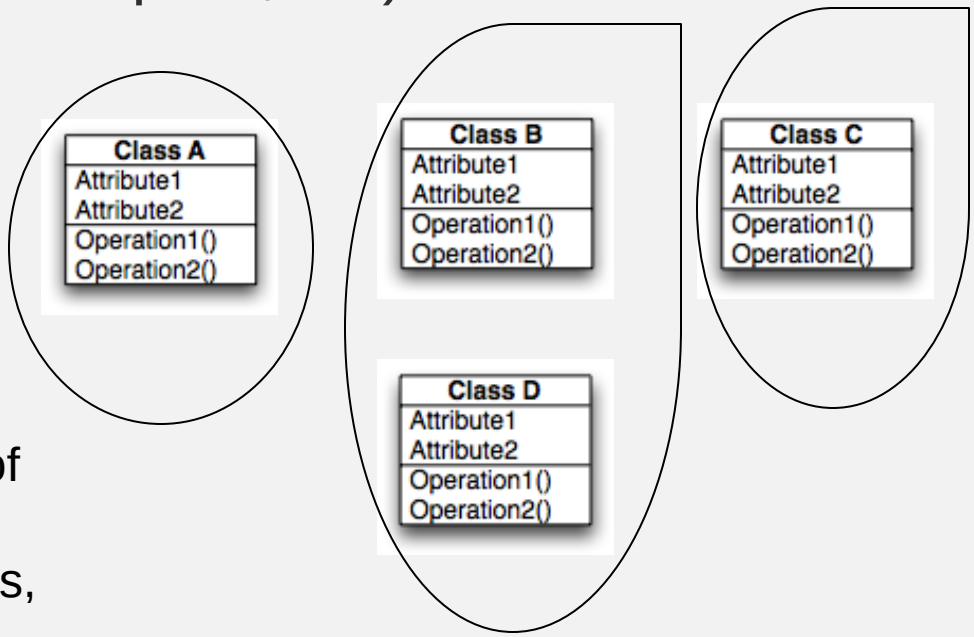
Design: Architecture Styles

Lecture 11

Subsystem decomposition

Subsystem decomposition: Identification of subsystems, services, and their association to each other (hierarchical, peer-to-peer, etc)



interaction should be within subsystems

Collection of classes, associations, operations, events and constraints

1. objects identified in one use case into the same subsystem.
2. Create a dedicated subsystem for objects used for moving data among subsystems.
3. Minimize the number of associations crossing subsystem boundaries.
4. All objects in the same subsystem should be functionally related

Architectural Style vs Architecture

Architectural Style: A pattern for a subsystem decomposition

Software Architecture: Instance of an architectural style.

Examples of Architectural Styles

Client/Server

Peer-To-Peer

Repository

Model/View/Controller

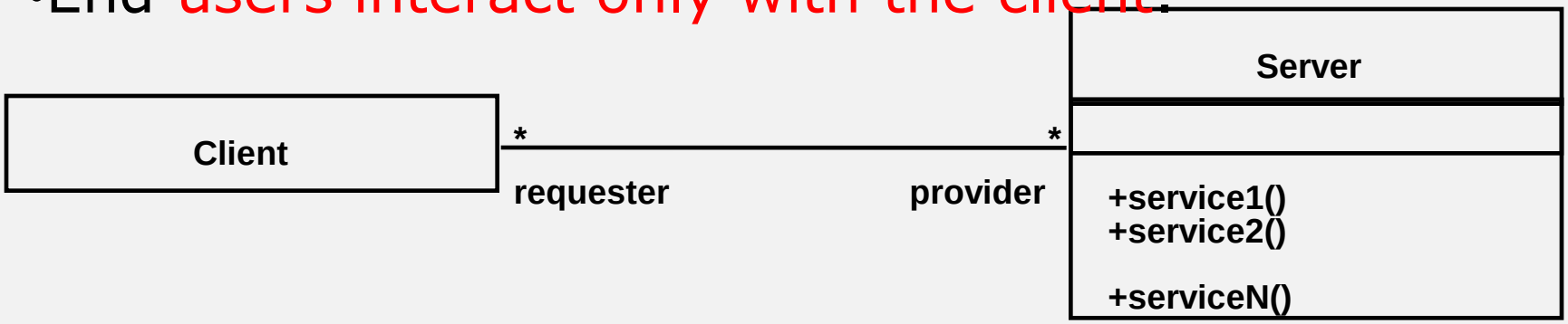
Three-tier, Four-tier Architecture

Pipes and Filters

Client/Server Architectural Style

One or many servers provide services to instances of subsystems, called clients

- Each client calls on the server, which performs some service and returns the result
  - The clients know the *interface* of the server
  - The server does not need to know the interface of the client
- The response in general is immediate
- End users interact only with the client.



Client/Server Architectures

Often used in the design of database systems

- Front-end: User application (client)
- Back end: Database access and manipulation (server)

Functions performed by client:

- Input from the user (Customized user interface)
- Front-end processing of input data

Functions performed by the database server:

- Centralized data management
- Data integrity and database consistency
- Database security

Design Goals for Client/Server Architectures

Service Portability	Server runs on many operating systems and many networking environments
Location Transparency	Server might itself be distributed, but provides a single "logical" service to the user
High Performance	Client optimized for interactive display-intensive tasks; Server optimized for CPU-intensive operations
Scalability	Server can handle large # of clients
Flexibility	User interface of client supports a variety of end devices (PDA, Handy, laptop, wearable computer)
Reliability	Server should be able to survive client and communication problems

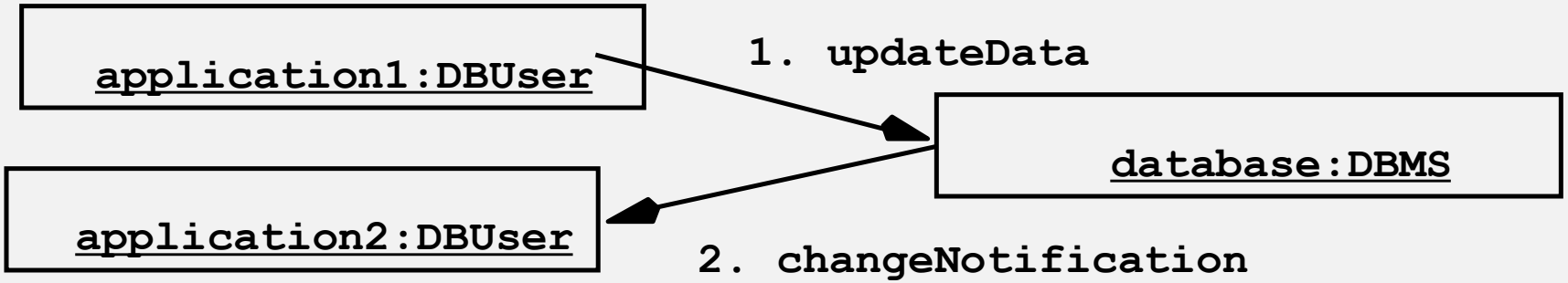
Problems with Client/Server Architectures

Client/Server systems do not provide peer-to-peer communication

Peer-to-peer communication is often needed

Example:

Database must process queries from application and should be able to send notifications to the application when data have changed

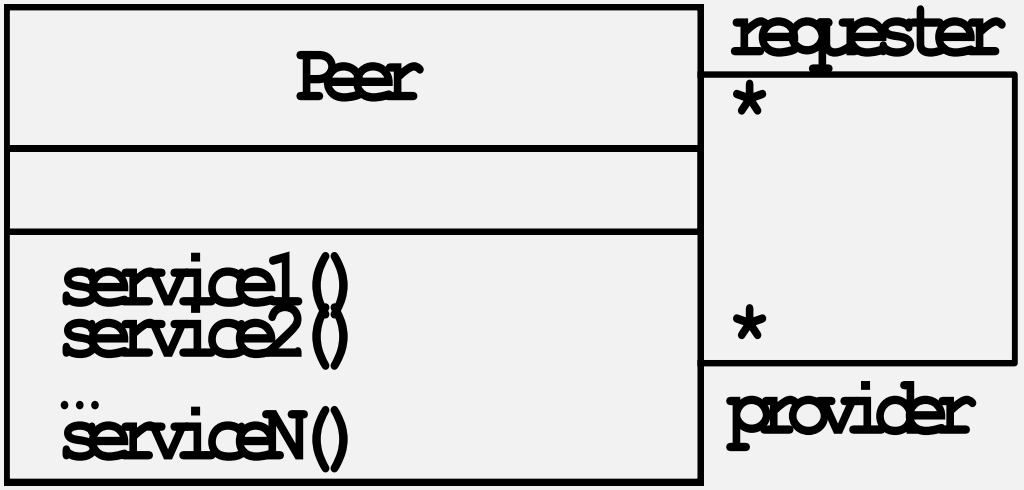


Peer-to-Peer Architectural Style

Generalization of Client/Server Architectural Style
Client can be servers and servers can be clients

Introduction a new abstraction: Peer
Client and servers can be both
How do we model this statement? With Inheritance?

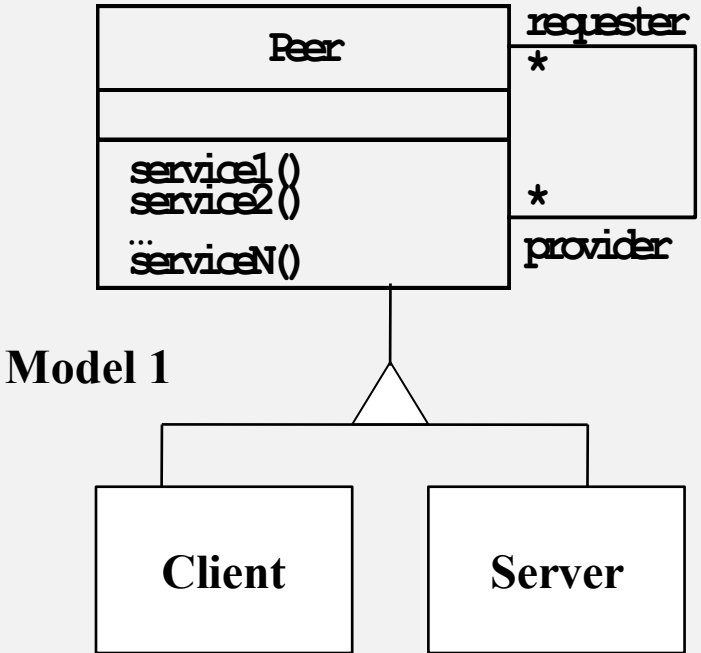
Proposal 1: "A peer can be either a client or a server"
Proposal 2: "A peer can be a client as well as a server"

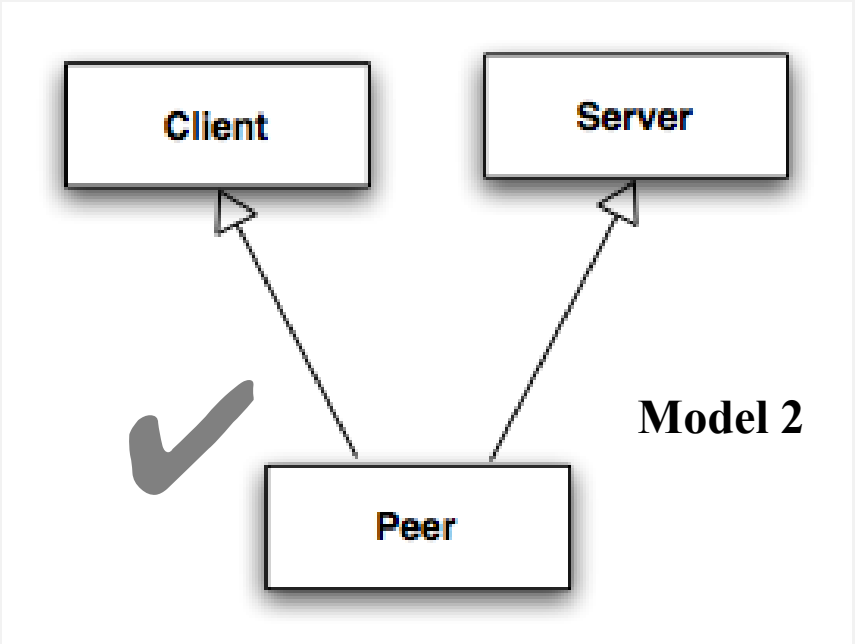


Peer-to-Peer Architectural Style

Problem statement
"Clients can be server and servers can be client"

Which model is correct?
Model 1: "A peer can be either a client or a server"
Model 2: "A peer can be a client as well as a server"

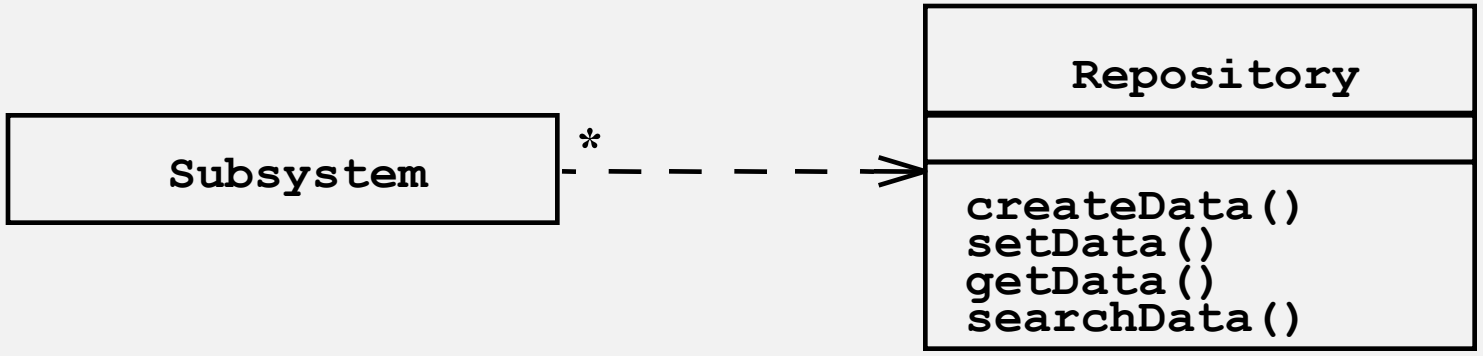




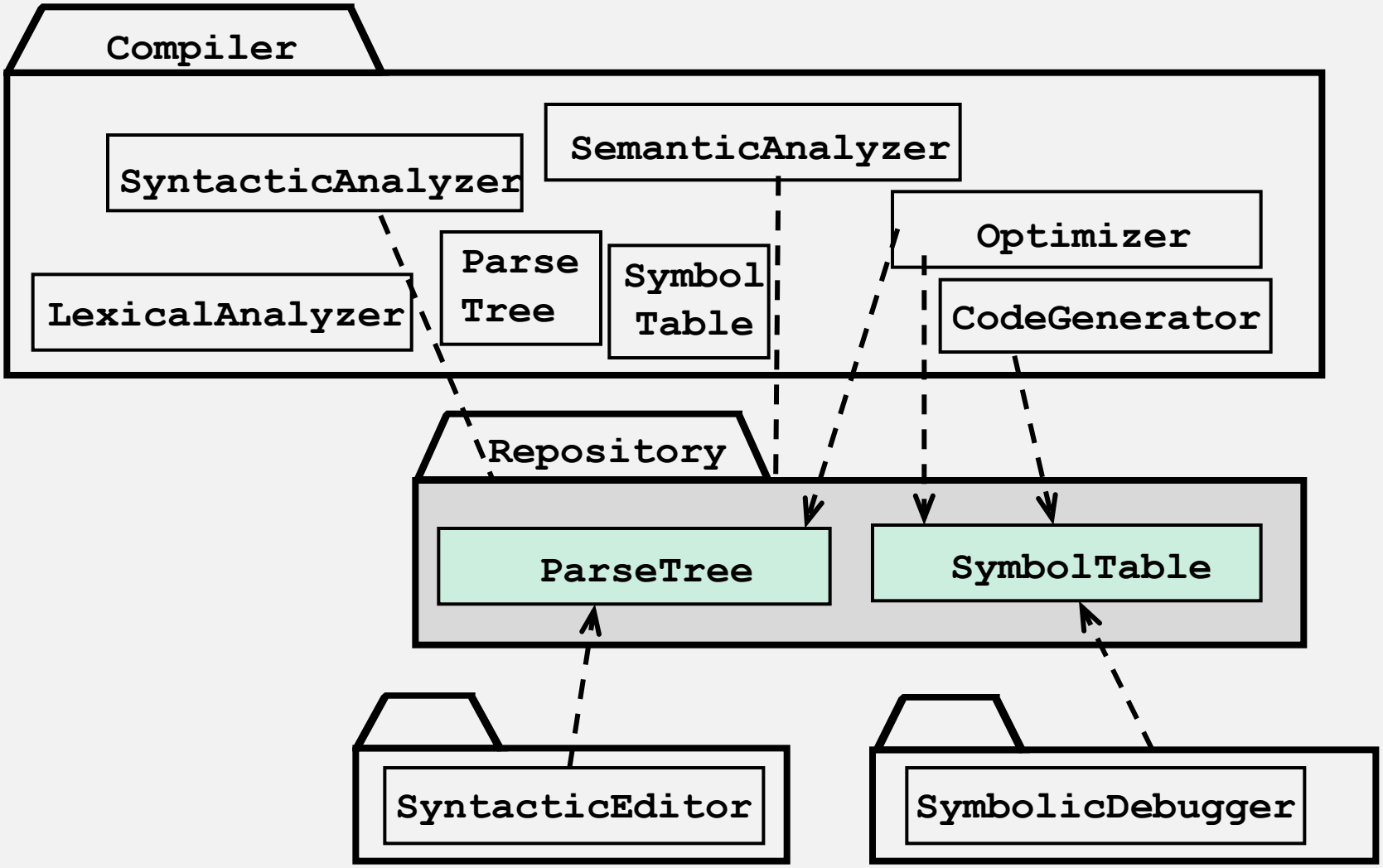
Repository Architectural Style

Subsystems access and modify data from a single data structure called the repository

- Subsystems are loosely coupled (interact only through the repository)
- Control flow is dictated by the repository through triggers or by the subsystems through locks and synchronization primitives



Repository Architecture Example: Incremental Development Environment (IDE)



Model-View-Controller Architectural Style

Problem: In systems with high coupling changes to the user interface (boundary objects) often force changes to the entity objects (data)

- The user interface cannot be re-implemented without changing the representation of the entity objects
- The entity objects cannot be reorganized without changing the user interface

Solution: Decoupling! The model-view-controller architectural style decouples data access (entity objects) and data presentation (boundary objects)

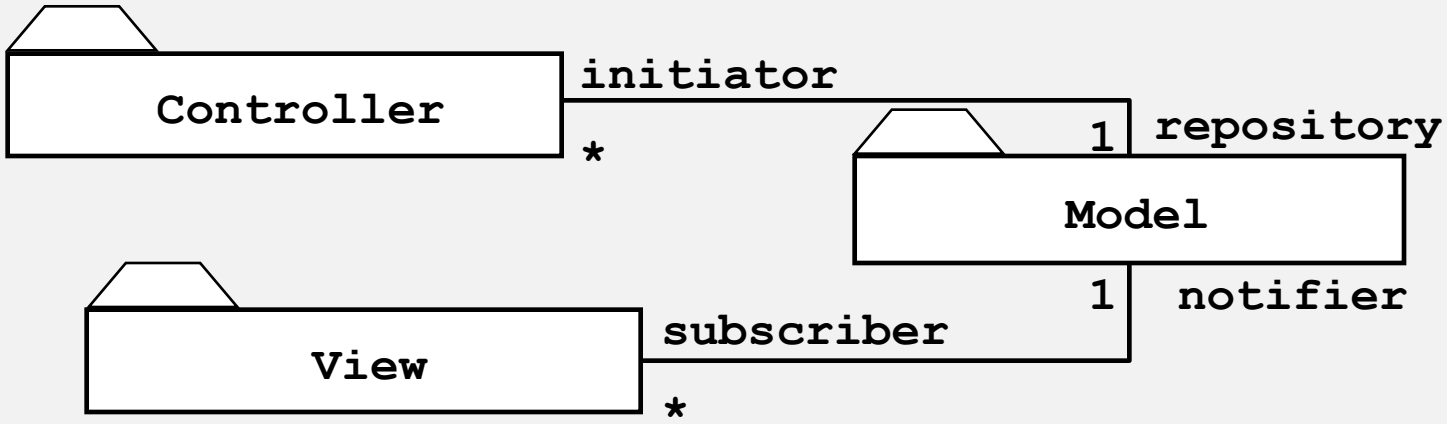
- The Data Presentation subsystem is called the View
- The Data Access subsystem is called the Model
- The Controller subsystem mediates between View (data presentation) and Model (data access)

Often called MVC.

Model-View-Controller Architectural Style

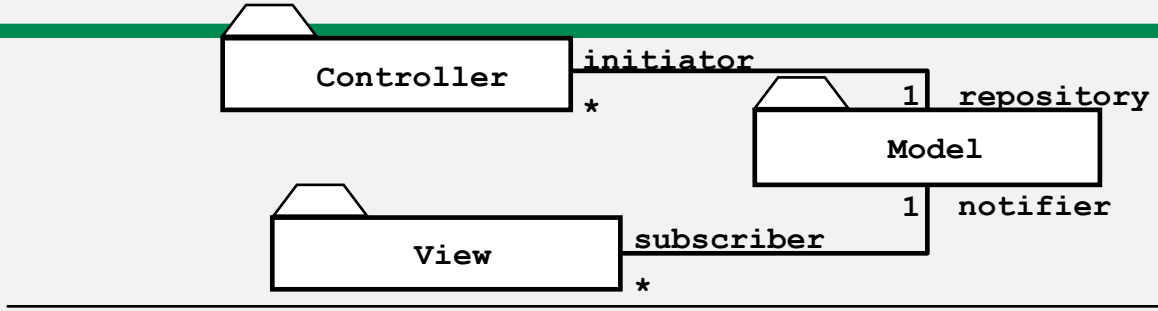
Subsystems are classified into 3 different types

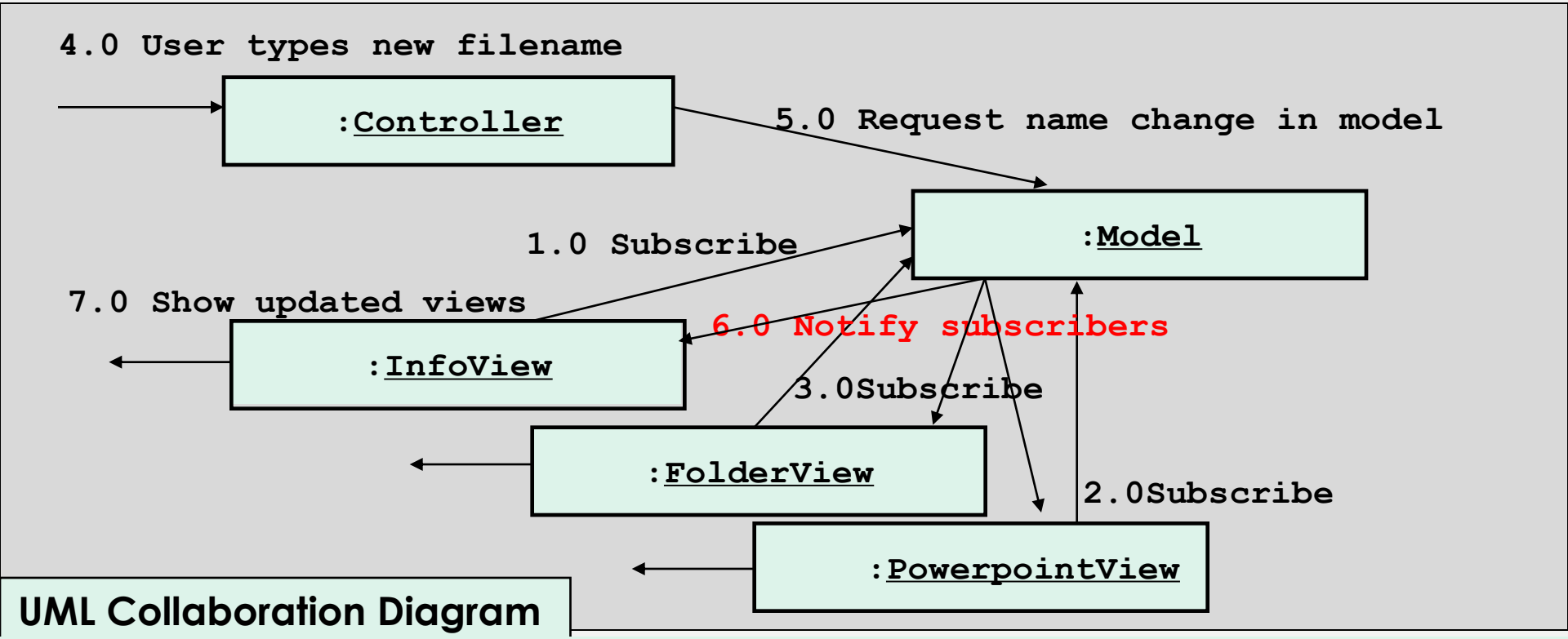
- Model subsystem: Responsible for application domain knowledge
- View subsystem: Responsible for displaying application domain objects to the use
- Controller subsystem: Responsible for sequence of interactions with the user and notifying views of changes in the model



Example: Modeling the Sequence of Events in MVC

UML Class Diagram





UML Collaboration Diagram

A Collaboration Diagram is an instance diagram that visualizes the interactions between objects as a flow of messages. Messages can be events or calls to operations

3-Layer-Architectural Style
3-Tier Architecture

Definition: 3-Layer Architectural Style

An architectural style, where an application consists of 3 hierarchically ordered subsystems

A user interface, middleware and a database system

The middleware subsystem services data requests between the user interface and the database subsystem

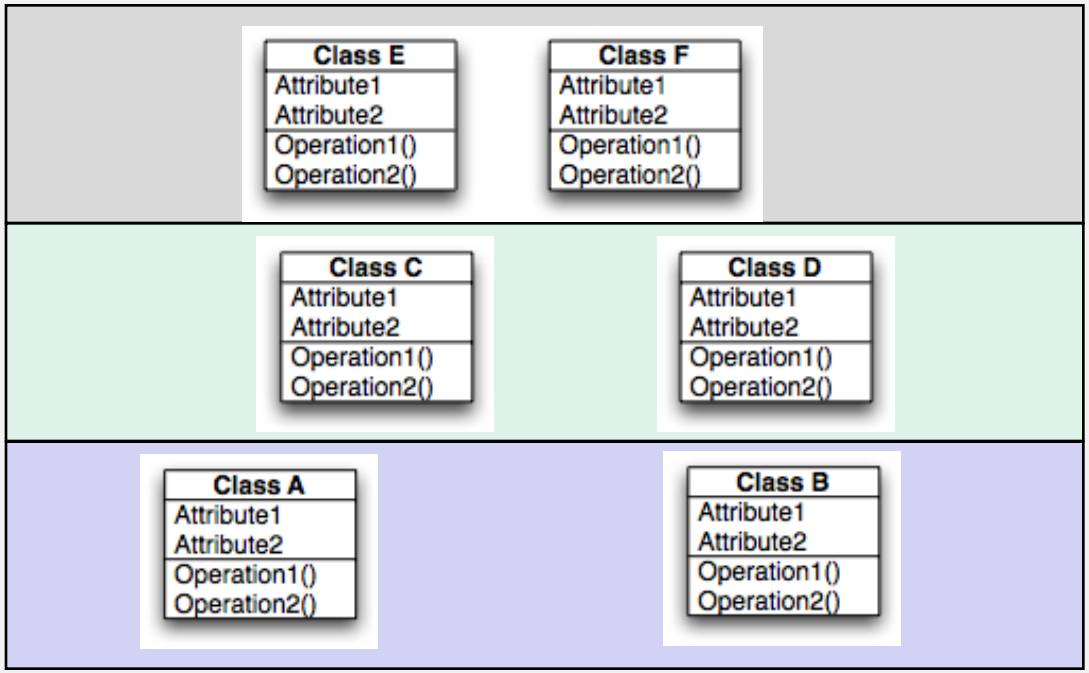
Definition: 3-Tier Architecture

A software architecture where the 3 layers are allocated on 3 separate hardware nodes

Note: Layer is a type (e.g. class, subsystem) and Tier is an instance (e.g. object, hardware node)
Layer and Tier are often used interchangeably.

Virtual Machines in 3-Layer Architectural Style

A 3-Layer Architectural Style is a hierarchy of 3 virtual machines usually called presentation, application and data layer



Presentation Layer (Client Layer)

Application Layer (Middleware, Business Logic)

Data Layer

Operating System, Libraries

Example of a 3-Layer Architectural Style

Three-Layer architectural style are often used for the development of Websites:

1. The Web Browser implements the user interface
2. The Web Server serves requests from the web browser
3. The Database manages and provides access to the persistent data.

Example of a 4-Layer Architectural Style

4-Layer-architectural styles (4-Tier Architectures) are usually used for the development of electronic commerce sites. The layers are

1. The Web Browser, providing the user interface
2. A Web Server, serving static HTML requests
3. An Application Server, providing session management (for example the contents of an electronic shopping cart) and processing of dynamic HTML requests
4. A back end Database, that manages and provides access to the persistent data
   - In current 4-tier architectures, this is usually a relational Database management system (RDBMS).

MVC vs. 3-Tier Architectural Style

The MVC architectural style is nonhierarchical (triangular):

- View subsystem sends updates to the Controller subsystem
- Controller subsystem updates the Model subsystem
- View subsystem is updated directly from the Model subsystem

The 3-tier architectural style is hierarchical (linear):

- The presentation layer never communicates directly with the data layer (opaque architecture)
- All communication must pass through the middleware layer

Pipes and Filters

A pipeline consists of a chain of processing elements (processes, threads, etc.), arranged so that the output of one element is the input to the next element

Usually some amount of buffering is provided between consecutive elements

The information that flows in these pipelines is often a stream of records, bytes or bits.

Pipes and Filters Architectural Style

An architectural style that consists of two subsystems called pipes and filters

Filter: A subsystem that does a processing step

Pipe: A Pipe is a connection between two processing steps

Each filter has an input pipe and an output pipe.

The data from the input pipe are processed by the filter and then moved to the output pipe

Example of a Pipes-and-Filters architecture: Unix

Unix shell command: ls -a l cat

Summary

System Design

An activity that reduces the gap between the problem and an existing (virtual) machine

Design Goals Definition

Describes the important system qualities

Defines the values against which options are evaluated

Subsystem Decomposition

Decomposes the overall system into manageable parts by using the principles of cohesion and coherence

Architectural Style

A pattern of a typical subsystem decomposition

Software architecture

An instance of an architectural style

Client Server, Peer-to-Peer, Model-View-Controller.

Thank You