Survey on G-MVC Architecture by providing View Rules to handle the Action Events of Controller and States of Model

Vishal Ketankumar Shah
M.E. Computer Engineering, L.J.I.E.T

***Abstract*- In feats to support the growing trend of the Java Programming language and to promote generalized MVC architecture of web-based and mobile based web service for fastest development of projects and products in less time. Every software applications that interact with a user require a graphical user interface (GUI) and hidden basic data operations. Model-View-Controller (MVC) is a common design pattern to integrate a GUI with the application based domain logic and data source operation. MVC separates the representation of the application domain logic (Model) from the display page of the application's state (View) and graphical user interaction control (Controller). The aim of this research focuses on generalized MVC (G-MVC) architecture which is built on the top of MVC design pattern. Our G-MVC uses for developing web applications and JSON and XML based web service for mobile applications without separate code of domain logics and business logics. Our proposed architecture acts like a mediator between application's states and database. It saves time for coding of programmer as well as reduces the lines of code by using G-MVC architecture. This architecture supports JSON and XML based web services interact to develop mobile based applications through Universal Resource Locator (URL).**

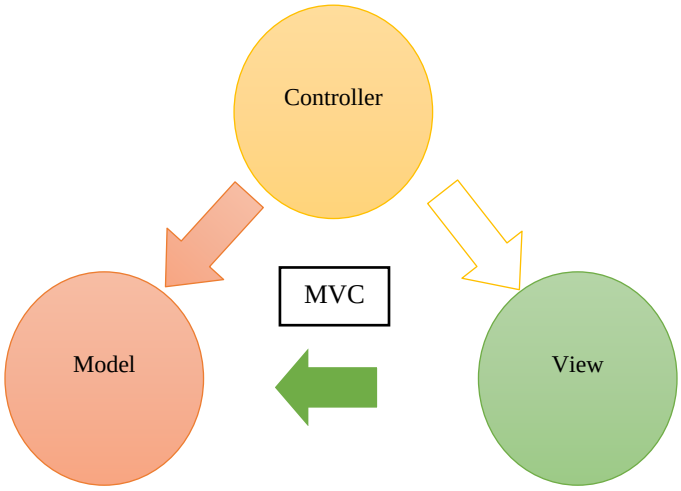
I. INTRODUCTION

In software development, a design pattern is a universal repeatable solution to a usually occurring problem in the process of implementing software solutions. A design pattern isn't finished design that can be transformed directly into code. It is template for how to solve a problem that can be used in many trail and errors over a significant period of time. Same design pattern use for developing multiple domain applications without changing the application logic or workflow logic to interact with domain and infrastructure components according to user requirements. In design pattern, the business logic should be placed in the model (M) of MVC. Every software program that requires at least a bit of interactivity with users requires a user interface. This makes the integration of the user interface with the application domain a recurring engineering problem. Design patterns provide generic solution schemes for recurring design problems, offering reference materials that give engineers access to the field's systematic knowledge [1]. It's one kind of software architecture – the structure of the system – that separates domain/application/business logic from the rest of the user interface. It does this by separating the application into three parts: the model, the view, the controller

Model-View-Controller (MVC) is a software design pattern. Using this pattern, the system is divided into three modules, every module has own function. This is a typical multi-tier structure designing ideas. MVC is a software architectural design mostly for implementing the user interfaces. It divides a given software application into three interconnection parts, so as to separate internal representation of information from the ways that information to or accepted from the user.[3]

1. The Model: The model is the database communication and logical structure. It deals with the actual data in the database with high level associated classes. The logic for inserting, updating, deleting and retrieving data is written in the model.
2. The View: The view deals with User Interface that the users get to interact with the user. Using view the user can interact with in application screens and can send multiple requests to the server for performing operations.
3. The Controller: The controller is the mediator between view and model. It takes request from the user and interact with the model to give response back to the users.

In simple manner to say MVC (Model-View-Controller) as "SMART Model, THIN Controller, and DUMB Views". There is the relationship of the model, the view and the controller [4], shown as Fig.



Today's current trend to use MVC design pattern (architecture) use in multiple types of application which are below:

a. Web Based MVC
b. Desktop Based MVC
c. Mobile Based MVC

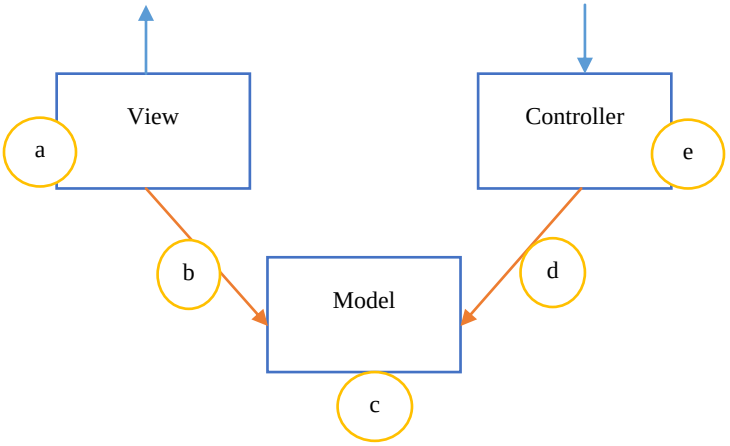
In all above MVC, "Model" mainly use for the any of database operation for the application and connected with the backed portion of the application structure. "View" is font end which is providing the inputs for performing the operation from user. In web application "View" is HTML, HTML5, CSS, CSS3, JS, JQuery controls. In desktop application "View" is graphical based swing input controls and ActiveX controls. In mobile application is XML based controls in the android, iPhone, windows, WAP. Controller to manage the user inputs which comes from the view side and decide to handle the methods of model for which kind of methods to operate for the database tables. For mobile and desktop based application to create JSON, XML services which are work as message based MVC which are deployed on the web server. So services are part of the web application and communicate with desktop and mobile application for the central data models.

II. BACKGROUND THEORY OF MVC

MVC was one of the seminal insights in the early development of graphical user interfaces, and one of the first approaches to describe and implement software constructs in terms of their responsibilities.

Trygve Reenskaug introduced MVC into Smaltalk-76 while visiting Xerox Parc[4][5] in the 1970s. In the 1980s, Jim Althoff and others implemented a version of MVC for the Smaltalk-80 class library. It was only later, in a 1988 article in The Journal of Object Technology, that MVC was expressed as a general concepts. [6] The three key components of the Smalltalk'80 MVC pattern are Model, View and Controller.

The Model component is responsible for the domain data and logic. This component has no reference to the other components of the triad. As such, the application logic does not depend on the presentation of domain data. The View component is responsible for displaying model data. The last one, Controller is responsible for handling user input.



a. View display data from the Model
b. View shows the model and reforms itself when some data is changed
c. Domain data
d. Controller shows the model and triggers model methods on certain user actions
e. Controller handles user inputs

View and Controller work as a pair allowing the user to interact with via the user interface. For example, the user interface may provide a text box allowing the user to enter a user name. The View is responsible for rendering the text box. The user can change the text and press buttons (e.g., enter) – such events are handled by the Controller. The Model maintains the domain data. Often, the application has one Model and a set of View-Controller pairs working with it. Although View and Controller work in pairs, they are considered as two separate entities with minimal coupling. That is, displaying data and handling user input are treated as distinct activities enhancing separation of concerns.

III. TYPES OF MVC

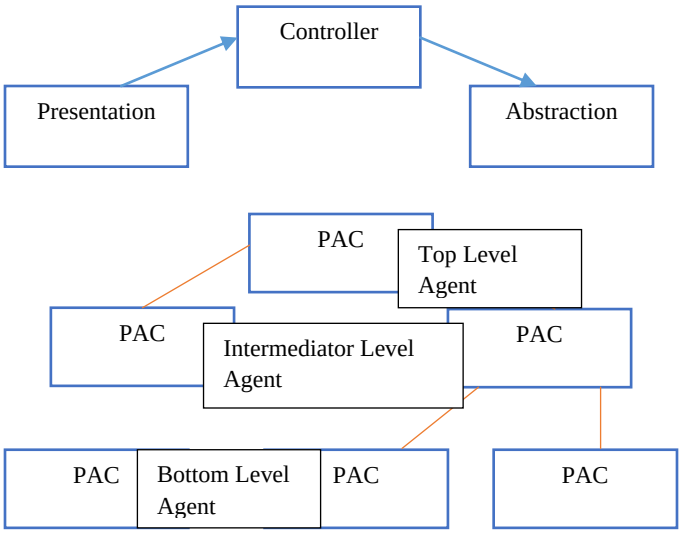
The MVC pattern has subsequently evolved, giving rise to variants such as HMVC, MVA, MVP, MVVM, and others that adapted MVC to different contexts.

- HMVC – Hierarchical model-view-controller
- MVA – Model-view-adapter
- MVP – Model-view-presenter
- MVVM – Model-view-view-model

The use of the MVC pattern in web applications exploded in popularity after the introduction of spring framework for Java. The introduction of the frameworks Rails and Django, both of which had a strong emphasis on rapid deployment, increased its popularity outside the traditional enterprise environment in which MVC has long been popular. MVC web frameworks now hold large market shares relative to non-MVC web toolkits.[4]

IV. HMVC

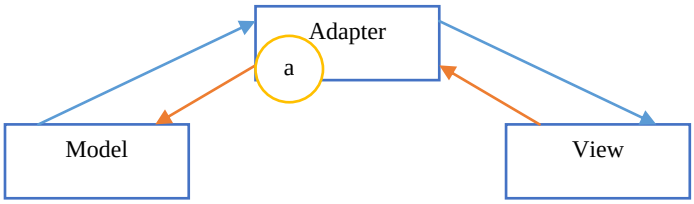
In this architecture, there are multiple variation of MVC, MVA, and MVP. This kind of MVC called as Presentation-Abstraction-Controller (PAC). The controller selects the model and then selects the view, so there is an approval mechanism by the controller. The model presents the view from accessing the data source directly.



This above figure as PAC put as hierarchical level of tops level, intermediator level, and bottom level. There are multiple intermediator level as multiple levels for their business application logics.

V. MVA

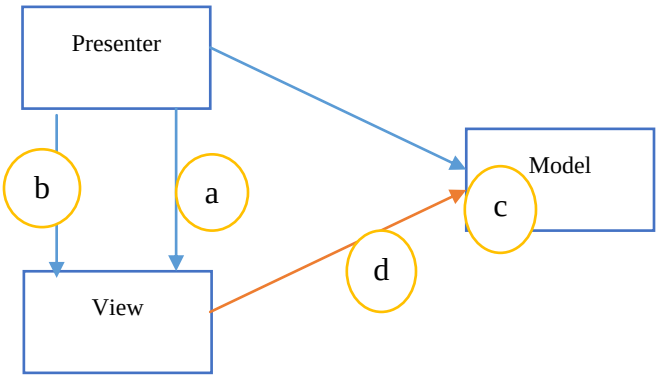
In complex programming applications that presents large amount of data to users, developers often wish to separate data model and user interface view. So that changes to user interface will not affect data handling and that the data can be recognized without changing the user interface. The Adapter holds a pointer both to the Model and to the View and directly calls methods on both. At the same time, it attaches itself as a listener both to the Model and to the View in order to receive events. It receives property change events from the Model and action events (checkbox ticked, text entered, etc.) from the View, and then routes appropriate changes to the other side. The Adapter is entirely responsible for keeping the Model and the View in sync; the Model and View are both relatively dumb structures, knowing nothing about the other



a. Control all logic data and view components as listener

VI. MVP

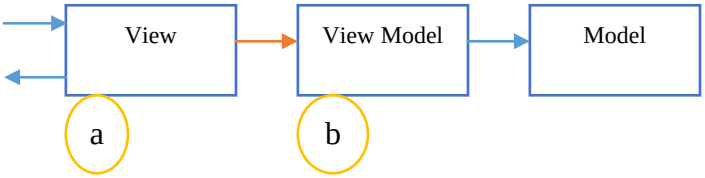
The Model component represents pure domain data. As in earlier patterns, Model is unaware of the presentation logic, but it still provides notification when the data is changed. The role of View in MVP remained almost unchanged. View is responsible for displaying the data on the user interface. It also supports basic handling of user input: it delegates user actions to the Presenter by direct calls. The Presenter is responsible for keeping the application synchronized. Presenter handles user input, invokes domain methods, keeps the Model in consistent state and provides extra logic to update the View when necessary.



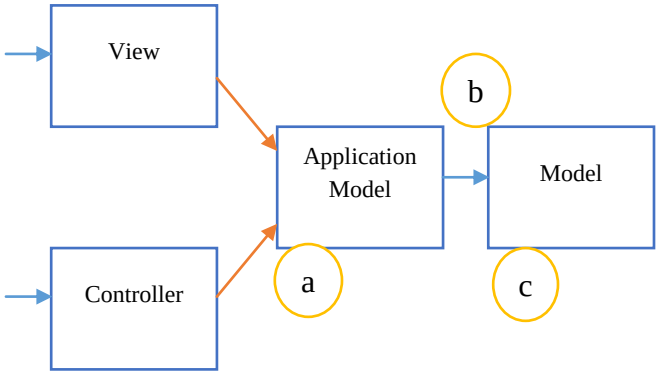
a. Presentation has direct access to view
b. View routes events to Presenter
c. Purely domain data
d. Simple 2-way data binding View can reflect and modify the Model data

VII. MVVM

The pattern has linear structure. The View is responsible for rendering the user interface; it can observe the View Model, trigger it methods and modify its properties when needed. View Model is responsible for handling view state and user interaction; it has access to the domain Model, so it could work with domain data and invoke business logic. View Model is unaware of View. Model is responsible for handling domain data and is unaware of View Model. This approach allows creating different views for the same data and observer synchronization allows this views working simultaneously.



a. View is bounded to View Model in a declarative manner, contains no extra logics
b. Handles view state and work with domain data



a. Application model mediates the cooperation and handles View state
b. Application model observes the model and triggers its methods when necessary
c. Domain model

VIII. TYPES OF MVC ARCHITECTURE MODEL

Architecture is complex structure to define or design very carefully based on different layers. And provide the relation between the layers to interact with one another. This structure is define in Object Oriented Programming and Structure Programing language for developing web, mobile and desktop based applications. In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design [9], which are known as architecture pattern. Mainly architecture (design pattern) has algorithm strategies, computation process, execution environment, implementation strategy, and global structure in the domain and interconnections. There are mainly two types of MVC architecture to be work for developing any of application for any of context.

- Push MVC model
- Pull MVC model

Push MVC model

According to the push model, user actions should be interpreted by the Controller which will generate the data and push it on to the View, hence the “push”. In short, "push" application, a server is pushing or sending data to clients at its own initiation. The classic example is that of two different stock quote programs. In a push application, the server sends a message to the client whenever the price of the stock changes.

As per J2EE perspective, this is where the framework creates context objects what are "pushed" or made available to the templating language like JSP's allowing them either via tags or scripting to get their values and display them on the page. Struts and Expresso a good examples of this. In Expresso you create Output objects that are made just for rendering the View.

In Java based framework, there are Struts 1.X and String MVC following Push based MVC model.

Pull MVC model

According to the pull model assumes that the user requires some kind of output (like a list items from the database). The View will access the Controller in order to get the data it needs in order to display the user the kind of output he requested. This is much like the View is “pulling” data from the Controller. In short, "pull" application, the server is pulling or waiting for and receiving messages initiated by clients. The classic example is that of two different stock quote programs. In a pull application, the server will respond to client requests to get the current price of the stock.

As per J2EE perspective, this is where you have one or a few objects that are made available to all templates. The big difference is the java developer does not need to create any sort of output object, they just make backend model objects available to the View templates. Webwork and Maverick are a good example of this, they a provide accessor methods(getters/setters) in their Action/Model classes that allow the View to "pull" whatever they like as long as they know of the API for that Action.

In Java based framework, there is Struts 2.X following Push based MVC model.

Example with Mechanism

Web applications are stateless as they use the stateless protocol HTTP. So, once a request is served to the client then there is no direct way of notifying him/her about the changes made to the Model. To handle this scenario there are two mechanisms -

Push Mechanism: forcibly sending the response back to the client whenever any change to the model happens, but this is not very easy to implement as the server would require to maintain client info. There are other challenges as well. HTTP requests are processed by first establishing the connection to the server, sending the requests, receiving the response, and finally terminating the connection. Once the request has been served the connection is lost. How would the server then send back the updated response? Since the HTTP protocol is stateless the client may simply close the browser window without even notifying the server that it no longer needs any updated information.

Pull Mechanism: this mechanism is normally used to handle the above mentioned scenario. This requires the client to pull the updated data rather than the server pushing the updated data to the client. Either by having a simple button (clicking which will make another fresh request to the server and the client would get the updated data in response) or by having a script which would automatically be making the HTTP requests - maybe periodically after a certain time-interval. This is how Live Cricket Scores are displayed.

IX. LITERATURE SURVEY

In "Web Application Development Using Model/View/Controller Design Pattern [10]" by Avraham Leff, Jemes T. Rayfield introduces the concept of flexible We-Application partitioning, a programming model and implementation infrastructure that allows developers to apply the Model/View/Controller design pattern in a partition-independent manner and location dependent environment. Applications are developed are tested in a single address-space; they can then be deployed to various client/server architectures and partitioning decisions without changing the application's source code. Fwap supports single-mvc(smvc), thin-client and dual-mvc (dmvc) architectures implementing the algorithms and infrastructure needed to enable fwaplications to scale over non-trivial application Models. We are also working with a customer to validate the fwap concepts and implementation.[10]

In "Combining HTML 5 with MVC Framework to Simplify Real-Time Collaboration for Web Development [11]" by Yuehui Yu, Lei Ning, Weizhong Liu proposes a joint framework for HTML 5 specification and a double model architecture for real-time web collaboration development. [11] A Simple Collaboration Modeling Language is developed to illustrate the shared model concept. [11] With client-side library and server-side components, programmers are able to concentrate on collaborative logic instead of application details. These concepts can be presented in SCML and decomposed into JavaScript objects respectively. By using our RTWCC library, these objects have been given this ability: as a front-end representative to the shared states. [11] The rest work is the development of the user interface, which is important for the user's experience in collaborative environment but not mentioned in the above statements. An ordinary programmer can do this job on the API of our (Real Time Web Collaboration Client) RTWCC library. [11] Same as "A Web Architectural Study of HTML5 with MVC Framework [12]" by Jatin Chakra proposes The HTML5 based MVC model is a view based model that provide the platform independent development model.[12] Also include this HTML5 feature representation for Mobile and Web based application under MVC framework.[12] HTML5 provides the structural and presentation based changes to the web contents over the mobile and web system.[12] HTML5 integrated MVC architecture and different components of MVC model under the view point of analysis.

In "A Database and Web Application Based on MVC Architecture [13]" by Diana M. Selfa, Maya Carrillo, Ma. del Rocío Boone purpose of illustrating a successful application built under MVC, in this work we introduce different phases of analysis, design and implementation of a database and web application using UML.[13] As central component of the application, it has a database made up by fifteen relations and a user interface supported by seventeen web pages.[13] It was also evident that the construction of clear and understandable models for the users and software engineers allows the construction of quality systems with predictable development times.[13]

In "Balanced MVC Architecture for Developing Service-based Mobile Applications [14]" by Hyun Jung La and Soo Dong Kim propose a unique, ideal and practical architecture for service-based mobile applications, called balanced Model-View-Controller (MVC) architecture.[8] The architecture is devised by adopting three architectural principles; being thin-client, being layered with MVC, and being balanced between client side and server side. define methods to partition the functionality optimally between client and provider sides, and to design a balanced MVC architecture for Service-based Mobile Applications (SMA) by adopting three architectural

principles; being thin-client, being layered with MVC, and being balanced between client side and server side. [8] The methods are based on artifacts such as use case model and object model to provide the high quality of mobile based application based on the effectively services.

In "Building Desktop Applications with Web Services in a Message-based MVC Paradigm [7]" by Xiaohong Qiu present an approach of building desktop applications with Web Services in an explicit message-based MVC paradigm. [7] By integrating with our publish/subscribe messaging middleware, it makes Scalable Vector Graphics (SVG) browser (a Microsoft PowerPoint like client application) with Web Service style interfaces universally accessible from different client platforms.[7] As SVG is an application of theW3C DOM, we can generalize the approach for other W3C or similar DOM based applications. Our approach suggests that one need not develop special "collaborative" applications. Rather any application developed as a Web service can be made collaborative using the tools and architectural principles. [7]

In "Implementation of a Five-tiers Architecture Based on Struts and Hibernate for Web Applications [14]" by LI Zhuo-ling, ZHU Shi-don, ZHANG Xin present an efficient five-tiers architecture based on integrating Struts and Hibernate frameworks, described the implementation of a web application in details as a sample focusing on utilizing [14] Presentation layer, Controller layer, Business Logic layer, Data Persistence layer and Database layer. Struts can high coupling problems among logic, business and data, together with that Hibernate can deal with the object persistence and encapsulate database operations. In this context, MVC pattern can be achieved by using Struts framework, and database operations can be encapsulated by utilizing Hibernate framework. [14] In "J2EE and MVC Architecture [15]" by Manish Bhatt proposed that MVC Architecture to convert event in appropriate action, understand for the model for changing the states, View gets the data forms itself by rendering and notify state change. Use W3C Xfrom concept use for automatically updating the views by updating the forms work like as Struts Pull MVC. [15] Use the database operation for the Hibernate like frameworks for the data services.[15] Based on the development architecture composed by the Struts and the Hibernate, it passes the data by the value object which the layers corresponding to, and strictly controls the visit to the persistence layer by the users. In this way, it can protect the business data effectively. [15]

X. CONCLUSION

Based on the Literature survey, I am going to conclude that there are different types of MVC architecture are hard to code, not easy to single modification, also developer required more time to finding out the root cause of applications when the errors occurs. Developer do MVC code for the single form design and write logics as code, so each individual form data to follow same architecture with different code for the storing the data entries due to the different entries. Also different code for the mobile based service APIs. In simple manner, multiple types of form data to follow individual web MVC and mobile service APIs codes. So it's sustain process. By the use our proposed architecture I hope it will defiantly beneficial for the each developers because of providing easy to use, less complexity and complicated on coding part with lesser time with efficient code structure. With this Generalized-MVC architecture pre built provide the JSON and XML based API services without any extra code of the applications. So, providing the view rules for handling the web requests and mobile services (action events) of controller. With this controller to handle model's state on web applications. Through this architecture develop web and mobile services (JSON & XML) applications very easily with lesser and efficient code fragments.

REFERENCES

1. P. Clements and M. Shaw, "The Golden Age of Software Architecture: Revisited," IEEE Software, vol. 26, no. 4, pp. 70–72, 2009.
2. Wu Wei, Lu Jian-de, Research of Layer Pattern on Developing of J2EE Application, Microcomputer Development, Vol.15, No.1, 2005.1, pp.125-127
3. "More deeply, the framework exists to separate the representation of information from user interaction." The DCI Architecture: A New Vision of Object-Oriented Programming -Trygve Reenskaug and James Coplien - March 20, 2009
4. Hot Frameworks - http://hotframeworks.com/
5. A. Bower and B. McGlashan, "Twisting the triad," Tutorial Paper for European Smalltalk User Group (ESUP), 2000.
6. S. Burbeck, "Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc)," Smalltalk-80 v2. 5. ParcPlace, 1992.
7. Xiaohong Qiu, "Building Desktop Applications with Web Services in a Message-based MVC Paradigm", Web Services, 2004. Proceedings. IEEE International Conference on Page(s): 765 - 768

8. Hyun Jung La and Soo Dong Kim, “Balanced MVC Architecture for Developing Service-based Mobile Applications”, e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on Page(s): 292 – 299
9. https://en.wikipedia.org/wiki/Software_design_pattern
10. Avraham Leff, James T. Rayfield “Web-Application Development Using the ModelNiewlController Design Pattern” IEEE on Page(s): 118 – 127
11. Yuehui Yu, Lei Ning, Weizhong Liu “Combining HTML 5 with MVC Framework to Simplify Real-Time Collaboration for Web Development” – IEEE on Pages: 29-32
12. A Web Architectural Study of HTML5 with MVC Framework – Jatin Chhikara – IJARCSSE Volume 3, Issue 12, December 2013, ISSN : 2277 128X, Page 451 - 454
13. Diana M. Selfa, Maya Carrillo, Ma. del Rocío Boone “A Database and Web Application Based on MVC Architecture” Proceedings of the 16th IEEE International Conference on Electronics, Communications and Computers IEEE
14. LI Zhuo-ling, ZHU Shi-don, ZHANG Xin “Implementation of a Five-tiers Architecture Based on Struts and Hibernate for Web Applications” IEEE
15. Manish Bhatt “J2EE and MVC Architecture” JGRCST – Volume 1, Issue 2 - July’ 2014