ANALYZING AND IMPROVING WEB APPLICATION QUALITY USING DESIGN PATTERNS

Shikha bhatia
Research fallow, M.Tech. Department of Computer Science
lovely professional university, Jalandhar

Mr. Harshpreet Singh
Assistant Professor, Department. of Computer Science
lovely professional university, Jalandhar

ABSTRACT

With the mounting demand of web applications, a number of issues allied to its quality have came in existence. In the meadow of web applications, it is very thorny to develop high quality web applications. A design pattern is a general repeatable solution to a generally stirring problem in software design. It should be noted that design pattern is not a finished product that can be directly transformed into source code. Rather design pattern is a depiction or template that describes how to find solution of a problem that can be used in many different situations. Past research has shown that design patterns greatly improved the execution speed of a software application. Design pattern are classified as creational design patterns, structural design pattern, behavioral design pattern, etc. MVC design pattern is very productive for architecting interactive software systems and web applications. This design pattern is partition-independent, because it is expressed in terms of an interactive application running in a single address space. We will design and analyze an algorithm by using MVC approach to improve the performance of web based application. The objective of our study will be to reduce one of the major object oriented features i.e. coupling between model and view segments of web based application. The implementation for the same will be done in by using .NET framework.

General Terms

Design pattern, MVC, Software.

Keywords

Design pattern, MVC, Execution Speed.

1. INTRODUCTION

Design pattern is one of the important concepts in the field of software engineering. In general the **design pattern** is a repeatable solution to a commonly occurring problem in software design. It should be noted that design pattern is an intermediate product and is not a final design that can be transformed directly into code. Technically design pattern is a description or templates that helps in find a solution to the problem that can be used in different situations.

Design patterns [1] are one of the emerging concepts in software engineering. Design pattern evolves from civil engineering. In general one can say that design pattern is a blueprint for solving a specific problem, allowing the benefits of an optimal solution to be carried forward to new implementations.

Traditionally it was very difficult to develop high quality web based application. But modern days provide enough support in the form of right development process, tools, procedures and users to make high quality web application.

The advantage of design patterns is that it can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation.

While working with design patterns, it becomes important to remember, Without a doubt the major principles of using design patterns is that the design pattern should be structured, allowing each one to be created from a template.

At the very least, a template should contain:

- Pattern name that must be short and descriptive
- Intent that defines and set the goal of the design pattern
- Participants defines the responsibilities of the classes in the pattern
- Motivation defines the need to use such design pattern
- Applicability : uses for this pattern;
- Structure is graphical in nature that display diagrams of the patterns classes
- Collaborations defines interfaces between the participants
- Consequences explain the trade-offs and forces that exist within the pattern if any.

By using the concept of MVC one is able to improve the performance of web application. The Model/view/Controller [2] (or MVC) design pattern is a useful way to architect interactive software systems that separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes.

Model. The model [4] manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).

View. The view manages the display of information.

Controller. The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

It is important to note that both the view and the controller [3] depend on the model. However, the model depends on neither the view nor the controller. This is one the key benefits of the separation. This separation allows the model to be built and tested independent of the visual presentation. The separation between view and controller is secondary in many rich-client applications, and, in fact, many user interface frameworks implement the roles as one object. In Web applications, on the other hand, the separation between view (the browser) and controller (the server-side components handling the HTTP request) is very well defined [5].

Model-View-Controller is a fundamental design pattern for the separation of user interface logic from business logic. Unfortunately, the popularity of the pattern has resulted in a number of faulty descriptions [7]. In particular, the term "controller" has been used to mean different things in different contexts. Fortunately, the advent of Web applications has helped resolve some of the ambiguity because the separation between the view and the controller is so apparent.

The performance of web based application is mainly improved by using one of the main features of MVC i.e. separation of concerns. It provides an isolation of the application's presentation layer that displays the data in the user interface, from the way the data is actually processed. In other words, it isolates the application's data from how the data is actually processed by the application's business logic layer. The biggest advantage of the MVC design pattern is that you have a nice isolation of these components/layers and you can change any one of them without the rest being affected.[7]

coupling is the most important factor to be considered because unnecessary coupling may make the system unstable and may cause reduction in the system's performance. Coupling is one the imperative gauge of interdependency. It is always preferred to have stumpy coupling between modules and lofty cohesion within a module. If a design model is highly coupled, the system is difficult to implement, to test and to maintain overtime so there is need to remove module So coupling is thought to be a desirable goal in software construction, leading to better values for external software qualities such as maintainability, reusability and so on. Past research has shown wonderful effects of using MVC design pattern in various application software. In this study we have use MVC design pattern to improve the performance of web application by reducing the dependency between two major modules in MVC approach i.e. modeling and view.

2. RELATED WORK

One of the most famous and most used is surely MVC [5, 6], which divides each application (or part of it) into three different fundamental elements and states the rules for linking them together. While MVC provides a highly organized and uniform decomposition of the tasks performed by an interactive desktop application, the components prescribed by the MVC are not agnostic of the web development environment. However, most of the web based descriptions give their own interpretation while implementing MVC for web based application. This often results in the misapplication of the pattern.

In the MVC pattern, due to the layered system, [7] each subsystem in accordance with the hierarchy to organize themselves to achieve the allocation and cooperation of development and developers can work in parallel. Such a system enables to loose coupling between layers, reduce workload of development and maintenance, but also lay a solid foundation for the future upgrade.

Disadvantages of MVC

Although the MVC pattern well allocates the task, but there are some problems, mainly include:

1. The tight coupling between view, controller and model. View and Controller components are direct call model. It shows that the changing data of model need to be consistent with the view, thus increase coupling degree of data information and displayed pages.

2. In the model tier, tightly coupled with the business components that interact with the different roles

Rizvi et al. proposed an evolution of MVC pattern called MVC-web [5]. MVC-web that can be a way to reach loose coupling among different components and can also be used as a general guideline while implementing MVC for web based application.

To reduce the indirect coupling between Model and View, a new component "Dispatcher" is introduced between View and Model as shown in Figure 1. The Dispatcher registers both Model and View for event notification purpose. Whenever Model causes an event due to the change of state, a notification is passed to the Dispatcher. The Dispatcher scans all its registered View and the appropriate View method is invoked and the View updates itself. There is another benefit of using Dispatcher. The burden of registering different listeners and its management is transferred to Dispatcher from the Controller, which implements a very complex logic for transfer of control.

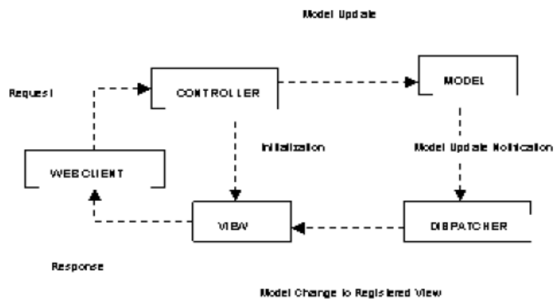


Figure-1[5]

Advantage of adopting this pattern is twofold. The coupling between Model and View is looser than the original one, due to the placement of Dispatcher between them and it serves as an implementation guideline for web development. Adherence to guidelines MVC-web will improve the flexibility, maintainability and scalability of web application greatly.

The main design patterns used by MVC-web are the Observer, Composite, and Strategy one [9]. The observer pattern is used to notify dependent objects about changes. The composite pattern enables to treat individual objects and compositions uniformly, such as views and composite views. The strategy pattern makes algorithms interchangeable, e.g. the controller class can be configured with one of many behaviors. In general the MVC-web pattern and its design patterns address design for change.

Layered architecture [7] that is the most accepted and the most broadly used design method of enterprise application software. This architecture is used to improve the system architecture and to build flexible and reusable software architecture.

There is a technology to improve the problems in MVC by use design patterns, consequently, weakening the coupling among layers, reducing the complexity of system development and providing a flexible framework for evolution of application system. The following is the improved MVC framework.

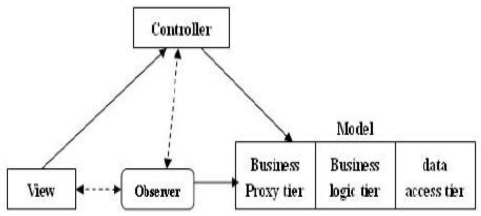


Figure-2 [7]

There is Observer pattern to solve the tight coupling between model and view with controller. The relationship between Model and View of MVC is the same as objective and observer. When the model data update, view and controller will be notified update at the same time. For example, in a Web system, view information update when the data changing. Such occasions, using this pattern can reduce the coupling degree between data information and displayed pages, separate the displayed pages and the data.

3. OBJECTIVE OF STUDY

Some of the objectives of this research are as given below:

- To comprehend the need and working of design patterns
- To realize the importance of MVC design pattern
- To devise an algorithm using MVC approach for web based applications to reduce coupling between modeling and viewing segments of web based application

4. PROBLEM DEFINTION

Research has exposed that by using the notion of design pattern in web application one is able to improve the performance because the design pattern can promote reusability and consistency of the web application. Without the adoption of design patterns or wrong selection of them will make the development of web application more complex and hard to be maintained.

In this paper we want to propose an algorithmic approach for using design pattern in web application so that the performance of web application can be improved. We will focus in providing MVC algorithm that reduce the coupling in modeling and viewing a web application.

In modern software applications that need to deal with the staging of a huge quantity of data to the end user, a software developer tries to make the web application loosely coupled via separating the data (model) from the user interface (view) concerns. To meet the same purpose MVC design patterns were introduced. The model-view-controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

Here is the list of the major advantages of this pattern.

- It provides a clean separation of concerns.
- It is easier to test code that implements this pattern.
- It promotes better code organization, extensibility, scalability and code re-use.
- It facilitates de-coupling the application's layers.

5. PROPOSED ALGORITHM

We have designed and implemented an MVC algorithm that will improve the performance of web application by separating model, view and control portion of web application. The following algorithm implements the concept of model/view/ controller. In the following algorithm there are three modules.

The first **modal** module read the web application then manages the data. It also Send request for reading state, process instruction and change the status of application state that means it generates an appropriate user interface.

The second **view** module display the information on the screen. It gets its own data from the model. The view is automatically notified by the model of changes in state that require a screen update.

The third **controller** module controls the mouse and keyboard event that means input that is given by the user and then converts the event into an appropriate user action, understandable for the model. The controller notifies the model of the user action, possibly resulting in a change in the model's state.

```
Procedure mvcModel()
Read webapplication
Manage data and its behavior
Send request for reading state, process instruction and change
the status of application state
    End mvcmodel()
    Proceduremvc view()
Set the view for displaying an information in an effective
manner
    End mvcview()
    Procedure mvccontroller()
        Get mouse events()
                If (event is vevent) then
                        Call view()
                Else
                        Call model()
                Endif
Get keyboard event()
If (event is vevent) then
Call view()
                Else
                        Call model()
                Endif
```

End mvccontroller

6. FLOW DIAGRAMS

There are three different procedures: model, view, and controller. Every module has its own working. The first flow diagram that is figure 3 represents the first module that means model module that read the web application then manages the data. It also Send request for reading state, process instruction and change the status of application state that means it generates an appropriate user interface.

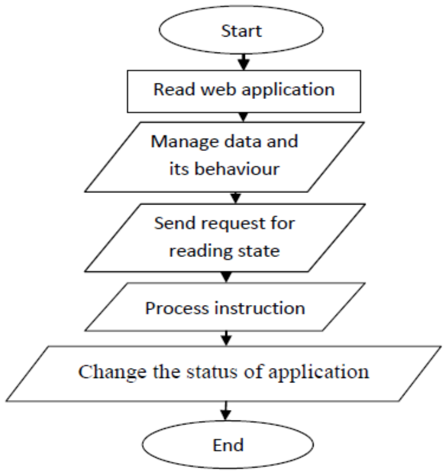

Start
Read web application
Manage data and its behaviour
Send request for reading state
Process instruction
Change the status of application
End

Figure-3

Second module, view displays the information on the screen. The figure 4 represents the view module that displays the information on the screen.

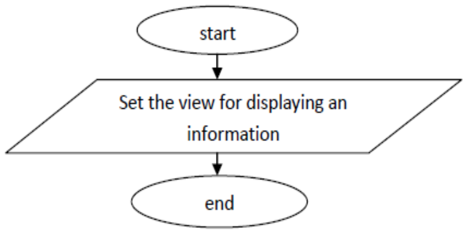


Figure-4

The figure 5 represents the controller module that controls the mouse and keyboard event that means input that is given by the user and then converts the event into an appropriate user action, comprehensible for the model.

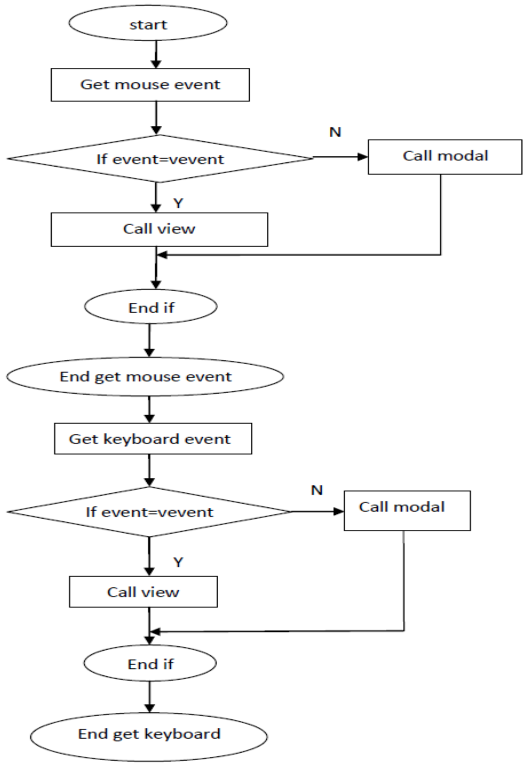
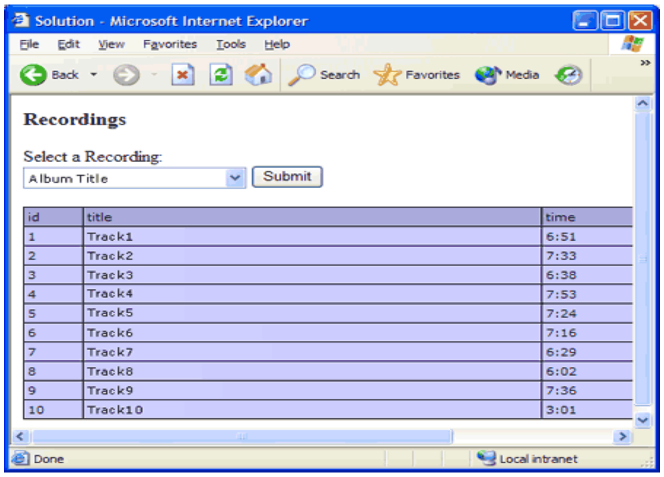


Figure-5

7. ANALYSI S

The following section of this paper will explain the practical implementation of Model-View-Controller pattern in ASP.NET. The following section shows how web application is implemented by separating the model, view, and controller roles in software. Since we have decompose the whole web application into three major modules called modeling, view and control, now the model module is independent of view module and vice versa, that gives us more flexibility in viewing a web application. By using the above said approach one is able to reduce the interdependency of modeling and view section of web application. The following web application example shows a drop-down list, which displays recordings that are stored in an organized manner i.e. in database. The practical implementation shows the improvement in the design of web application. The following .NET implementation shows the coupling between model and view is reduced, that otherwise is of great significance in traditional approach. The reduction in coupling obviously improved the performance of web application.


Solution - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Recordings
Select a Recording:
Album Title
Submit
Done
Local intranet

id	title	time
1	Track1	6:51
2	Track2	7:33
3	Track3	6:38
4	Track4	7:53
5	Track5	7:24
6	Track6	7:16
7	Track7	6:29
8	Track8	6:02
9	Track9	7:36
10	Track10	3:01

Figure -10 Example Web page

The user selects a specific recording from the drop-down list and then clicks the Submit button. The application then retrieves the list of all tracks from this recording from the database and displays the results in a table. All three solutions described in this pattern implement the exact same functionality.

8. CONCLUSIONS

One of the foremost causes of using design patterns is essentially parallel to those for using any object oriented techniques. Design patterns prop up reuse, without constraining implementations to a specific platform or language, and promote efficient use of design time. They can be seen as the design equivalent of modular or object oriented programming. Their strengths, however, can also be seen as their weakness. By not being implementations, the programmer is still required to actually code them, and as such any errors in the elucidation will be translated into the final source code. This means that different programmers may end up with different implementations of the same pattern, possibly even with different behaviors. The above study shows that use of MVC in web based application is explicit than the traditional one. One of the major advantages of adopting this pattern that the coupling between Model and View is looser than the original one. In general MVC-web will increase the flexibility, maintainability and scalability in web application.

9. FUTURE WORK

This work can be improved by optimizing several other metrics of web based application like redundancy in code, cohesion within module, memory consumption, and buffer used in communication etc. The system can be further improved by making this language independent.

10. REFERENCES

[1]http://ezinearticles.com/? Design-Patterns-for-Software-Engineering&id=550329 (online).

[2] G.E. Krasner and S.T. Pope, A Cookbook for Using the Model-View-Controller User-Interface Paradigm in Smalltalk-80, SICS Publication, 26-49, Journal of Object-Oriented Programming, August/September, 1988.

[3] Buschmann, F. et al, Pattern-Oriented Software Architecture: A System of Patterns, John Wiley and Sons, 1996, 123- 168.

[4] http://msdn.microsoft.com/en-us/library/ff649643.aspx (online)

[5] Dr. S.A.M Rizvi, Syad Imtiyaz Hassan, " Achieving Loose Coupling Between Different Components of Model-View-Controller For Web Based Application", Proceedings of the 3rd National Conference; INDIACom-2009 Computing For Nation Development, February 26 – 27, 2009 Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

[6] Harshad P. Prajapati, Vipul K. Dhabi, "High Quality Web-Application Development on Java EE Platform", IEEE International Advance Computing Conference (IACC 2009), Patiala, India, 6-7 March 2009.

[7] Chen Liyan et. al "Research on Framework Developing Technology based on MVC", Advances in Information Sciences and Service Sciences. Volume 3, Number 3, April 2011

[8] Morales-Chaparro et. al "MVC Web design patterns and Rich Internet Applications",2007

[9] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Company, 1995.