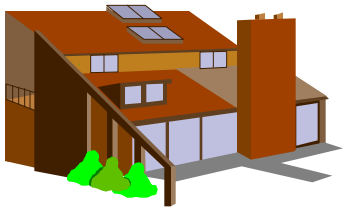
XIX. Object-Oriented Architectures

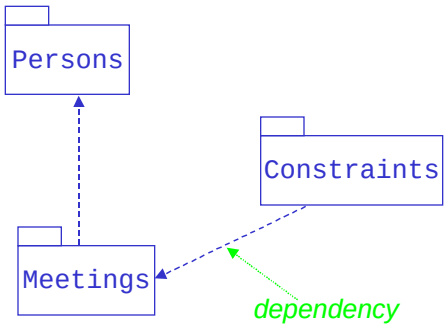
UML Packages
Client-Server vs Peer-to-Peer
Horizontal Layers and Vertical Partitions
The Model-View-Controller Architecture
Broker Architectures for Distributed Systems



Packages

- A package in UML is a grouping of elements; these elements
  - ✓ May be packages (representing subsystems or modules);
  - ✓ May be classes;
  - ✓ Each element of a software architecture (subsystem, module or class) is owned by a single package;
  - ✓ Packages may reference other packages.
- There are many criteria to use in decomposing a software system into packages:
  - ✓ Ownership -- who is responsible from which diagrams;
  - ✓ Application -- each application has its own obvious partitions; e.g., a university dept model may be partitioned into staff, courses, degree programmes,…
  - ✓ Clusters of classes used together, e.g., course, course description, instructor, student,…

A Package Diagram

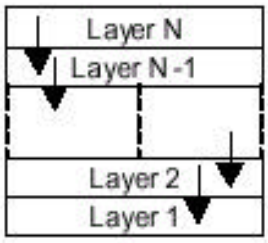


- A ***dependency*** means that if you change a class in one package (`Meetings`), you ***may*** have to change something in the other (`Constraints`).
- The concept is similar to compilation dependencies.
- It's desirable to minimize dependency cycles, if at all possible.

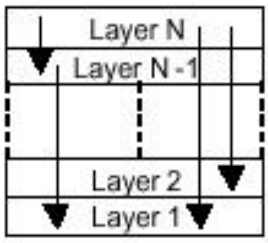
Decomposition into Subsystems

- A software system may be decomposed into ***horizontal layers***, and/or ***vertical partitions***.
- For a horizontal layer decomposition, each layer corresponds to one or more subsystems, and each layer uses services provided by the layers below it.
- Layered architectures have two forms:
  - ✓ ***closed architecture*** - each layer only uses services of the layer immediatebelow;
  - ✓ ***open architecture*** - a layer can use services from any lower layer.

Closed vs Open Layered Architecture



Closed architecture—messages may only be sent to the adjacent lower layer.

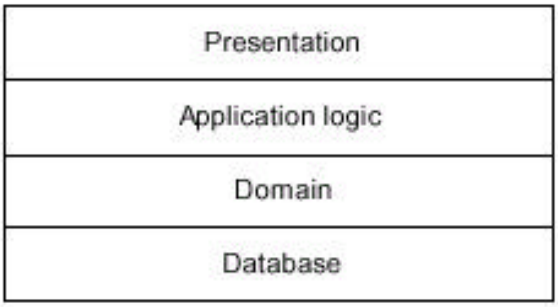


Open architecture—messages can be sent to any lower layer.

Closed vs Open Layered Architectures

- Closed layered architectures
  - ✓ Minimize dependencies between layers and reduce the impact of a change to the interface of any one layer.
- Open layered architectures
  - ✓ Lead to more compact code, since the services of all lower layers can be accessed directly without the need for extra program code to pass messages through each intervening layer;
  - ✓ Break the encapsulation of layers, increase dependencies between layers and increase the complexity of changes to the system.

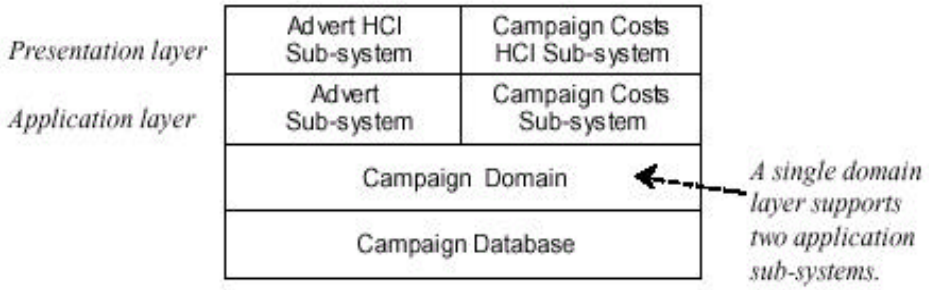
Four-Layer Architectures for Information Systems



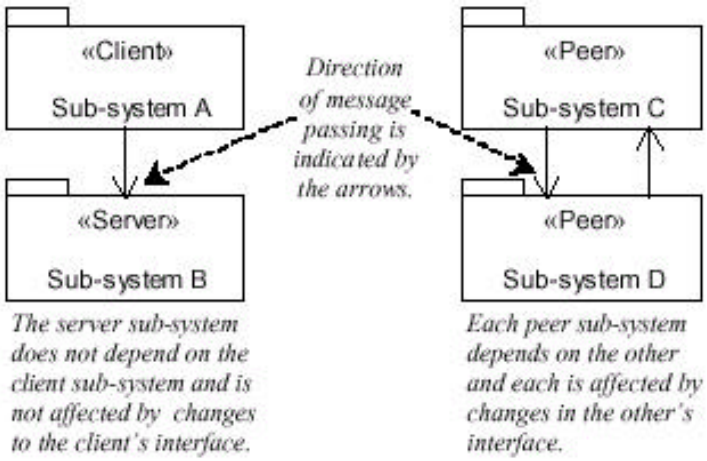
This is a variation of the 3-tier architecture we discussed earlier

Vertical Partitioning

- Now the idea is to partition each layer into subsystems.
- Partitioning identifies ***weakly coupled*** subsystems within a layer.
- Each partition provides a self-contained service for the rest of the system.



Styles of Communication Between Subsystems



The Model View Controller (MVC) Architecture

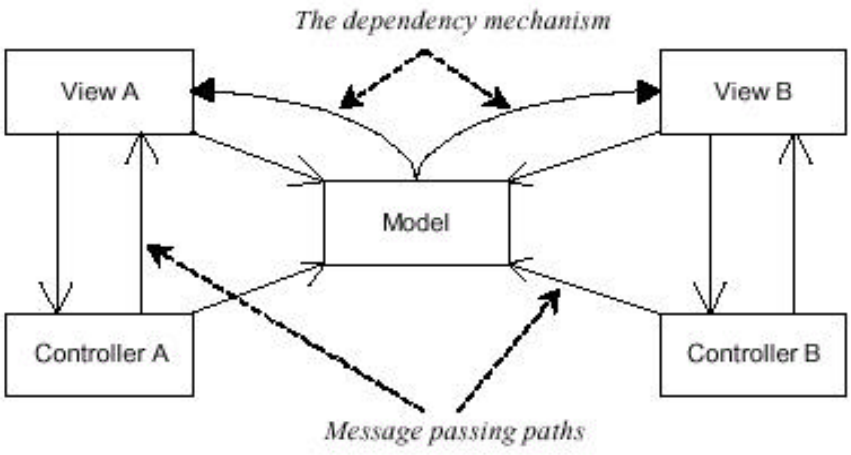
- First used with Smalltalk but has since become widely used as an architecture for object-oriented software systems.
- Capable of supporting user requirements that are presented through differing interface styles
- Aids maintainability and portability
- This architecture is best suited for software systems where user interfaces play an important role.

The MVC Architecture

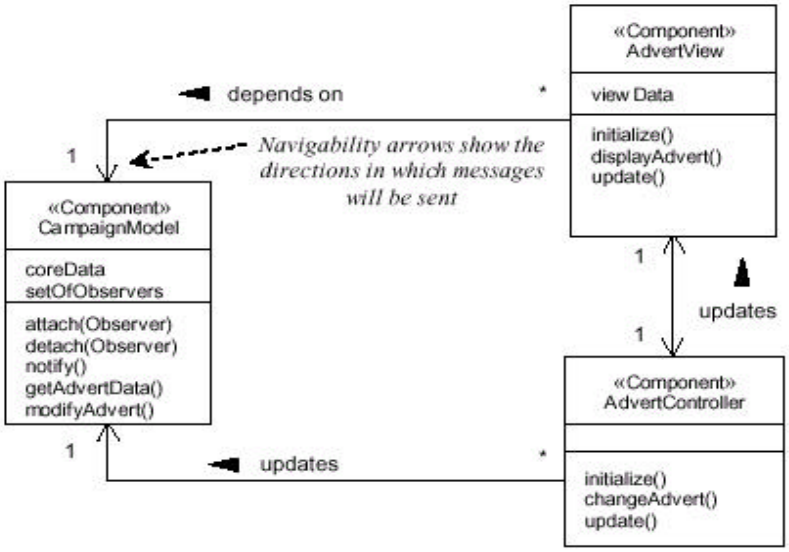
Consists of subsystems which are classified into one of the following three types:

- ***Model*** -- provides the main functionality of the application and is aware of each of its dependent view and controller components.
- ***View*** -- each view corresponds to a particular style and format of presentation of information to the user.
  - It retrieves data from the model and updates its presentations when data has been changed in one of the other views.
  - It creates its own associated controller;
- ***Controller*** -- accepts user input in the form of events that trigger the execution of operations within the model
  - These may cause changes to the model, and in turn may trigger updates in all views ensuring that they are all up to date.
- ***Dependency Mechanism***: enables the model to inform each view that the model data has changed and as a result the view must update itself

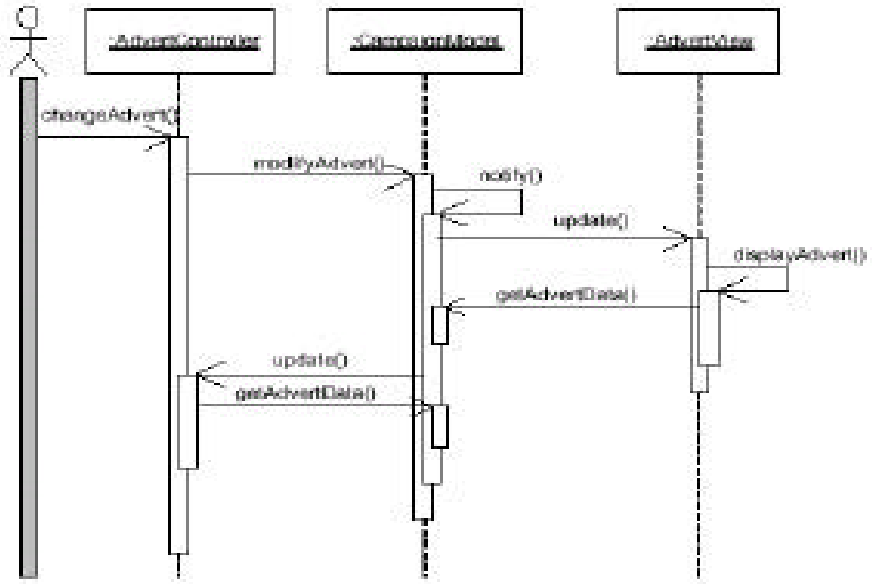
Model View Controller (MVC)



Responsibilities of MVC Components



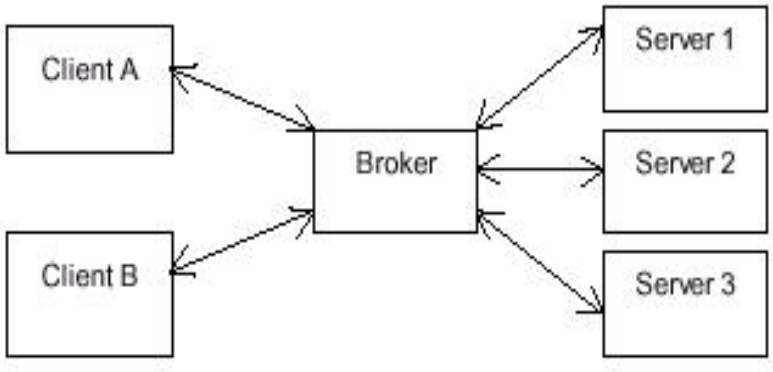
MVC Component Interaction



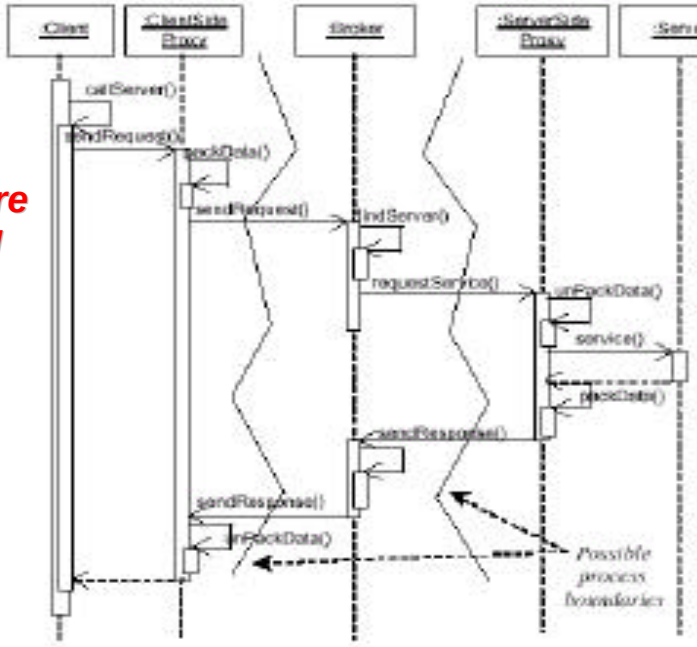
Broker Architectures for Distributed Systems

- A broker increases the flexibility of the system by decoupling the client and server components
  - Each client sends its requests to the broker rather than communicating directly with the server component
  - The broker then forwards the service request to an appropriate server
- The client need not know where the server is locate (it may be in local or remote computer)
- Only the broker needs to know the location of the servers that it handles

Simplified Broker Architecture



Broker Architecture for Local Server



Threading and Concurrency

- Each independent flow of control can be modelled as an active object that represents a process or thread that can initiate control activity.
  - ✓ A ***process*** is a heavyweight flow (known to the operating systems itself) that can execute concurrently with other processes
  - ✓ A ***thread*** is a lightweight flow that can execute concurrently with other threads within the same process.
- Dynamic model of the design identifies concurrent parts of the system:
  - ✓ Sequence diagrams imply sequential threads of execution - sequences of messages that invoke each other procedurally;
  - ✓ State and activity diagrams can model concurrent execution where different event sequences can lead to concurrent execution.

Additional Readings

- [Booch99] Booch, G. Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide*. Chapter 22. Addison-Wesley.
- [Rumbaugh91] Rumbaugh, J et al. *Object-Oriented Modeling and Design.* Chapter 9, Prentice-Hall.