Software Design and Architecture

MVC-I and MVC-II Architecture

Model-View-Controller (MVC)

Model-View-Controller programming is the application of three-way factoring whereby objects of different classes take over the operations related to the application domain (the model), the display of the application's state (the view), and the user interaction with the model and the view (the controller).

- ***Models***: The model of an application is the domain-specific software simulation or implementation of the application's central structure.

- ***Views***: In this metaphor, views deal with everything graphical: they request data from their model and display the data.

- ***Controllers***: Controllers contain the interface between their associated models and views and the input devices (e.g., keyboard, pointing device, time).

MVC-I

- The MVC-I is a simple version of MVC architecture where the system is simply decomposed into two sub-systems: The Controller-View and the Model.

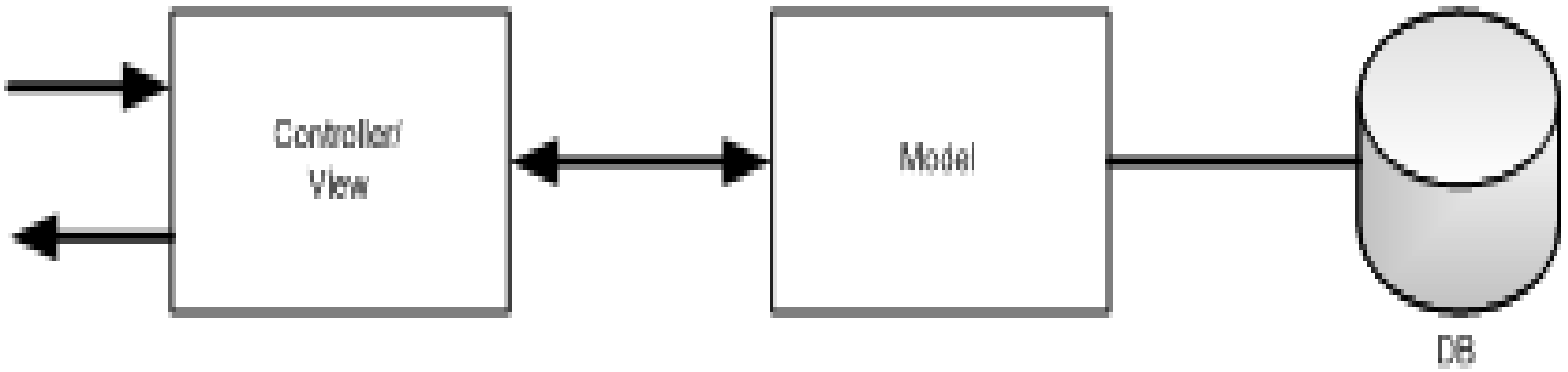
- Basically, the Controller-View takes care of input and output processing and their interfaces, the Model module copes with all core functionality and the data.

- The Controller-View module registers with (attaches to) the data module.

- The Model module notifies the 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.

- The connection between the Controller-View and the Model can be designed in a pattern of subscribe-notify whereby the Controller-View subscribes to the Model and the Model notifies the Controller-View of any changes.

- In other words, the Controller-View is an observer of the data in the Model module.



MVC-I Architecture

- Simple GUI example designed in MVC-I.

- The View has a GUI interface with two text fields, for the user to enter a new number in one of the text fields and the accumulated summation is displayed in the other text field.

- The summation is held in the Model module.

- Model provides all business logics including all getter and setter methods.

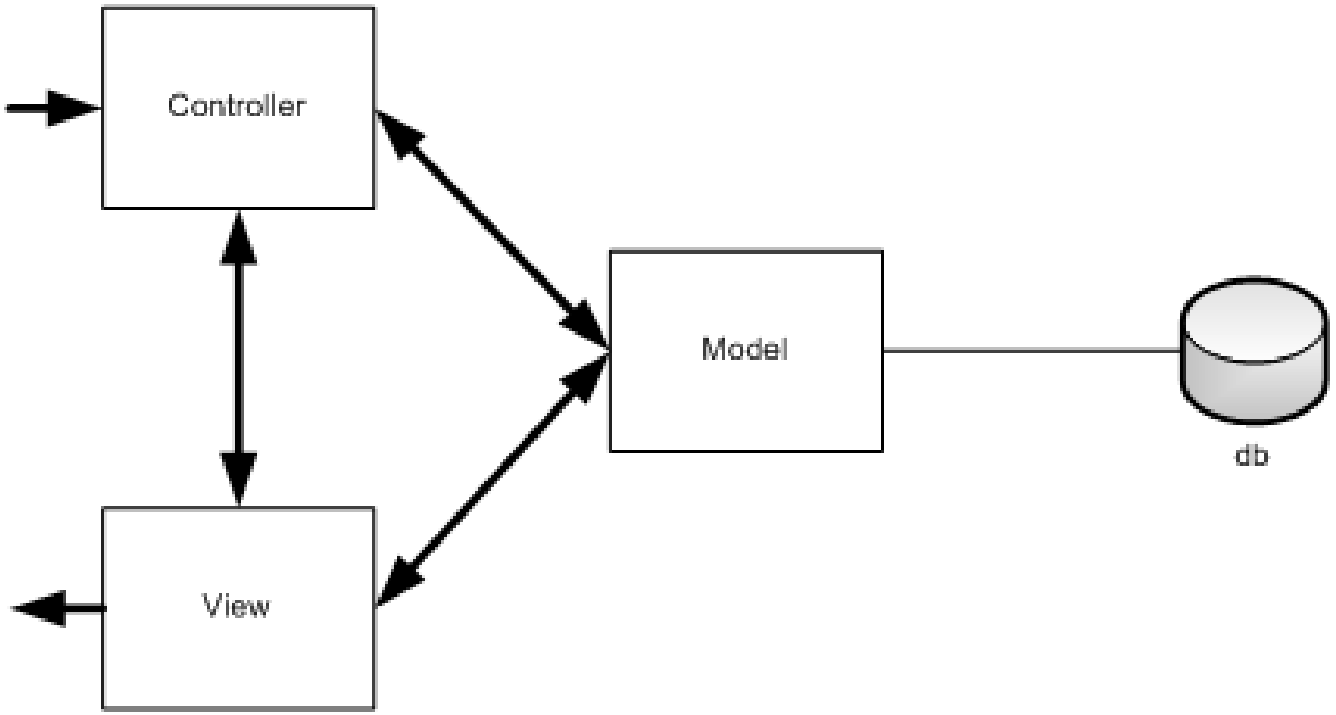
- Whenever the data in the Model is updated it will notify the registered GUI components of changes, and then the GUI components will update their displays.

- This is why we say that the data in the Model of MVC architecture is active rather than passive.

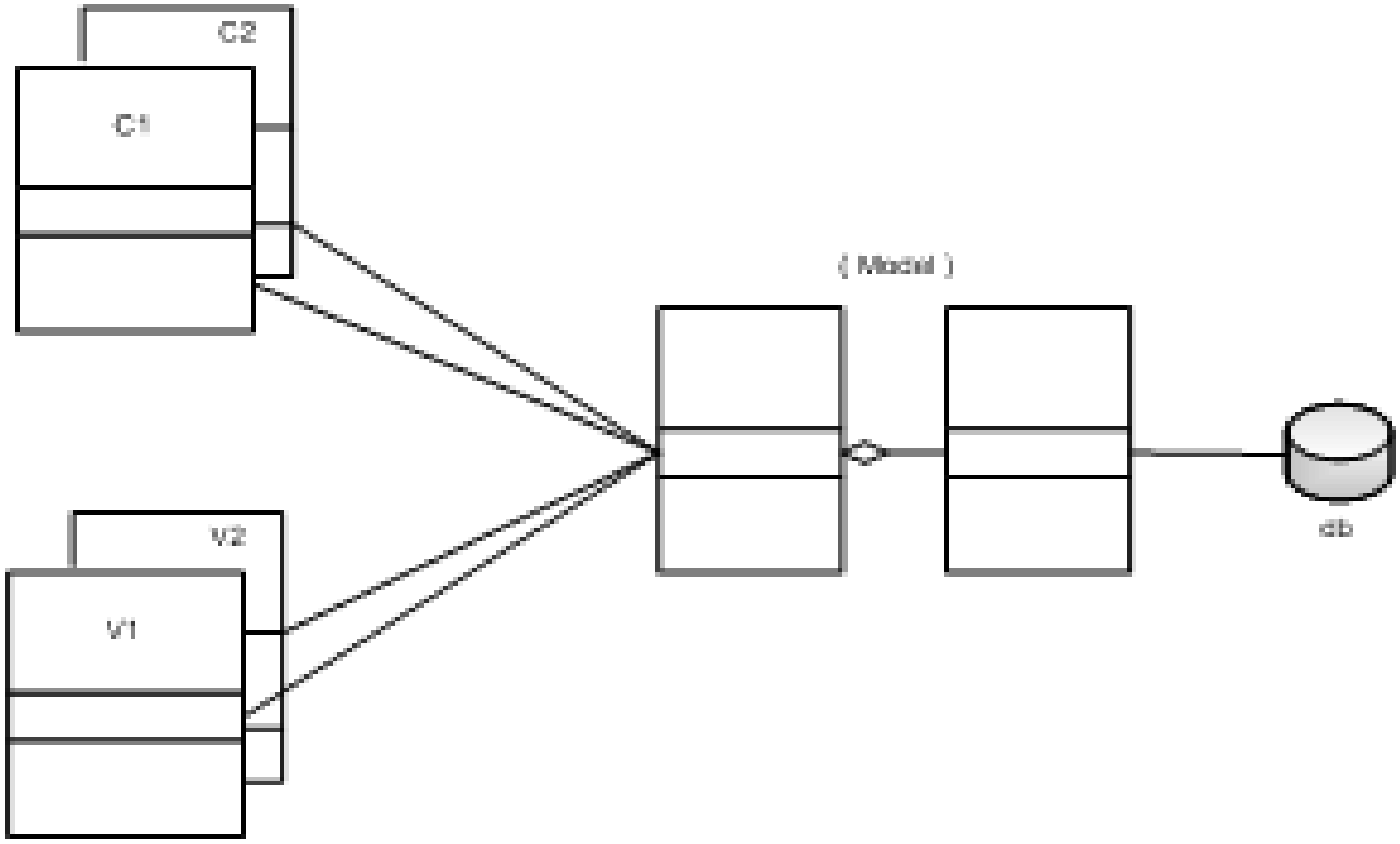
- Actually, for this specific example there is no need to have separated Model to notify the change because *actionPerformed()* can take care all necessary changes.

MVC-II

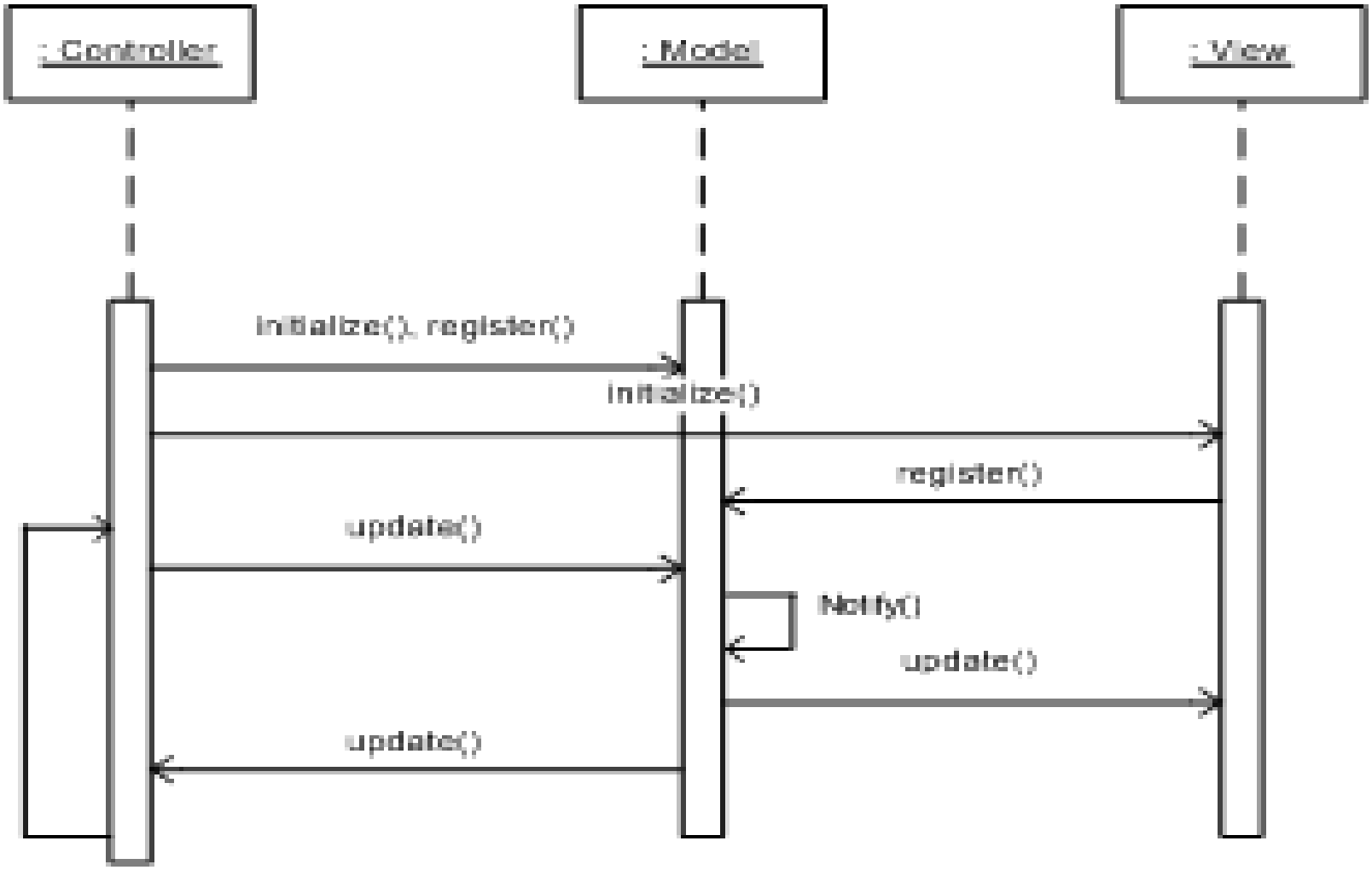
- The MVC-II is developed from MVC-I architecture.
- The model module provides all the core functionalities and data supported by a database.
- The view module displays the data.
- The controller module takes input requests, validate input data, initiates Model and View modules and their connections and dispatches tasks.
- Whenever the data is changed in Model module the View and Controller are notified for the changes.



MVC-II Architecture



Detailed MVC-II Architecture

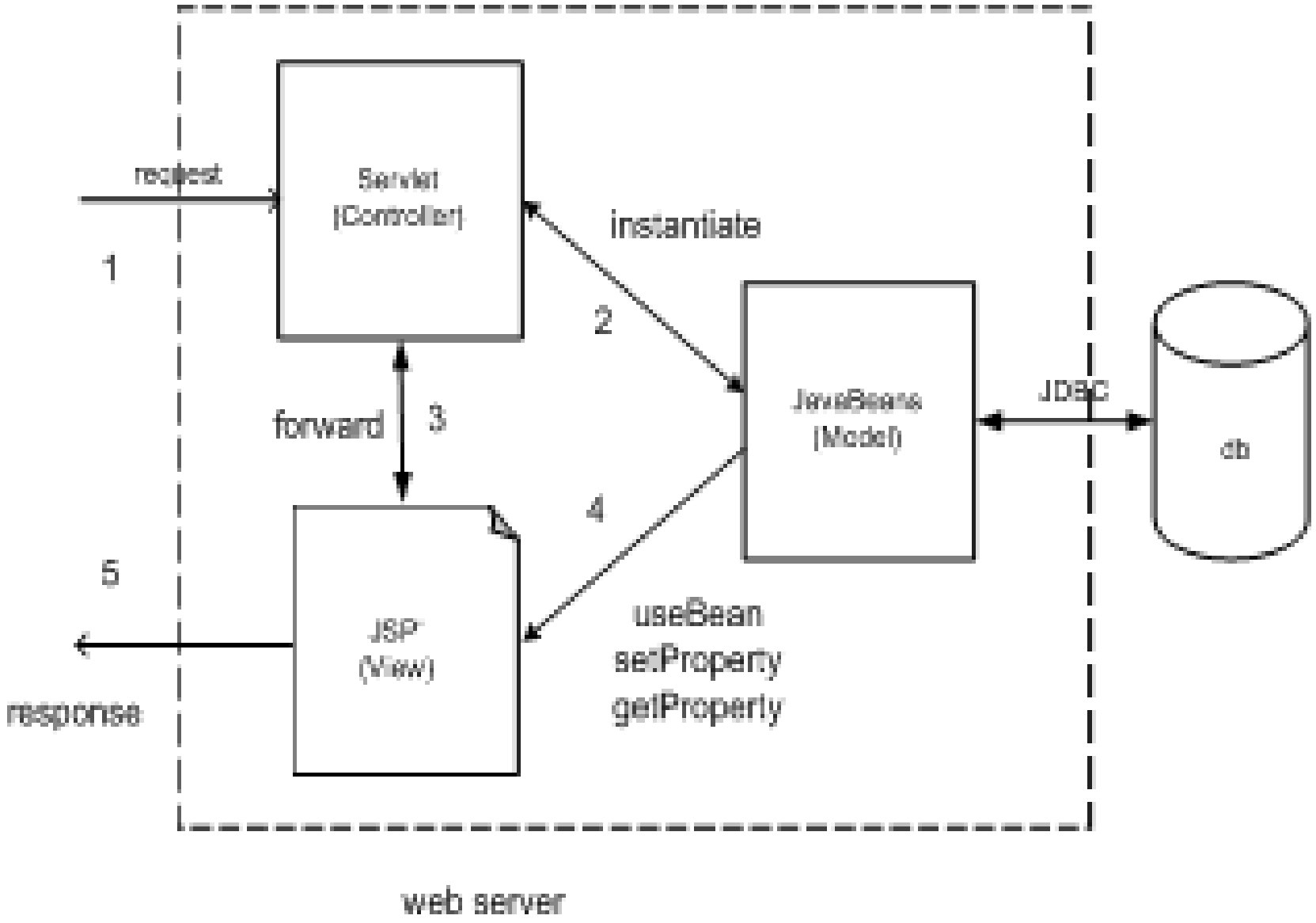


Sequence Diagram of MVC Architecture

- After clients start up the MVC application, the controller initializes the Model and View, and attaches the View and itself to the Model (this is called registration with the Model.

- Later, the Controller intercepts a user request either directly from command line or through the View interface, and forwards the request to the Model to update the data in the Model.

- The changes in the Model will trigger the Model to notify all attached or registered listeners of all changes, and the interface in the View will also be updated right way.



MVC Architecture on Java Web Platform

Applicable domain of MVC architecture

- Suitable for interactive applications where multiple views are needed for a single data model and the interfaces are prone to data changes frequently.

- Suitable for applications where there are clear divisions between controller, view, and data modules so that different professionals can be assigned to work on different aspects of such applications concurrently.

Benefits

- There are many MVC vendor framework toolkits available.
- Multiple views synchronized with same data model
- Easy to plug-in new or change interface views, thus allowing updating the interface views with new technologies without overhang the rest of the system.
- Very effective for developments if Graphics expertise professionals, programming professionals, and data base development professionals are working in a team in a designed project.

Limitations

- Does not fit well 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.

Summary

- Introduce MVC-I and MVC-II Architecture
- Applicable domain of MVC
- Benefits and Limitations of MVC