Choosing your iOS Application Architecture

By Vinit Dembra

The first hurdle that comes in the way of developing a new top-notch iOS application is choosing the right software architecture.

Selecting the right Software Architecture depends on three main factors:

1. Technology
2. Methodology
3. Set Project requirements

It is quite difficult to highlight where to use or where not use certain architecture solutions.

One of the available options is an old yet interesting architecture for iOS development known as MVC and the second option is the modern architecture, known as VIPER.

VIPER, being the modern one is capable of creating certain business layers with solid software principles and clear architecture design of the application. At the same time, it makes it difficult to choose the curve for the development.

There are three major patterns for developing iOS applications

- MVP (Model-View-Presenter)
- MVC (Model-View-Controller)
- MVVM (Model-View-View-Model)

We will also be discussing the additional pattern - VIPER

Understanding the anatomy of Model, View, Presenter and Controller

1. ***The Model*** - Also known as Data Layer, the Model holds responsibility mainly for writing the business logic, handling database transactions, and network requests.

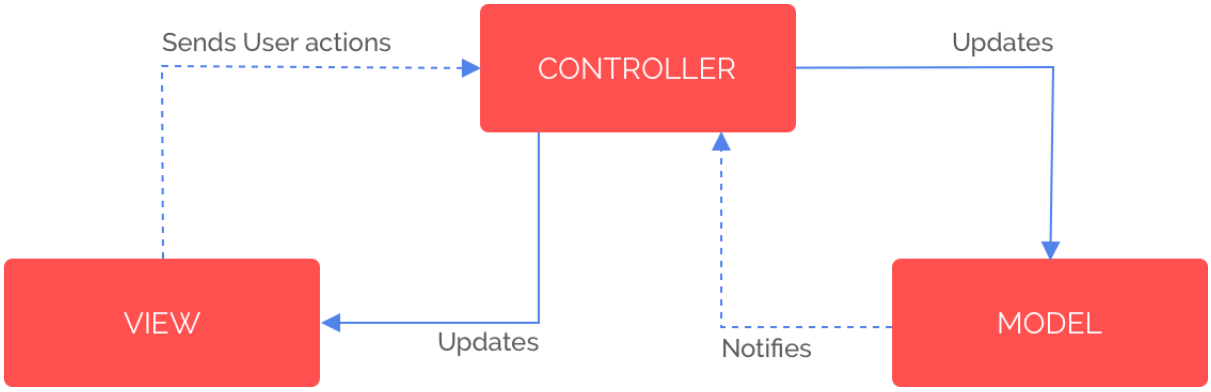
2. ***The View*** - Also known as Presentation layer, the View layer is something which is visible to the user and is utilized to present the data to the user. It helps the user to interact, manipulate, and edit the data. Some of the standard views in iOS view controller are the labels and buttons.

3. ***The Controller*** - The Controller layer acts as a mediator connecting views to the data.

4. ***The Presenter*** - The presenter is responsible for making a connection between the events triggered by the view and the model. In iOS development, the presenter is normally used inside an extension class from the main view controller class.

Demystifying MVC, MVP, MVVM and VIPER

1. MVC architecture (Model - View - Controller)

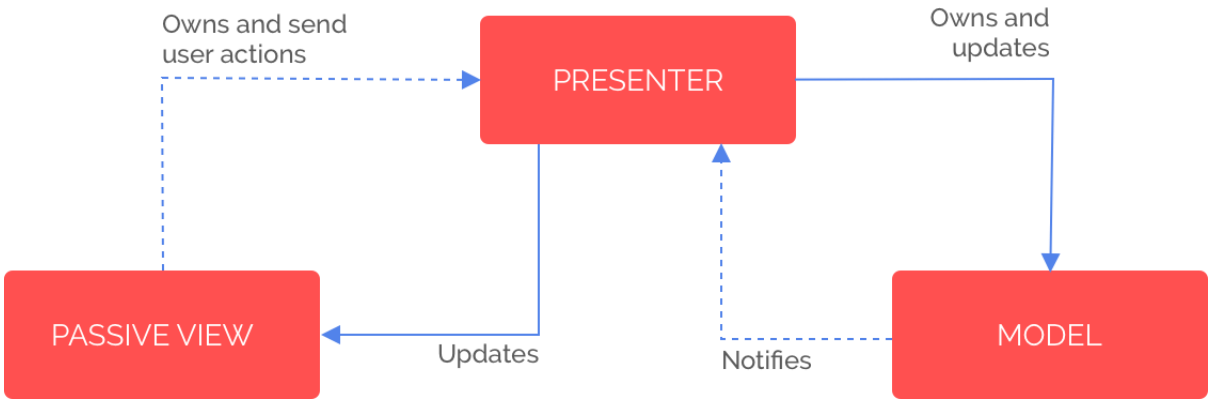


In terms of architecture, MVC is a Front Controller based approach. It is responsible for determining which view to display and which view should be tightly coupled to the model. The central idea behind MVC is that Model and View presenter should never be able to communicate with each other.

In MVC model, view and controller play the lead roles. The beauty of MVC lies in its modular architecture, i.e., a separation of roles which allows developers to carry out modifications and change the requests easily without injecting bugs in the app.

For example, say you want to change the way in which the data is organised or fetched, all that is required to do is change the model, without affecting others.

2. MVP architecture (Model - View - Presenter)

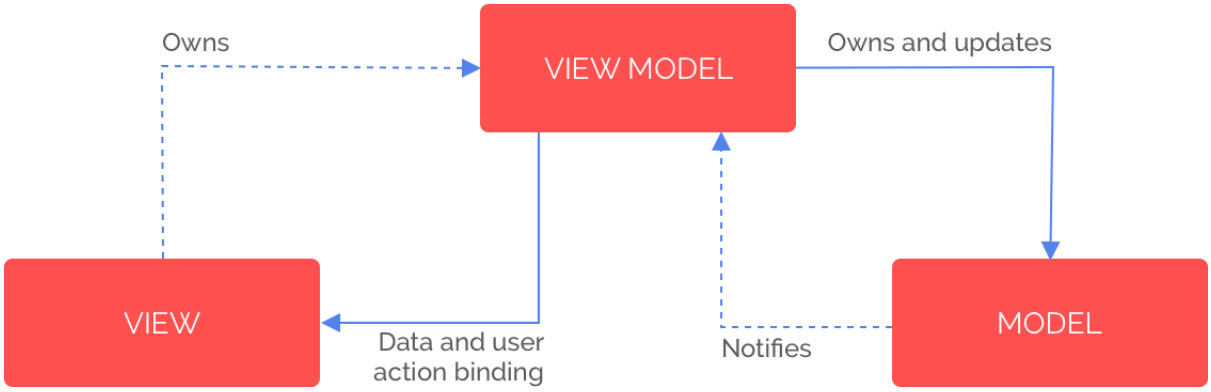


Architecturally MVC is a Page Controller based approach, and the view is loosely coupled to the model. Technically, you can look upon MVP as an evolution to MVC. MVC works as a great architecture pattern, but the drawback of using MVC is that it forces you to write per controller per view.

In MVP, the View draws data from the Presenter, which in turn draws data from the Model. The supremacy of MVP is that you can have a single view working with multiple presenters and single presenter can work with multiple views.

In MVP, the Presenter interacts with the View to access the model. Thus MVP is at a higher architectural level than MVC. But the problem with MVP is that due to the lack of controller, the control flow has to be handled by the presenter. Hence, Presenter is held responsible for presenting as well as updating the model.

3. MVVM architecture (Model - View - View - Model)



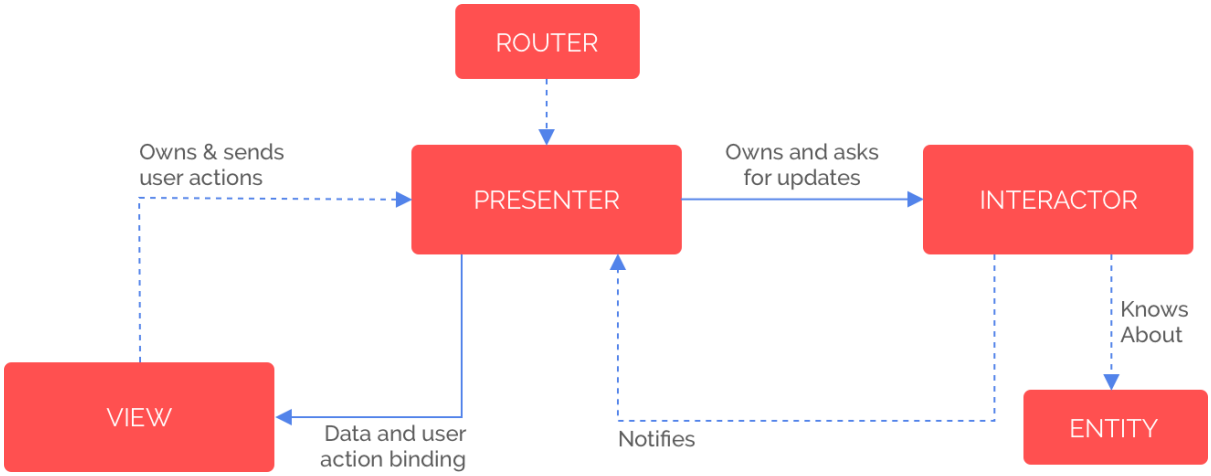
In MVVM architecture, Presenters and Controllers are prone to collect additional business logic, sprinkled in over time. At some time, developers find themselves with large unwieldy presenters that are difficult to break apart.

So in MVVM, the View Model has support from view, especially data binding, which helps to reduce a part of logic codes.

The View Model holds an excellent place to put the validation logic for user input, view presentation logic and network requests.

MVVM with data binding in iOS has benefits of modularity and easier testing, also reducing the amount of the code that has to be written to connect the view and model.

4. VIPER :



VIPER takes inspiration from clean architecture and shares the responsibilities in 5 layers - View, Interactor, Presenter, Entities and Router.

Interactor - Contains business logic and encapsulates data source access and business logic.

Presenter - Contains the UI and invokes the methods of the interactor.

Entities - Plain old swift classes used by the interactor which are the basic objects used for interaction.

Router - Also named as navigation layer, router is responsible for the navigation flows emerging from the presenter.

VIPER is highly criticized by developer community because of its complexity, but it has way more benefits if those complexities are ignored.

Let me conclude the article by summarizing on how to select the suitable Architecture for your iOS app:

When you have a simple utility application where the development time would fall under 2 to 4 weeks, MVC will be the right choice for you.

If you have a simple workflow with a complex utility application that would take up to 4 to 8 weeks for development, you can opt for MVP.

If you have a role driven app which requires more flexibility to change the UI while having to maintain the same logical components of the code, go for MVVM.

And last, but not the least, if you have a highly complex workflow with the development time taking around 6 to 8 months, and you require a scalable app, then you must definitely opt for VIPER.

Contact us for further details

Vinit Dembra
Technology Specialist – JAVA
vinitd.in@mouritech.com

MOURI Tech
www.mouritech.com