlets, and enterprise applications. The MVC architecture is not limited to Java; MVC pattern exist for many languages and systems including Python, PHP and Microsoft Active Server Page (ASP) application environment. Even the latest frameworks Struts, Spring and Microsoft’s .Net are based on this design pattern. In short, most of the frameworks for developing Web 1.0 applications are based on MVC.

The rapid growth of network, Internet technologies and Web 2.0 [9] has brought fundamental changes in the new generation of computer technology. Although there is no hard-and-fast definition of Web 2.0, it refers broad range of websites that encourage interaction and collaborative work. Users do not just consume content, but they create, produce, edit, and remix content as well [10]. Information can be gathered from multiple sources in real time and assembled on a single web page. Examples of Web 2.0 applications include blogs [11], wikis [12] and popular web sites such as Google Maps [13], FaceBook [14] and MySpace [15]. Even these Web 2.0 applications are based on MVC. In fact they have two layers of MVC; client-side MVC and server-side MVC. So, MVC design pattern is even in next generation web application.

5. Conclusion

Design pattern has proven to be very useful in industry and is applicable to projects using either traditional or object-oriented methodologies. Design pattern approach of software development provides a guide to develop factored systems that facilitate coding, debugging, and maintenance. Adherence to design pattern oriented approach of software engineering improve the flexibility, maintainability and scalability of the application greatly. As the appropriate design patterns not only help in finding the right abstractions, but also offer other benefits mentioned above, design pattern approach of engineering software is a need of hour.

6. References


