A Comparative Analysis on Modeling and Implementing with MVC Architecture

Tamal Dey
Assistant Professor, Department of MCA
PES Institute of Technology
100 Feet Ring Road, BSK- III Stage, Bangalore- 560085

ABSTRACT
A comparative study of a model, described based on the Model-View-Controller architecture paradigm which is built over the integration of the components and helps easy access of web services. MVC approach eases the horizontal development and maintenance of large scale distributed web applications in three frameworks. First is Big Blob framework, in this all the processing logics are part of GUI. Second is MVC architecture that easily works in command line or web interface. Third is Modified MVC Design. In implementation phase MVC architecture deals with J2EE and JSP-Servlet environment for web and internet programming. Also applying the MVC architecture with Multiple Frameworks gives privilege to work with few new concepts like Struts and spring components.

GENERAL TERMS
User Interfaces, Visualization and modeling.

KEYWORDS
GUI, COCA, GET, POST, HTTP, Struts, Spring, Model, View, Controller etc.

1. INTRODUCTION
Web has the very complex issues these days. Since the desire of the companies and organizations are increasing so the complexity and the performance of the web programming matters. Complexity with the different types of communication devices is increasing. The business is demanding applications using the web and many communication devices. So with the increase load of the data on the internet we have to take care of the architecture issue [6].

Literature related to the topic is to identify Big Blog, MVC and Modified MVC architectural structures merits and demerits and make a set of comparison analysis for the architectural models in terms of implementation of each aspect. Next implement the MVC architecture with JSP-Servlet programming that helps to the architecture with multiple framework.

2. RELATED WORK
Various frameworks have been mentioned in literatures which are discussed below.

2.1 Big Blob Architecture
A common style of programming is to put all processing in the GUI. One common structure is "*big blob*" use this structure. It is useful for the absolutely simplest programs. This works correctly as long as the "model", the logic, is so small that it isn't worth putting into a separate class [2].

2.2 Model View Controller Architecture
Programming with graphical user interface (GUI) libraries makes easier to implement the model-view-controller (MVC) design. 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 [Figure 1]. More precisely, MVC can be broken down into three elements:

2.2.1 Model
The model represents data and the rules that govern access to and updates of this data. In enterprise software, a model often serves as a software approximation of a real-world process [2].

2.2.2 View
The view renders the contents of a model. It specifies exactly how the model data should be presented. When the model data changes, the view must update its presentation as needed. This can be achieved by using a *push model*, in which the view registers itself with the model for change notifications, or a *pull model*, in which the view is responsible for calling the model when it needs to retrieve the most current data [2].

2.2.3 Controller
The controller translates the user's interactions with the view into actions that the model will perform. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in an enterprise web application, they appear as `GET` and `POST` HTTP requests. Depending on the context, a controller may also select a new view -- for example, a web page of results -- to present back to the user [2].

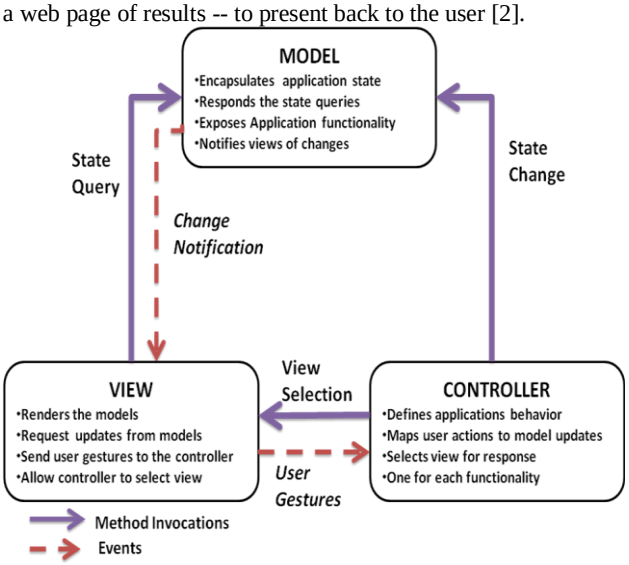

Fig 1: A Common MVC Implementation

2.3 Modified MVC Architecture
A more recent implementation of the MVC design places the controller between the model and the view. This design is common in the Apple Cocoa framework [Figure 2].

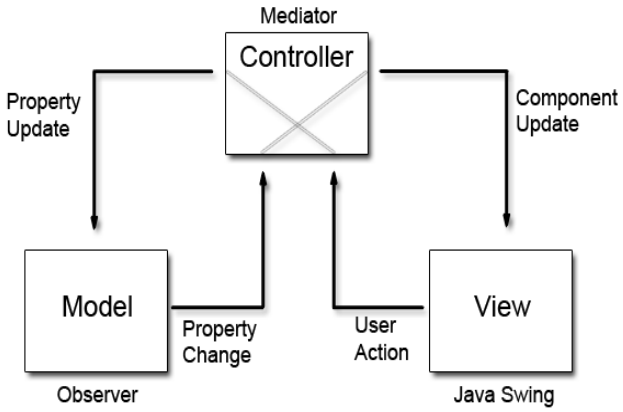

Fig 2: An MVC Design Placing the Controller between the Model and the View

The Cocoa and Cocoa Touch frameworks that power Mac OS X and iOS are tightly integrated into the Xcode development experience. Cocoa's high-level APIs make it easy to add animation, networking, and the native platform appearance and behavior to your application with only a few lines of code [6].

3. COMPARISION OF BIG-BLOB, MVC, MODIFIED MVC ARCHITECTURE

The *big blob* architectural programs are harder to read, maintain, and enhance. Paper can't really appreciate this when I start to build bigger programs. This works correctly as long as the "model", the logic, is so small that it isn't worth putting into a separate class. However, mixing model with presentation usually makes the program hard to read, and the inevitable growth of the program leads to a mess. This fails the simple Interface Independence test in web interface. Even changing model in any part also fails in this type of implementation [2].

The Model-View-Controller (MVC) architecture is easier to implement graphical user interface (GUI) libraries [2] because it provides a true decoupling of each part [3]. So it is easier to change the model part and view will notify the update the model does not carry a reference to the view but instead uses an event-notification model to notify interested parties of a change [Figure 3]. One of the consequences of this powerful design is that the many views can have the same underlying model. When a change in the data model occurs, each view is notified by a property change event and can update itself accordingly [3].

3.1 Solution with UI Model Architecture

Model View Controller structure can easily draw few possible solutions to implement the architecture. Firstly separate the user interface from the "model which makes huge improvement in simplicity, enhancements and maintenance are much easier with this structure [4]. Secondly model doesn't know about user interface. Lastly, model represent itself in text or graphically with the help of model class by override toString() method [5].

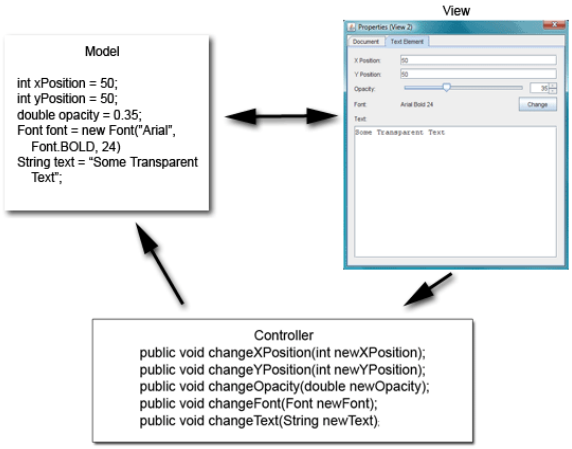

Fig 3: A Java SE Application Using MVC

The Modifying MVC architecture uses Cocoa Touch frameworks that power Mac OS X and iOS are tightly integrated into the Xcode development experience. Cocoa's high-level APIs make it easy to add animation, networking, and the native platform appearance and behavior to your application with only a few lines of code. In this Structure models encapsulate application data, Views display and edit that data, and Controllers mediate the logic between the two. By separating responsibilities in this manner, you end up with an application that is easier to design, implement, and maintain [3].

3.2 MVC architecture with JSP-Servlet environment

The MVC architecture was able to solve some of the problem of web and internet programming but still there were a lot of things missing from it. It was centred on the navigation of the JSP pages so there was the scope of the further development in the architecture point of view. During this process the next development was the Model 2 architecture. This problem was solved using the Servlet and JSP together. The Servlet handles the Initial request and partially process the data. It set up the beans then forward the result to the one of the JSP page. The Servlet decide the one of the page to be displayed from the list of pages [12].

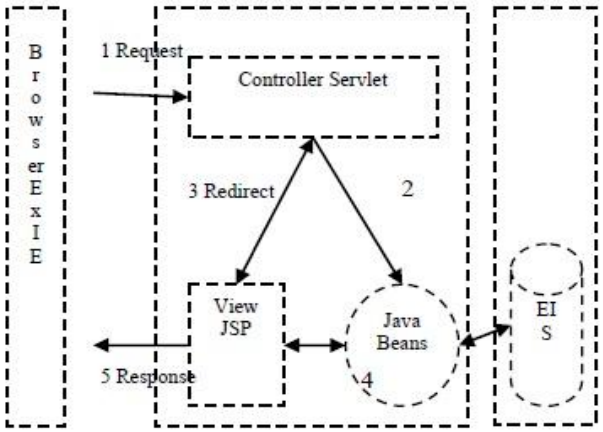

Fig 4: MVC architecture with JSP-Servlet

In this model all Control and application logic were handled by the Servlet. The Servlet was written in the java programming language. So it was also easier to handle the programming part of the Servlet. In this scenario the Servest becomes the power full for the complete application and it has emerged as the center point for the application. In the model architecture the Servlet

becomes the gatekeeper for the all common tasks. It provides the common services like authentication, authorization, error control and follow of the application. This architecture has solved the most of the problems. But still there were many new issues emerged while applying this architecture [11].

3.3 Applying architecture with Multiple Frameworks

Web and Internet is ever growing area and the demands for the applications are growing. A single framework is not capable to handle the architecture of the application. To meet the currents requirement of the applications it's necessary to design a architecture to implement the frameworks [4]. Struts framework have been designed and developed for the front end control of the web applications [Figure 5]. It provides the various features for the applications that interact to the users. It also follows the MVC design features. Spring Framework is the designed to handle the various tasks. The spring work for the desktop and internet based applications also. It follows the principals of the MVC. The simultaneous use of the Struts and spring frameworks in the single application with the applying the MVC Design principals so that can improve the performance of the applications. Struts Framework consists of three major blocks, Described in brief as follows [9].

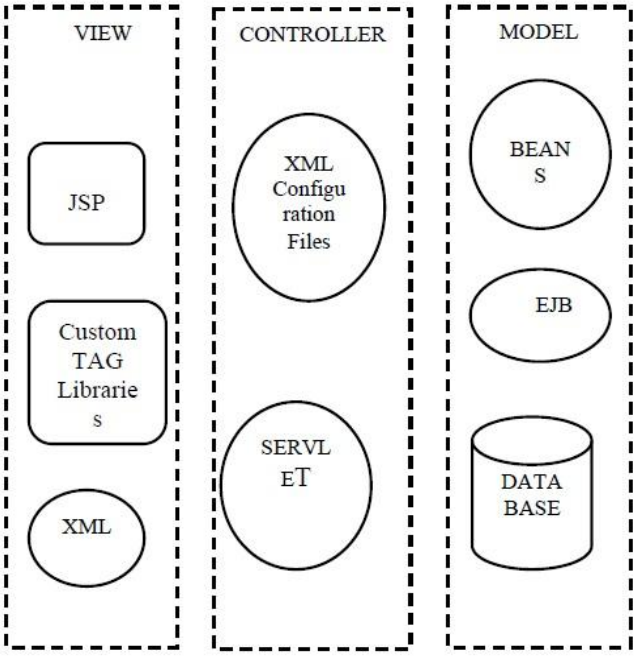

Fig 5: Struts model architecture

First is The View Block which controls the presentation part of the complete model. This contains following JSP files which you write for your specific application set of JSP custom tag libraries Resource files for internationalization [9].

Second Block is representing the Controller. This is for navigation the complete application. This contains XML configuration files; it contains the tags for the navigation of the paths [9].

Third Block is the Model. This part does the work of the Business Logic, Fetching and storing data to the database. This contains following Java Beans Enterprise Java Beans Database. Following figure shows the working of the components in the struts framework [9].

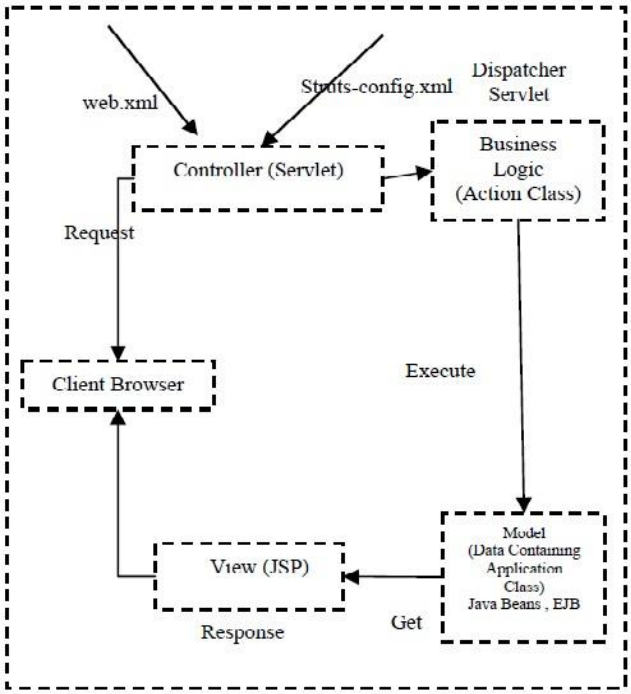

Fig 6: Component in Struts architecture

3.3.1 MVC in Struts

The major three parts of the MVC are as follows in the spring framework. Servlet controller (Controller Part) Java Server Pages or any other presentation technology (View Part) Application Business Logic: in the form of whatever suits the application (Model Part) [7].

Struts are a framework that implements a powerful and flexible controller based on the *Service To Worker* pattern. Struts' main advantages are: Integration flexibility: Struts' architecture provides flexibility for choosing the view and the model to be used. The view is based on the plug-ins concept. A plug-in is a dynamic mechanism by means of which a component or set of components that implement certain functionality in our application can be replaced by another ones, by simply modifying the application's configuration [Figure 7].

This model is implemented through JavaBeans, thus allowing its integration with other frameworks. It is supported by a solid community: Struts is a project from the Apache Software Foundation which has been consolidated as one of the most important organizations in the open source scope. In (Sing, 2002), SUN recommends using Struts as the framework for the Web tier [14].

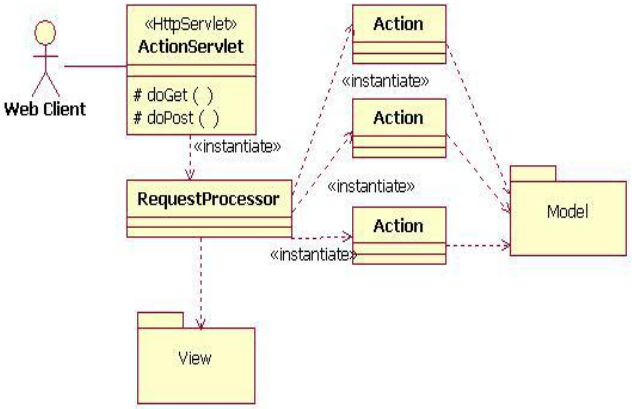

Fig 7: Struts's class diagram

Struts are composed of three main components:

1. The *ActionServlet* (the model's *Front Controller*), which is responsible for the application configuration and for receiving and analyzing the clients' requests. This component extracts from the configuration file (*strut-config.xml*) the general configuration parameters, the set of components that defines its behaviour (plug-ins) and the properties of each request. After performing these tasks, it delegates the control in the *RequestProcessor*.

2. The *RequestProcessor* (*Request Dispatcher* in the model), that creates an instance of the action (*Command* pattern) associated to the received request and executes it.

3. The *Action* (*Command* in the model). For eachoperation or use case, the developer creates an action (object) that inherits from the Action component. Each action is associated to a request type in Struts' configuration file.

3.3.2 Spring Components.

In the spring also follow the principals of the MVC. It has been designed more for the desktop and internet based applications. Spring consist of three core collaborating components [13].

1. Controller: Handles navigation logic and interacts with the Service tier for business logic.

2. Model: The contract between the Controller and the View Contains the data needed to render the View populated by the Controller.

3. View: Renders the response to the request Pulls data from the model. Core components in the spring MVC are as follows.

3.1 Dispatcher Servlet: Spring's Front Controller implementation.

It is the first controller which interacts to the requests that can also say it is an implementation of the Servlet. It controls the complete flow of the application.

3.2. Controller: User created component for handling requests encapsulates navigation logic delegates to the service objects for business logic.

3.3. View: Responsible for rendering output. Different views can be selected for the different types of output bases on the results and the viewing device, communication devices.

3.4. Model and View: It is the core part of the spring framework. It implements the business logic of the application. It is controlled by the controller. It stores the business logic and the view associated with it. Whenever it is executed it wills the data with the name of the view.

3.5. View Resolver: How the output is to be displayed depends on the result received from Model and View. It is used to map logical view names to actual view implementations. This part identifies and implement what is the output media and how to display it.

3.6. Handler Mapping: Strategy interface used by Dispatcher Servlet for mapping incoming requests to individual Controllers. It identifies the request and calls the respective handler to provide the services.

The following figure shows how the model will work. In this the dispatcher Servlet is the entry point for the application. The Struts parts do its work and send the request to the dispatcher Servlet [Figure 8]. The Servlet decides the handler. Then it will call to the controller. Controller will execute the Model and View [8].

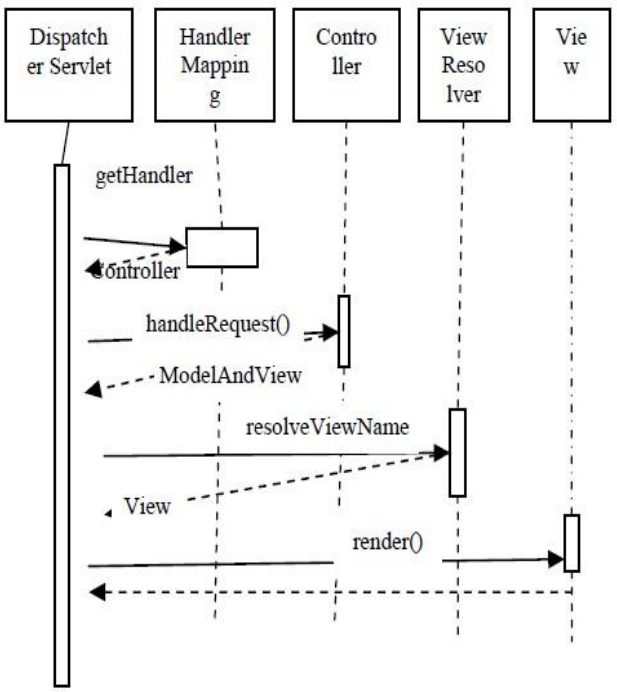

Fig 8: Sequence flow of application in the spring framework

4. PROPOSED METHODOLOGY

This approach is based a combination of applying the two frameworks struts and spring for application development scenario. The sequence diagram for the combined application is explained as above, which is the main driving force for the application. This approach assumes that basic knowledge of web applications is essential. A testing has been done for the above concepts and found successful work. Major benefits of the above architecture are as follows.

1. It will provide a very clean division between actions like action forms, controllers, handlers, JavaBeans models, and views.

2. Spring's MVC is very flexible. Unlike Struts, this forces your Action and Form objects into concrete inheritance by using advantage of both.

3. Spring MVC is entirely based on interfaces. Every part of the Spring MVC framework is configurable.

4. It provides controllers, making it easy to handling of many requests from User Interface.

5. JSP or any other technology can be used to display the view, results to the user on the any of the output device.

6. Spring Controllers are configured via Inversion of Controls. This makes them easy to test and integrated with other objects managed by spring.

7. Spring MVC web tiers are typically easier to test as compared to Struts web tiers, due to the avoidance of forced concrete inheritance and explicit dependence of controllers on the dispatcher Servlet.

8. Struts framework was designed for the web interface purpose only. The spring framework was developed for the desktop and internet applications. When both frameworks used as combined it will provide the flexibility of implementation.

9. Struts framework was designed for the web interface purpose only. The spring framework was developed for the desktop and internet applications. When both frameworks used as combined it will provide the flexibility of implementation.

5. IMPLEMENTATION

Different frameworks based on Model that facilitate the development of J2EE applications. Some of these are integrated with servers and tools specific of their corresponding J2EE providers. There are also open source frameworks supported by a solid community and widely spread in the last years. In order to select the most appropriate framework for that purposes and carried out a survey which focused on open source frameworks, due to its can improve the performance of the Large Database application in terms handling number of requests. Inexpensive costs and the technological maturity reached by some of them. A suitable framework must achieve the following two objectives: first, it must adapt to our model's specifications, and second it must be a good framework – as described in the background. After analyzing the most used, widely spread open source frameworks in the Java community (Struts, Cocoon, Maverick, SOFIA, Spring, WebWork, Tapestry, Turbina and JSF), it observed that none of these completely satisfied the established requirements. So it has been decided to use a different framework for each of the model's parts (the Model, the View and the Controller): It chose Struts as the Controller, Cocoon for the View and StrutsEJB for the Model. In the next point, It describe each of these frameworks and how they fit into our model.

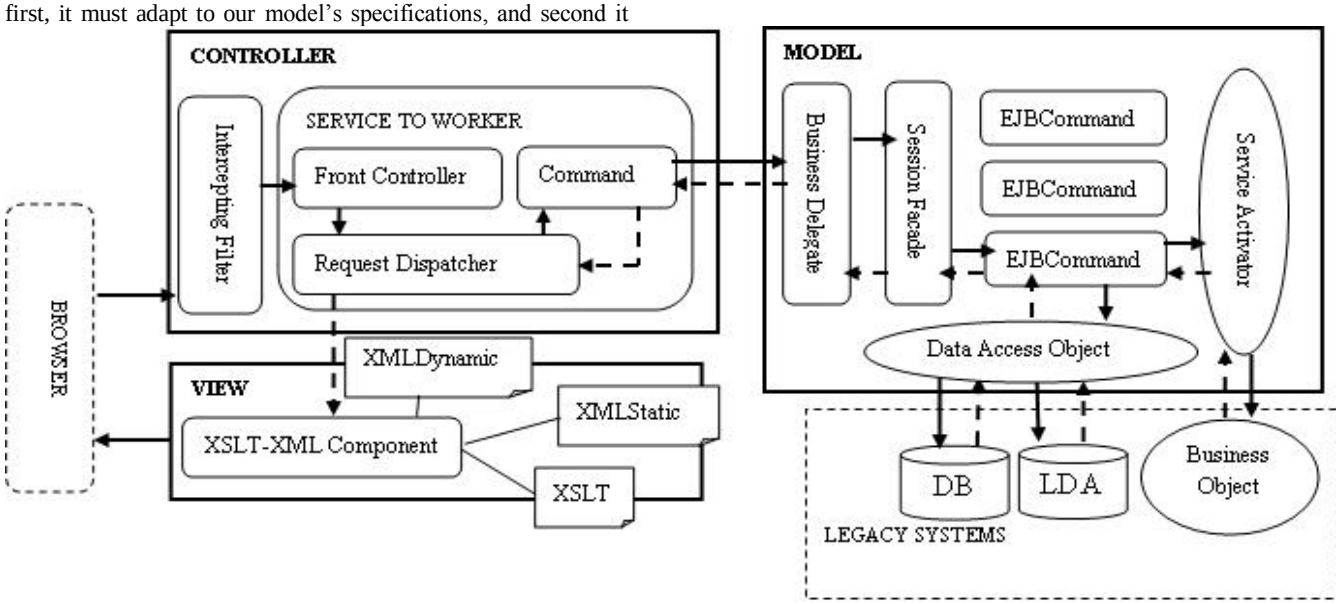

Fig 9: Model architecture

The primary target of the proposed model is to simplify the development of large applications based on the J2EE platform), thus providing a well structured architectural design, which allows for a complete decoupling of the system's main elements and synthesizes existing models, patterns and frameworks in the best way [14].

In this model, the controller serves as the application's entry point. It is implemented using only two patterns: the Intercepting Filter and the *Service To Worker*.

The *Intercepting Filter* is used in our model to implement the requests pre-processor; this system initially manages the entry requests from clients in the presentation layer. There are different types of requests, each one needing a particular processing scheme. Therefore, when a request arrives to the application, it should pass through a set of verifications before reaching the main processing phase – called the Front Controller – authentication, session validation, client IP address checking, request authorization, data codification, auditory or browser type used. The Intercepting Filter pattern is a flexible and highly decoupled way to intercept a request, applying a set of filters, thus rejecting or allowing the request to arrive to the initial process.

This initial process plays the controller's role: it analyzes each request to identify the operation to perform, thus invoking the business logic associated to each particular request and controlling the flow to the following view. In the proposed model [Figure: 9], our controller is designed following the Service To Worker pattern, which combines a set of smaller patterns that provide a complete and flexible solution to fulfil the requirements for an MVC controller while allowing the separation between actions – the model –, the view and the controller.

The Front Controller pattern describes a central point that manages the requests. In order to reach a greater flexibility and independence between the view and the model, the Front Controller assumes only the request analysis task, delegating in the Request Dispatcher the selection of the view and the action to perform. After the analysis phase, the Request Dispatcher will be in charge to select the command that encapsulates the operation to perform. Once this command has generated the result, the Request Dispatcher will select the next view to be shown to the user. Delegating these tasks in the Request Dispatcher gives our model a greater flexibility since it can introduce new views or models in the scenario by altering the component's behaviour.

The Command pattern represents each request by means of an object, therefore providing a very simple way to introduce new operations. In our model, the Command pattern is responsible for encapsulating the request information, parameters and the current state into a command object that contains the business logic. This command is then sent through the network to the model, where it is finally executed (EJB Command). By using this approach, it achieves a complete decoupling between the controller and the model, which represents the business data and implements the

rules to operate them. Following the same approach, in the model shown in [Figure 9], here introduce the business layer as a set of patterns that completely disconnect the controller and the view from the model, thus achieving MVC paradigm's objective. On the other hand, it defines another set of patterns in order to integrate our model with inherited systems or other business models.

In the first case it applys the Business Delegate, Session Facade and EJB Command patterns. The EJB Command pattern is a special case of the Command pattern where the business logic is encapsulated into a serializable object created by a remote client – the controller's Command – and sent through the network to the EJB container where it will be executed – by invoking its execute method. This scheme provides the advantages of the Command pattern in an environment where the business logic is distributed, therefore allowing the execution of business rules without overloading the application by a massive usage of EJBs.

For the second case, this has defined two patterns to ease the integration between the business model and the inherited systems or other business models. The Data Object Access pattern supplies a mechanism to abstract and encapsulate access to the data sources, therefore achieving warehouse independency. It also achieves a clear separation between the business logic and the data logic, increasing the applications' maintenance capabilities. The Service Activator pattern describes a way to access other business models and services in an asynchronous manner. When a message is received, The view is responsible for showing the data output by the MVC model. One of our models' goals is to decouple the presentation from the controller and the model, and to achieve this the model's output is first produced in XML format, for its later transformation by XSLT sheets into the final presentation shown to the client. XML/XSLT is an elegant way to separate the data from the presentation and to free it from any particular technology.

6. CONCLUSION

This paper identifies the primary difference between modified MVC design and more traditional version of MVC which is notifications of state changes in model objects communicating to the view through the controller. Hence, the controller mediates the flow of data between model and view objects in both directions. View objects use the controller to translate user actions into property updates on the model. In addition, changes in model state are communicated to view objects through an application's controller objects. Multiple framework architecture works better as compare to any single framework architecture with the effective of the multiple frameworks for the development of the large scale applications.

7. FUTURE RESEARCH

Open technologies are the best to attract the academic and research scholar to work. J2EE is the vast field now a day, its open technologies also. Architecture is never fixed its goes on changing with the change in the technology. There are many frameworks available to work with J2EE technologies, Single frame is never sufficient to provide the complete solution with all essential features of the application. There is a lot of scope to work further with many other frameworks to implement and enhance the MVC architecture.

8. ACKNOWLEDGMENT

My sincere thanks to Ms. Neelam Bawane, Assistant Professor, Department of MCA, PES Institute of Technology for contribute her innovative ideas and support towards development of the paper.

9. REFERENCES

[1] Apple Mac OS X as on 13th September 2011 http://developer.apple.com/technologies/mac/cocoa.html

[2] Java Notes as on 7th September 2011 http://leepoint.net/notes-java/index.html

[3] Java SE Application Design with MVC, Article by Robert Eckstein, March 2007

[4] Pattern Oriented Software Architecture- Vol. I by Frank Buschmann, Regine Munie, Hans Rohnert, Peter Sommerlad, Michal Stal - John Wiley & sons Publication, 2006

[5] Practical Object-Oriented Design with UML 2nd Edition by Mark Priestley, 2003

[6] Praveen Gupta et. al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, 1047-1051

[7] Erxiang Chen; Minghui Liu, "Research and Design on Library Management System Based on Struts and Hibernate Framework", in WASE International Conference on Information Engineering ICIE 09, 2009, Vol. 2, PP. 310-313

[8] Juanjuan Yan; Bo Chen; Xiu-e Gao, "Le Wang; Research of Structure Integration Based on Struts and Hibernate", in 2009 WRI World Congress on Computer Science and Information Engineering, 2009, vol. 7, PP. 530-534.

[9] Yonglei Tao: "Component- vs. application-level MVC architecture", in Frontiers in Education 2002 FIE 2002. 32nd Annual, 2002, Vol 1, PP. T2G-7 - T2G-10

[10] Meiyu Fang, "Design and Implement of a Web Examination System Using Struts and EJB", Seventh International Conference on in Web-based Learning 2008, 2008, pp. 25-28

[11] Shu-qiang Huang, Huan-ming Zhang," Research on Improved MVC Design Pattern Based on Struts and XSL", in Information Science and Engineering ISISE 08 International Symposium on, 2008, vol. 1 PP. 451 – 455.

[12] Wojciechowski, J.; Sakowicz, B.; Dura, K.; Napieralski, A.,"MVC model, struts framework and file upload issues in web applications based on J2EE platform", in Proceedings of the International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science 2004, PP 342-345.

[13] Wang Ning; Li Liming; Wang Yanzhang; Wang Yi-bing; Wang Jing, "Research on the Web Information System Development Platform Based on MVC Design Pattern",in IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008 , Vol 3,pp. 203-206

[14] Decoupling MVC: J2EE design pattern integration by Francisco macia-perez, virgilio gilart-iglesias, diego marcos-jorquera, juan manuel garcia-chamizo.