ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

A Comparative Study of Architectural Patterns used for website or enterprise Applications

Rajendra Satpute [1], Prof. P. M. Chawan [2], Prof. Imran Jamadar [3], Prof. Bhumesh Masram [4]

P.G. Student, Department of Computer Engineering and IT, VJTI, Mumbai, Maharashtra, India [1]

Associate Professor, Department of Computer Engineering and IT, VJTI, Mumbai, Maharashtra, India [2]

Assistant Professor, Department of Computer Engineering and IT, VJTI, Mumbai, Maharashtra, India [3]

Assistant Professor, Department of Computer Engineering and IT, VJTI, Mumbai, Maharashtra, India [4]

ABSTRACT: An architectural pattern provides the solution for designing a software architecture. For a particular functionality and nature of software, architectural patterns differ. An architectural pattern conveys only the image of the system, not the details about it. Even though we have many architectural patterns, choosing the right one for the system is tricky. It not only depends on the functionality to be delivered, but the technology and future use is also considered so as to minimize the rework cost as each architecture pattern defines the way to build an architecture which then defines the system.

KEYWORDS: Architectural patterns, software architecture, software architectural pattern.

I. INTRODUCTION

A pattern can be defined as an idea which is useful in one or more practical contexts to give reusable solution to a particular problem. Architectural patterns have a broader scope. Many software architectures may have same architectural pattern. It helps to solve many issues in software engineering. An architectural pattern is collection of architectural design decisions application in a given context. Some architectural patterns are basic while others are derived or improvement of the basic architectural patterns.
In this paper we are going to compare following architectural patterns -

- Multi-tier or 3-tier architectural pattern or State Logic Display
- Model View Controller
- Model view presenter
- Model View ViewModel

John J. Donovan developed N-tier architecture. Trygve Reenskaug [1] introduced MVC in 1970s. Mike Potel [2] pro-posed MVP in 1996. John Goss-man [4] defined MVVM in 2005.

II. MULTITIER ARCHITECTURE

The most widespread use of multi-tier architecture is the three-tier architecture. A three-tier architecture is typically composed of a state tier, logic tier, and display tier. Multitier architecture provides a model by which developers can create scalable, flexible and reusable applications. Due to segregation of application into tiers, developers can modify or add a specific layer, instead of reworking the entire application.
The major difference between tier and layer here is that the layer refers to the logical separation of concern while tier refers to the physical separation of concern.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

Modules of three-tier architecture which are user interface (display), business logic (logic), and computer data storage(state) are developed and maintained independently , mostly on separate platforms

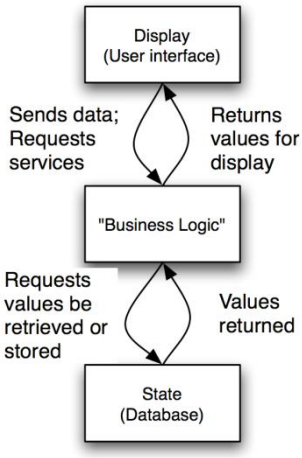

Fig. 1. State Logic Display or 3 tier architecture

It has the advantage of modular software with well-defined interfaces. Any of the three tiers in the three-tier architecture can be upgraded or replaced independently with changes in requirements or technology.
The middle tier may be multi-tiered for which this architectural pattern is also called multi-tier architecture or n-tier architecture. Data flow between tiers is part of the architecture. Middleware usually connect the two tiers.

III. MODEL VIEW CONTROLLER

Model View Controller contains three components – View, Model and Controller.

- View is your UI. It displays the data received from the controller as the result.
- Model is your business entities / data. It defines business rules for data means how the data can be manipulated or changed.
- Controller contains the logic that makes changes in the model depending on the actions triggered by UI. It manipulates the data.

Disadvantage of MVC is that it's difficult to unit test.
Controller does not know anything about View and a View can switch Controllers. A single controller can be used by multiple Views. View tracks the changes done to the model, so both are in sync with respect to the data.

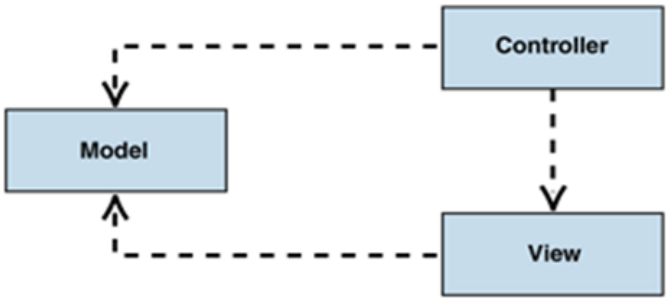

Fig.2 Model View Controller

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

IV. MODEL VIEW PRESENTER

MVP is derivation of MVC. It is mainly used for developing user interfaces. Presentation acts as a middle-man and controls all presentation logic. It provides facility for automated unit testing.

- Model defines data to be viewed or the data submitted by user interface.
- Presenter acts as a middle-man and connects both Model and View. It formats the data received from Model and displays it in View.
- View acts as a passive interface that views the data from model and captures events forwarding it presenter.

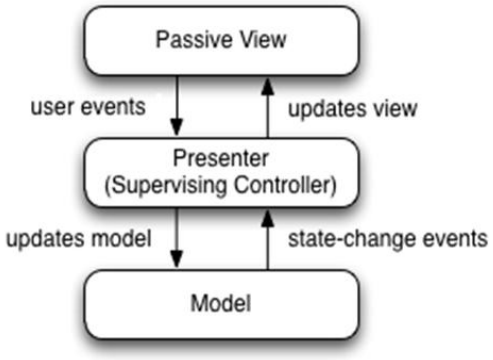

Fig.3 Model View Presenter

V. MODEL VIEW ViewModel

MVVM is derived from MVC. It was developed by Microsoft to develop simple event-driven programming of user interfaces. It is majorly used with Windows Presentation Foundation (WPF) and Silverlight. It is also known as model-view-binder when used with non .NET platforms.

- Model is the domain model or data access layer. It contains data model and business & validation logic.
- ViewModel acts as a middle-man and connects both Model and View. It handles view logic. It accesses the methods in model and provides this data to View in the format that it can use easily. Data-binding is used to link ViewModel and View.
- View is the system's user interface.

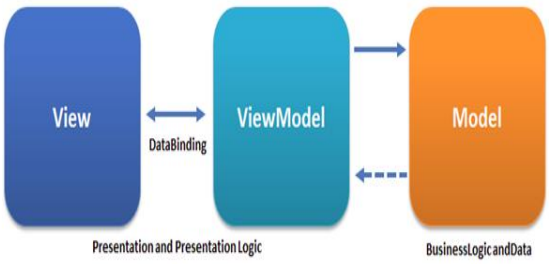

Fig.4 Model View ViewModel

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

VI. COMPARISON

All these architectural patterns are used for user interactive system designs. Following table will show in detail comparison between each architectural pattern.

Comparison:

	N-tier	MVC	MVP	MVVM
Input	Display tier	Controller	View	View
Connectivity	No direct connection between State and Display	Direct connection between View and Model	No direct connection between View and Model	No direct connection between View and Model
Instance of View or Display	Any number of instances of Display for each Logic tier.	Any number of View instances for each Controller	One instance of view per Presenter	Any number of View instances for each Controller
References	Bidirectional references between State & Logic and Logic & Display	View refers to Model, Controller refers to Model and passes data to View	Bidirectional references between Model & Presenter and Presenter & View	Bidirectional references between Model & ViewModel and ViewModel & View

Table 1. Comparison table for architectural patterns

VII. CONCLUSION

The survey in this paper shows that by the time, architectural patterns are evolving considering the requirements of functionality, usability, technology. Each architectural pattern offers difference in development and data flow.

REFERENCES

[1] Glenn E. Krasner and Stephen T. Pope. "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. ", SIGS Publications Denville, NJ, USA, Volume 1 Issue 3, Pages 26 – 49, Aug./Sept. 1988.
[2] Mike Potel. "MVP: Model-View-Presenter. The Taligent Programming Model for C++ and Java.", 1996
[3] Richard N. Taylor, Nenad Medvidovic, , and Eric M. Dashofy. "Software Architecture Foundations, Theory, and Practice", Wiley Publishing, 2009.
[4] Smith, Josh. "WPF Apps with the Model-View-ViewModel Design Pattern", Volume 24 Number 02, MSDN Magazine, February 2009.