Interaction-Oriented Architecture

The primary objective of interaction-oriented architecture is to separate the interaction of user from data abstraction and business data processing. The interaction-oriented software architecture decomposes the system into three major partitions –

- **Data module** – Data module provides the data abstraction and all business logic.
- **Control module** – Control module identifies the flow of control and system configuration actions.
- **View presentation module** – View presentation module is responsible for visual or audio presentation of data output and it also provides an interface for user input.

Interaction-oriented architecture has two major styles – **Model-View-Controller** (MVC) and **Presentation-Abstraction-Control** (PAC). Both MVC and PAC propose three components decomposition and are used for interactive applications such as web applications with multiple talks and user interactions. They are different in their flow of control and organization. PAC is an agent-based hierarchical architecture but MVC does not have a clear hierarchical structure.

Model-View-Controller (MVC)

MVC decomposes a given software application into three interconnected parts that help in separating the internal representations of information from the information presented to or accepted from the user.

Module	Function
Model	Encapsulation the underlying data and business logic
Controller	Respond to user action and direct the application flow
View	Formats and present the data from model to user.

Model

Model is a central component of MVC that directly manages the data, logic, and constraints of an application. It consists of data components, which maintain the raw application data and application logic for interface.

- It is an independent user interface and captures the behavior of application problem domain.
- It is the domain-specific software simulation or implementation of the application's central structure.
- When there has been change in its state, it gives notification to its associated view to produce updated output and the controller to change the available set of commands.

View

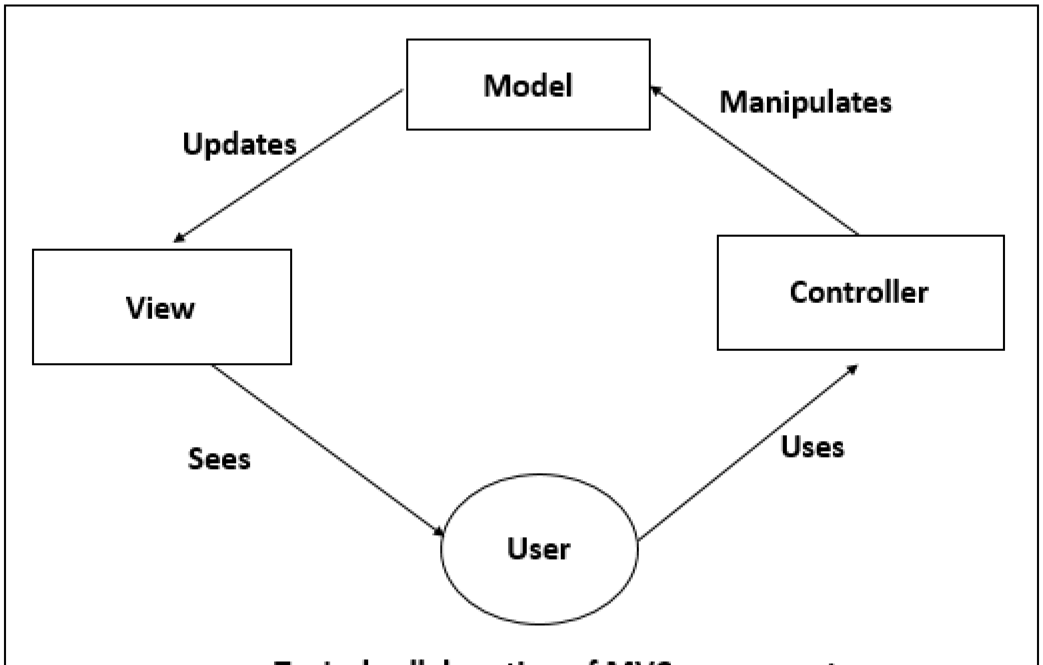
View can be used to represent any output of information in graphical form such as diagram or chart. It consists of presentation components which provide the visual representations of data

- Views request information from their model and generate an output representation to the user.
- Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

Controller

A controller accepts an input and converts it to commands for the model or view. It consists of input processing components which handle input from the user by modifying the model.

- It acts as an interface between the associated models and views and the input devices.
- It can send commands to the model to update the model's state and to its associated view to change the view's presentation of the model.



Typical collaboration of MVC component

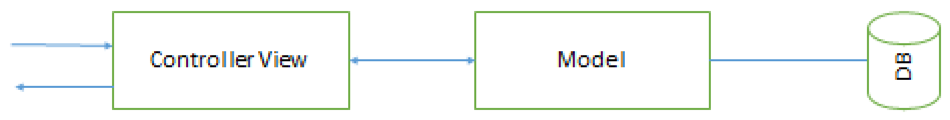
MVC - I

It is a simple version of MVC architecture where the system is divided into two sub-systems –

- **The Controller-View** – The controller-view acts as input /output interface and processing is done.
- **The Model** – The model provides all the data and domain services.

MVC-I Architecture

The model module notifies controller-view module of any data changes so that any graphics data display will be changed accordingly. The controller also takes appropriate action upon the changes.



MVC –I Architecture

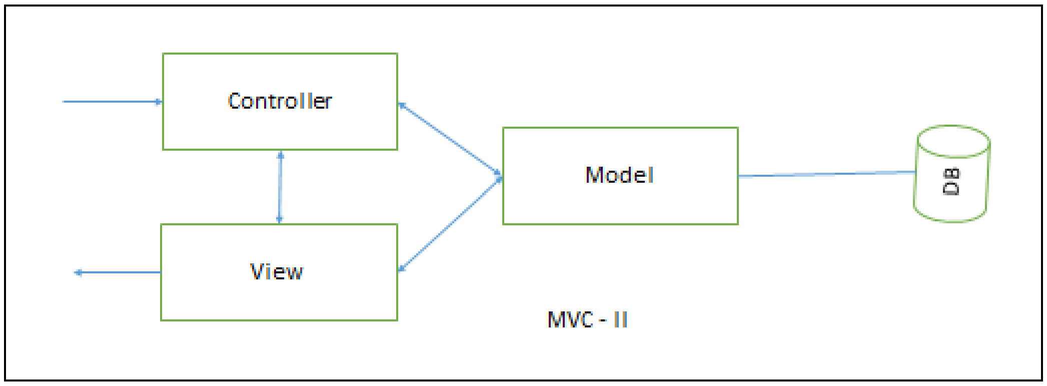
The connection between controller-view and model can be designed in a pattern (as shown in the above picture) of subscribe-notify whereby the controller-view subscribes to model and model notifies controller-view of any changes.

MVC - II

MVC–II is an enhancement of MVC-I architecture in which the view module and the controller module are separate. The model module plays an active role as in MVC-I by providing all the core functionality and data supported by database.

The view module presents data while controller module accepts input request, validates input data, initiates the model, the view, their connection, and also dispatches the task.

MVC-II Architecture



MVC Applications

MVC applications are effective for interactive applications where multiple views are needed for a single data model and easy to plug-in a new or change interface view.

MVC applications are suitable for applications where there are clear divisions between the modules so that different professionals can be assigned to work on different aspects of such applications concurrently.

Advantages

- There are many MVC vendor framework toolkits available.
- Multiple views synchronized with same data model.
- Easy to plug-in new or replace interface views.
- Used for application development where graphics expertise professionals, programming professionals, and data base development professionals are working in a designed project team.

Disadvantages

- Not suitable for agent-oriented applications such as interactive mobile and robotics applications.

- Multiple pairs of controllers and views based on the same data model make any data model change expensive.
- The division between the View and the Controller is not clear in some cases.

Presentation-Abstraction-Control (PAC)

In PAC, the system is arranged into a hierarchy of many cooperating agents (triads). It was developed from MVC to support the application requirement of multiple agents in addition to interactive requirements.

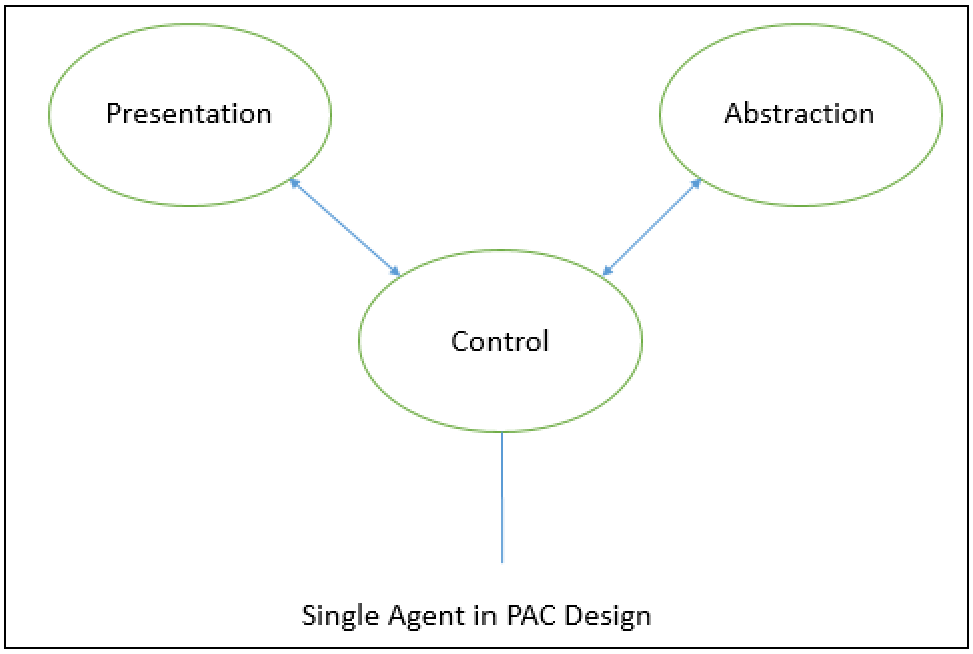
Each agent has three components –

- **The presentation component** – Formats the visual and audio presentation of data.
- **The abstraction component** – Retrieves and processes the data.
- **The control component** – Handles the task such as the flow of control and communication between the other two components.

The PAC architecture is similar to MVC, in the sense that presentation module is like view module of MVC. The abstraction module looks like model module of MVC and the control module is like the controller module of MVC, but they differ in their flow of control and organization.

There are no direct connections between abstraction component and presentation component in each agent. The control component in each agent is in charge of communications with other agents.

The following figure shows a block diagram for a single agent in PAC design.



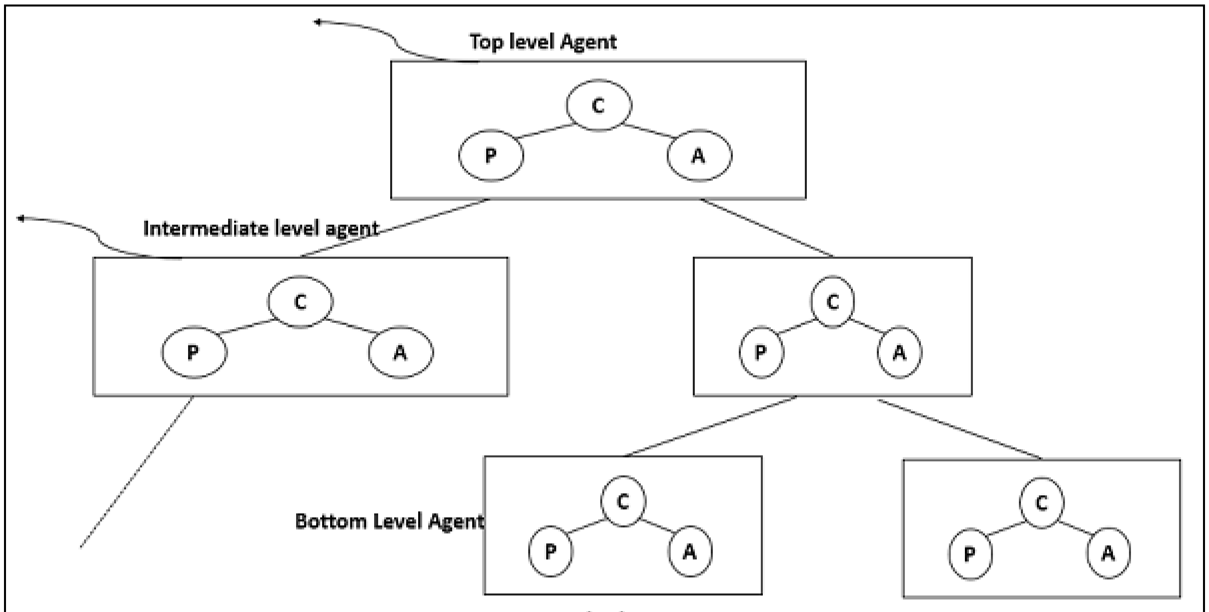
Single Agent in PAC Design

PAC with Multiple Agents

In PACs consisting of multiple agents, the top-level agent provides core data and business logics. The bottom level agents define detailed specific data and presentations. The intermediate level or middle level agent acts as coordinator of low-level agents.

- Each agent has its own specific assigned job.
- For some middle level agents the interactive presentations are not required, so they do not have a presentation component.
- The control component is required for all agents through which all the agents communicate with each other.

The following figure shows the Multiple Agents that take part in PAC.



Multiple agent in PAC

Applications

- Effective for an interactive system where the system can be decomposed into many cooperating agents in a hierarchical manner.
- Effective when the coupling among the agents is expected to be loose so that changes on an agent does not affect others.
- Effective for distributed system where all the agents are distantly distributed and each of them has its own functionalities with data and interactive interface.
- Suitable for applications with rich GUI components where each of them keeps its own current data and interactive interface and needs to communicate with other components.

Advantages

- Support for multi-tasking and multi-viewing
- Support for agent reusability and extensibility
- Easy to plug-in new agent or change an existing one
- Support for concurrency where multiple agents are running in parallel in different threads or different devices or computers

Disadvantages

- Overhead due to the control bridge between presentation and abstraction and the communication of controls among agents.
- Difficult to determine the right number of agents because of loose coupling and high independence among agents.
- Complete separation of presentation and abstraction by control in each agent generate development complexity since communications between agents only take place between the controls of agents