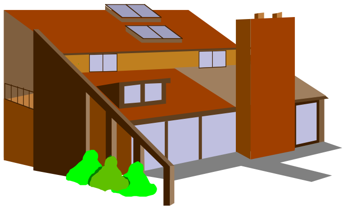
XVIII. Software Architectures

Software Architectures
UML Packages
Client-Server vs Peer-to-Peer
3-Tier and 4-Tier Architectures
Horizontal Layers and Vertical Partitions
The Model-View-Controller Architecture
Broker Architectures for Distributed Systems

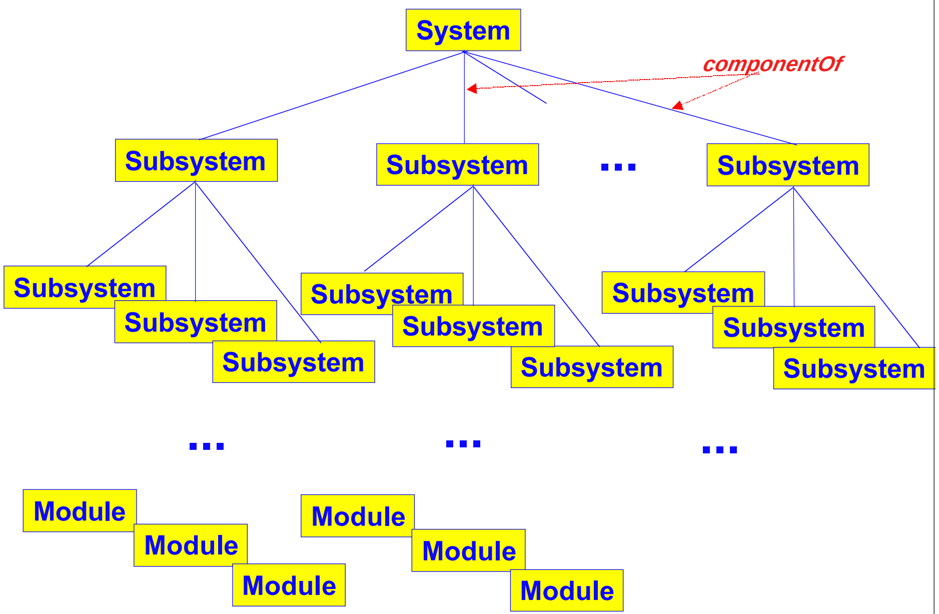


Software Architectures

- *A software architecture defines the components of a software system and how they use each other's functionality and data.*
- *For example, the **client-server** architecture consists of **servers**, which support some kind of service, and **clients** which request and use server services. With a client-server architecture, an information system need not be seen as a monolithic program.*
- *Instead, input/output functions are placed on clients, running on PCs and workstations; data storage is assigned to a server, implemented in terms of a DBMS (e.g., DB2, Ingres, Sybase or Oracle) and placed on a mainframe or mini. Consistency checking is located with the server, applications are located with clients.*
- ***Thick servers** offer a lot of functionality, **thin** ones little.*
- ***Thick clients** have their own services, **thin** ones get almost everything from servers.*
- *In these lecture notes, we emphasize **object-oriented architectures**.*

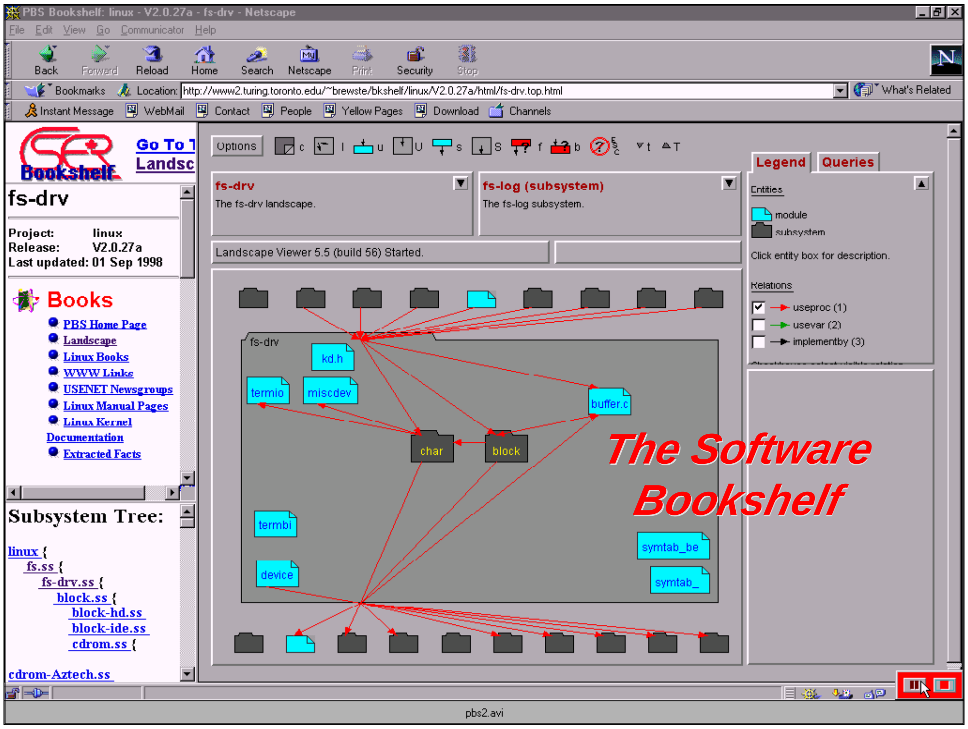
Subsystems

- *A **subsystem** is a component of a system or of another subsystem.*
- ***Modules** are atomic subsystems (which are not further decomposed into subsystems.)*
- *It's useful to subdivide a software system into subsystems*
  - ✓ *For better-managed software development;*
  - ✓ *For improved reuse potential (through components);*
  - ✓ *For improved portability (platform-specific code isolated to particular subsystems.)*
  - ✓ *For easier maintenance.*
- *Each subsystem has a well-defined interface with respect to the rest of the system.*



Components and Connectors

- The architecture shown in the previous slide is one example of a software architecture where the nodes represent subsystems or modules and the connectors between them describe "componentOf" relationships.
- There are many others kinds of connectors that can be used, such as:
  - ✓ *Uses* -- one component uses data defined in another component;
  - ✓ *Calls* -- one component calls methods defined in another component;
  - ✓ *I/O* -- the output of one component is fed as input to another;



fs-drv

Project: linux
Release: V2.0.27a
Last updated: 01 Sep 1998

fs-drv
The fs-drv landscape.

fs-log (subsystem)
The fs-log subsystem.

Landscape Viewer 5.5 (build 56) Started.

The Software Bookshelf

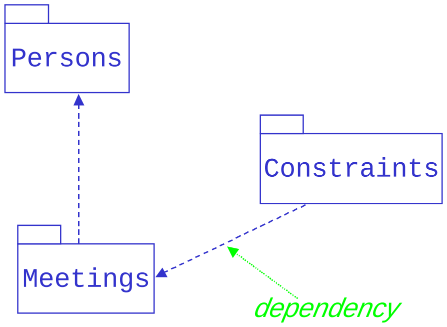
Architectural Styles

- It is useful to classify software architectures into classes of ***architectural styles***.
- For example, the client-server architecture discussed earlier is an architectural style.
- The styles we'll discuss below are as follows:
  - ✓ Pipes and filters;
  - ✓ Object-Orientation;
  - ✓ Event-Based
  - ✓ Layered;
  - ✓ Repository-Based;
  - ✓ Client-Server;
  - ✓ Three-Tier;
  - ✓ …more...

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

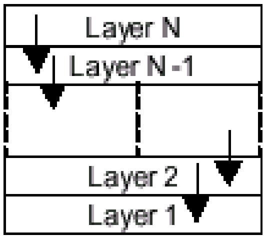


- A ***dependency*** means that if you change a class in one package (`Meetings`), you ***may*** have to change something in the other (`Constraints`).
- The concept is similar to compilation dependencies.
- It's desirable to minimize dependency cycles, if at all possible.

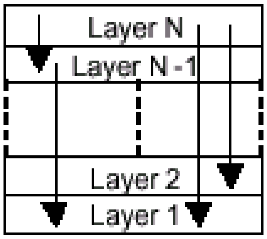
Decomposition into Subsystems

- 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.

Client Server Architectures

- A client server architecture consists of **service consumers** (clients) and **service providers** (servers). Clients and servers may or may not be running on dedicated machines.
- Information exchange between clients and servers is done through messages.
- Service requests and responses are accomplished through one of the following protocols:

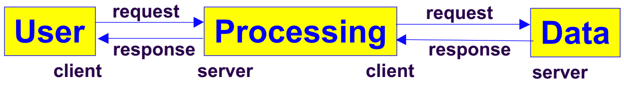
  Remote Procedure Call (RPC) -- client invokes a remotely located procedure, which is executed and the results sent to the client; RPC is widely supported;

  Remote Data Access (RDA) -- here the invoked procedure is a database query (say, in SQL) and the response is an often large set of data; supported by DBMS vendors;

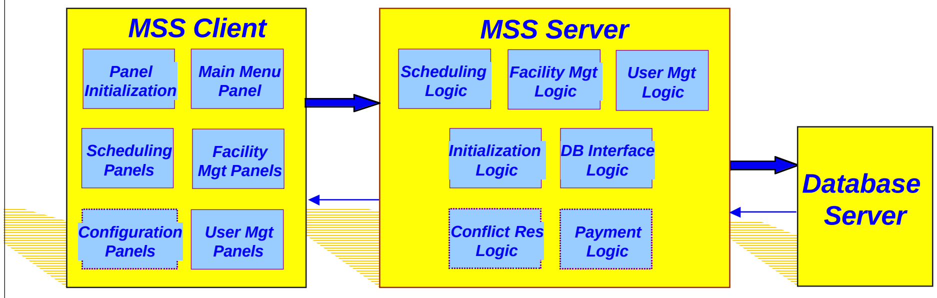
  Queued Message Processing -- here requests are queued and processed whenever possible.

Three-Tier Client Server Architectures

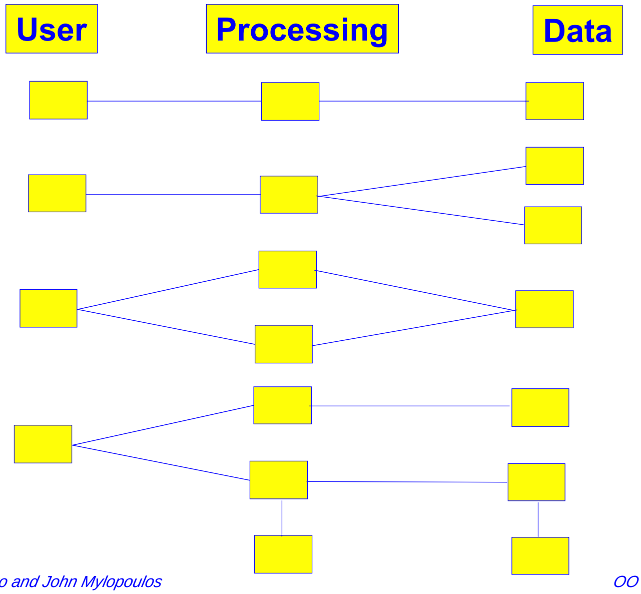
- Used widely in industry



- E.g., architecture for a meeting scheduling system (MSS)



Many Possible Variations



Web-Based Software Architectures

- *These are client server too, but they are based on WWW technologies.*
- *Such architectures are becoming very popular because of static HTML-based applications, but also dynamic ones, such as those that involve Ecommerce.*

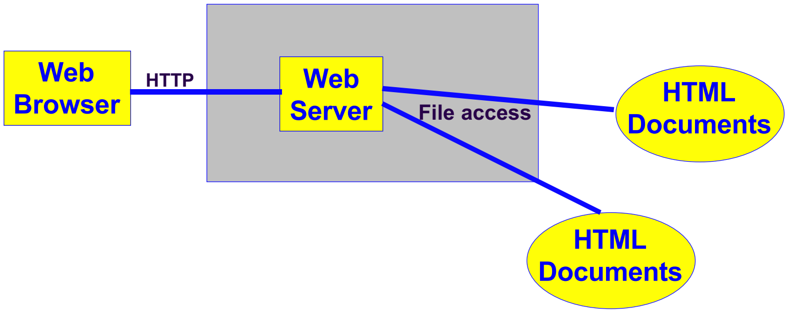
  *HTTP -- **HyperText Transfer Protocol**, used to transfer hypertext documents over the internet;*

  *HTML -- **HyperText Markup Language**, used to define hypertext documents;*

  *CGI -- **Common Gateway Interface** is a program (e.g., a unix shell script, or a perl script)*

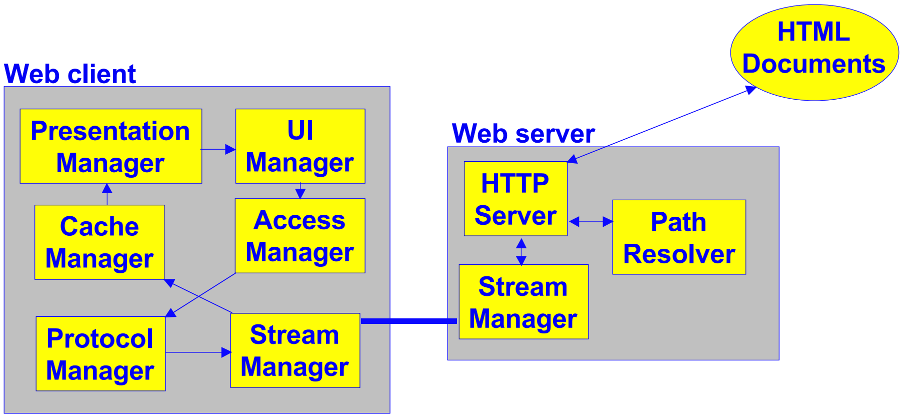
  CGI scripts are programs that reside on a web server and are executed with a click to retrieve data, generate graphics etc.

Static HTML-Based Architecture



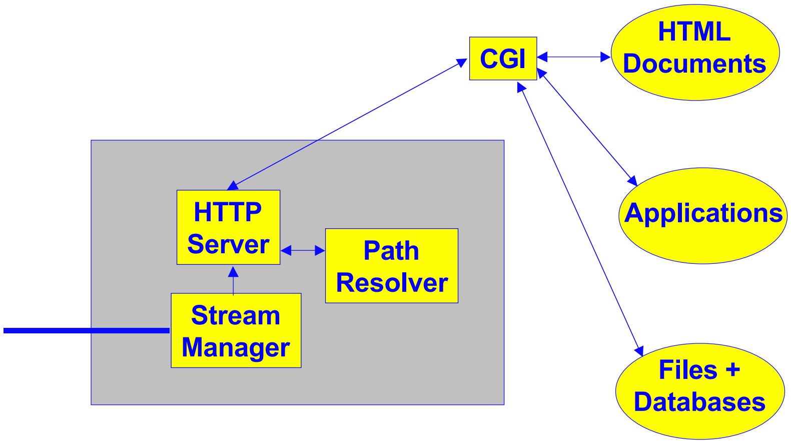
- *This architecture basically retrieves and displays HTML documents that reside on the web server site.*

More Detailed Static Architecture



- *Arrows indicate data and/or control flow.*

Dynamic HTML-Based Architecture



CGI

HTML Documents

Applications

Files + Databases

HTTP Server

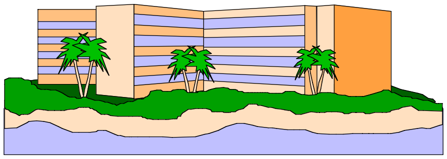
Path Resolver

Stream Manager

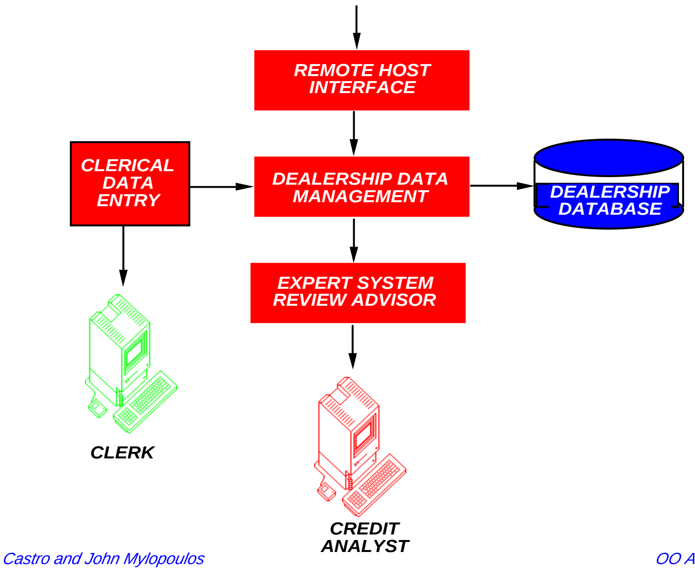
- The CGI gateway serves as demon which dispatches a request, dealt with by an application or a database server.

Document Interchange Example

- ANALYST, the General Motors Dealer Review Advisor.
- Assists credit analysts in 230 GM Acceptance Corporation branch offices analyzing dealership operations in order to decide on credit applications.
- Offers many benefits, including faster reviews, reduced training of personnel and consistency in decision-making.
- Uses an expert system, integrated into vast, conventional data processing architecture



ANALYST Local Architecture



REMOTE HOST INTERFACE

CLERICAL DATA ENTRY

DEALERSHIP DATA MANAGEMENT

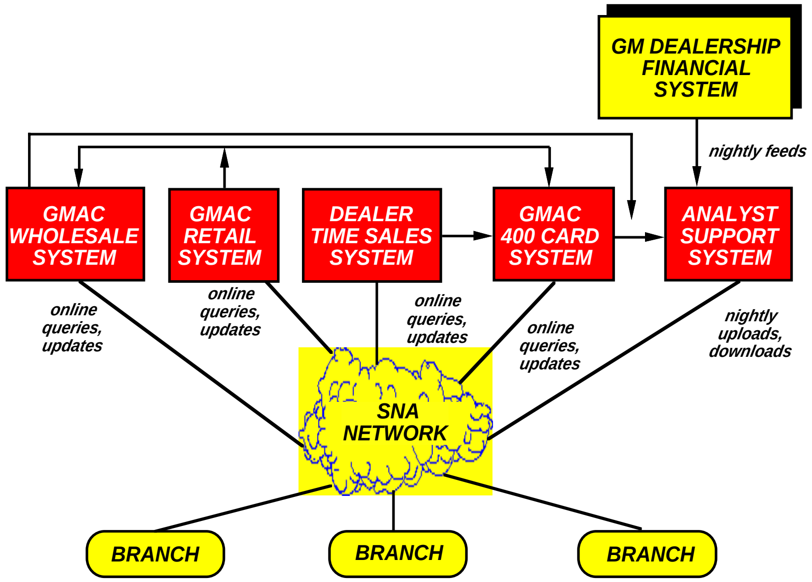
DEALERSHIP DATABASE

EXPERT SYSTEM REVIEW ADVISOR

CLERK

CREDIT ANALYST

ANALYST Global Architecture



GM DEALERSHIP FINANCIAL SYSTEM

nightly feeds

GMAC WHOLESALE SYSTEM

GMAC RETAIL SYSTEM

DEALER TIME SALES SYSTEM

GMAC 400 CARD SYSTEM

ANALYST SUPPORT SYSTEM

online queries, updates

online queries, updates

online queries, updates

online queries, updates

nightly uploads, downloads

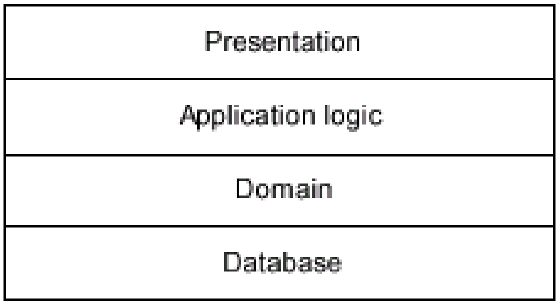
SNA NETWORK

BRANCH

BRANCH

BRANCH

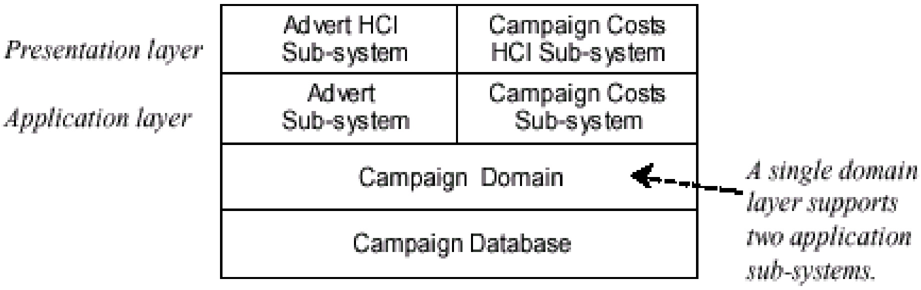
Four-Layer Architectures for Information Systems



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

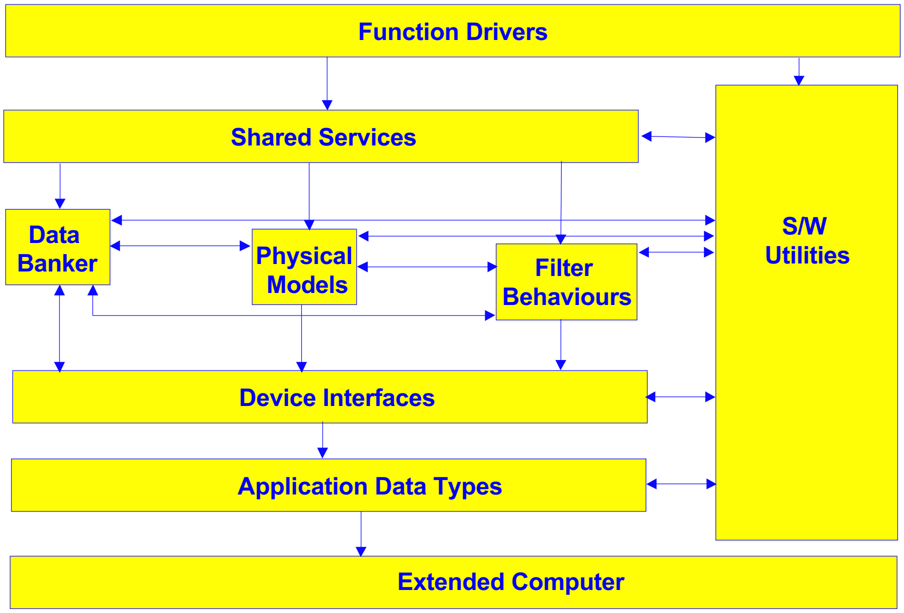
Vertical Partitioning

- 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.



A single domain layer supports two application sub-systems.

Architecture for the A-7E Aircraft



Function Drivers

Shared Services

Data Banker

Physical Models

Filter Behaviours

S/W Utilities

Device Interfaces

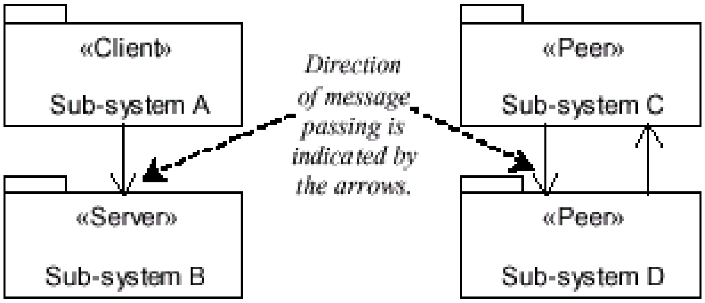
Application Data Types

Extended Computer

Notes on the A-7E Architecture

- ❑ This is a "uses" architecture, I.e., shows which component uses resources in another component.
- ❑ Modules in the different components of the architecture:
  - ✓ Extended computer: virtual memory module, parallelism module, timer module;
  - ✓ Device interfaces: air data module, audible signal device module, Doppler radar set module,…;
  - ✓ Function driving module: flight information display module,, panel module, ground test module,…;
  - ✓ Application data types: numeric, state transition data types;
  - ✓ Data banker module: singular values module, complex event module,…;
  - ✓ Physical model: aircraft motion module, earth characteristics module, human factors module;
  - ✓ Software utilities: powerup module.

Styles of Communication: Client-Server vs Peer-to-Peer



Direction of message passing is indicated by the arrows.

The server sub-system does not depend on the client sub-system and is not affected by changes to the client's interface.

Each peer sub-system depends on the other and each is affected by changes in the other's interface.

The Model View Controller (MVC) Architecture

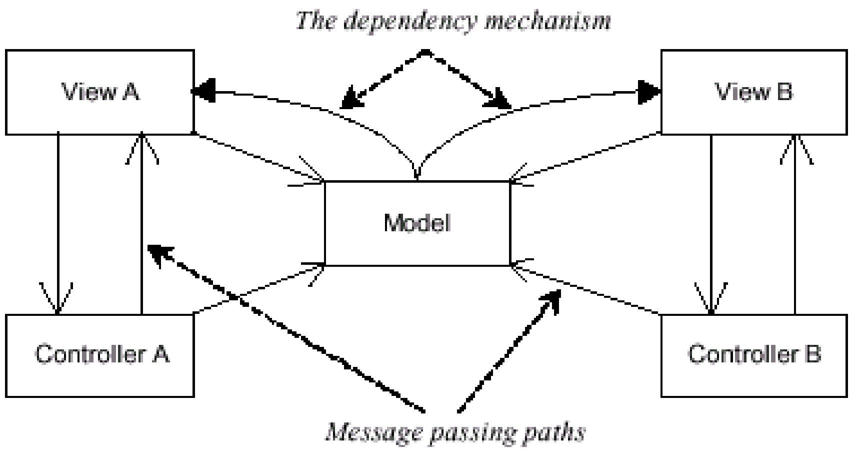
- First used with Smalltalk but has since become widely used as an architecture for object-oriented software systems.
- Capable of supporting user requirements that are presented through differing interface styles
- Aids maintainability and portability
- This architecture is best suited for software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

The MVC Architecture

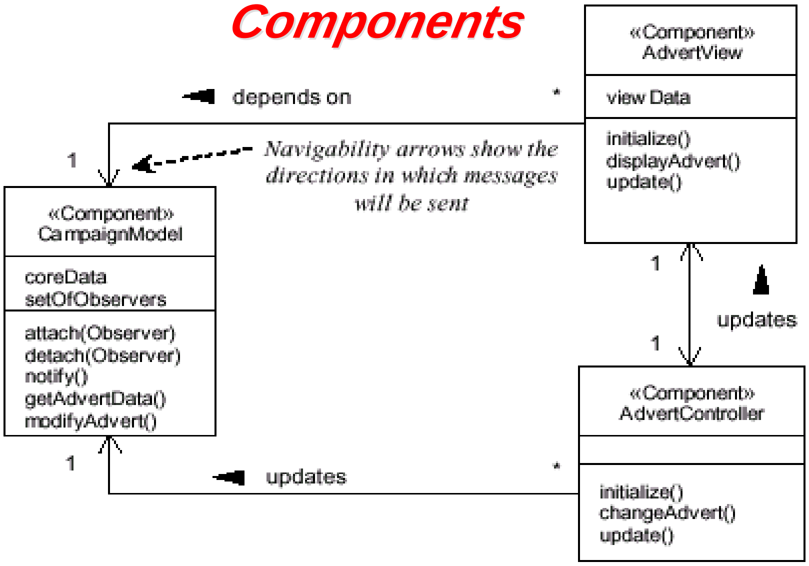
Consists of subsystems which are classified into one of the following three types:

- ***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

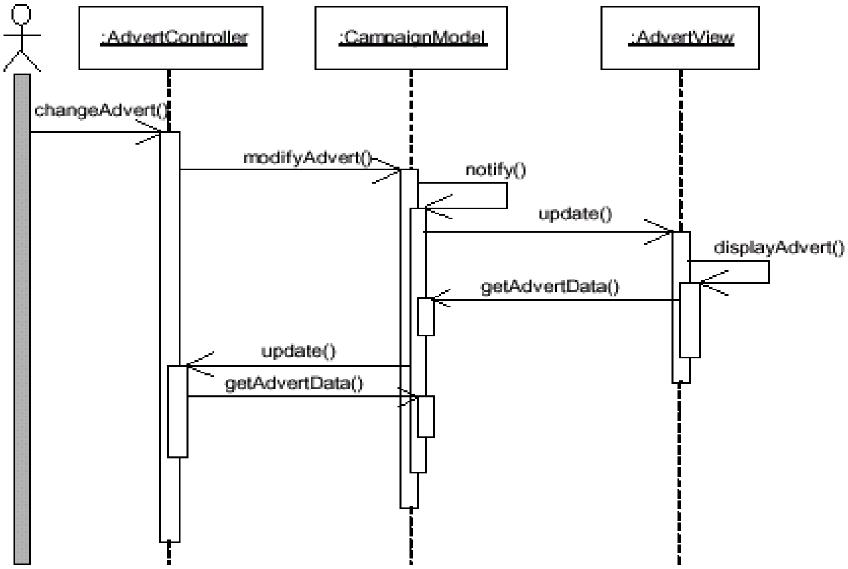
Model View Controller (MVC)



Responsibilities of MVC Components



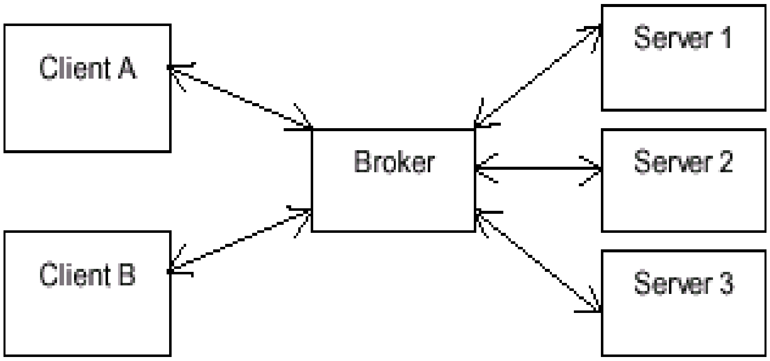
MVC Component Interaction



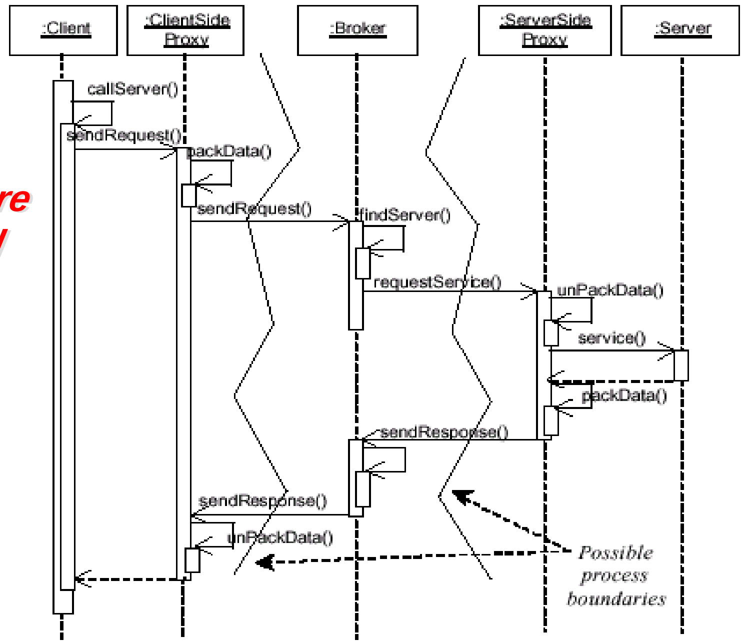
Broker Architectures for Distributed Systems

- 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.

Summary

- Architectural **system** (I.e., hardware and software) design addresses issues of hardware and software location, interconnectivity, distribution of I/O processes, data stores and application processes.
- Architectural system design is the most important step of system design and can, literally, make or break an information system development project.

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.