Architectural Styles

Software Engineering I
Lecture 08

Bernd Bruegge, Ph.D.
Applied Software Engineering
Technische Universitaet Muenchen

Architectural Style & Software Architecture

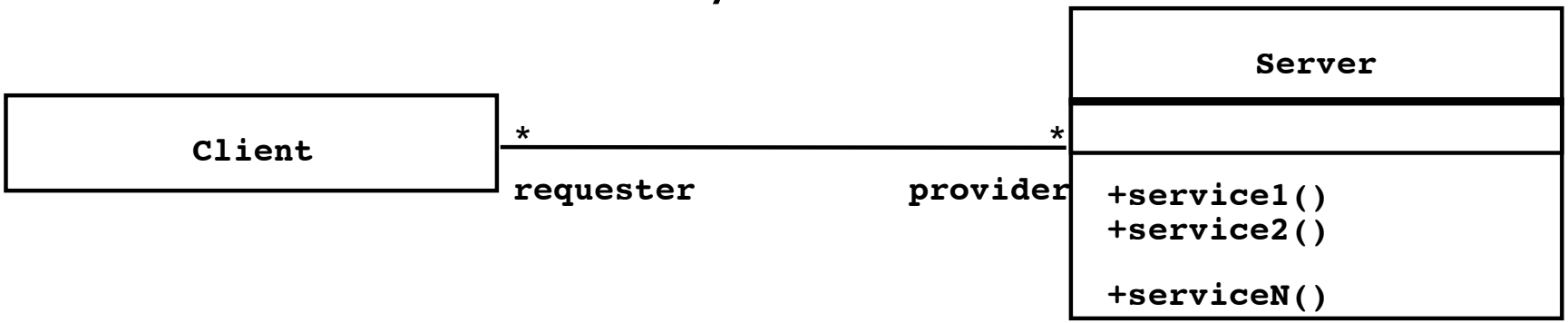
- Subsystem decomposition: Identification of subsystems, services, and their relationship to each other.
- Architectural Style: A pattern for subsystem decomposition
- Software Architecture: Instance of an architectural style

There are many architectural styles

- Client/Server
- Peer-To-Peer
- Repository
- Model/View/Controller
- Three-tier, Four-tier
- 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

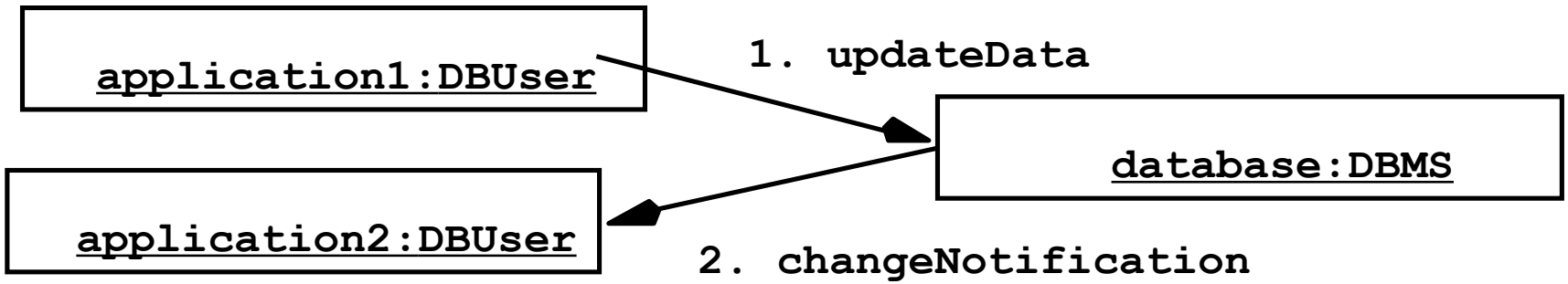
Design Goals for Client/Server Architectures

- 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

A measure of success with which the observed behavior of a system confirms to the specification of its behavior (Chapter 11: Testing)

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

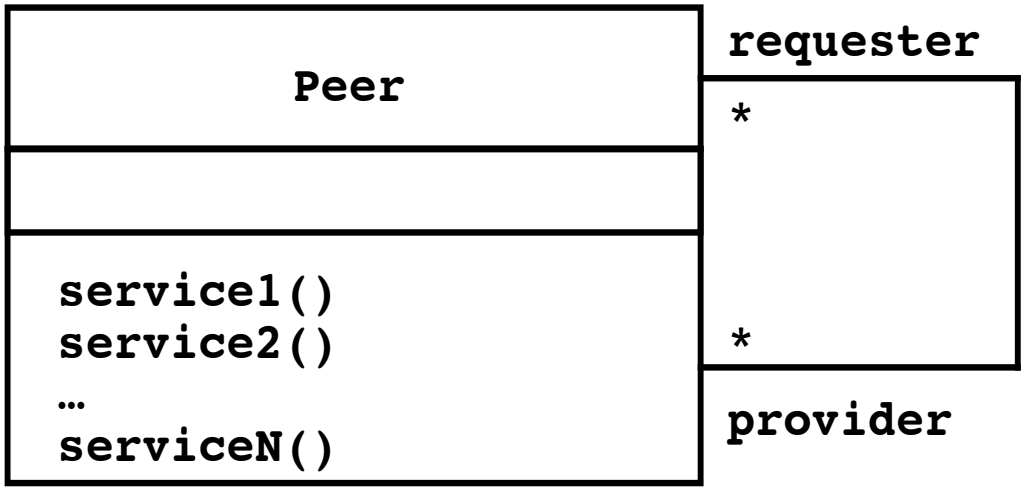


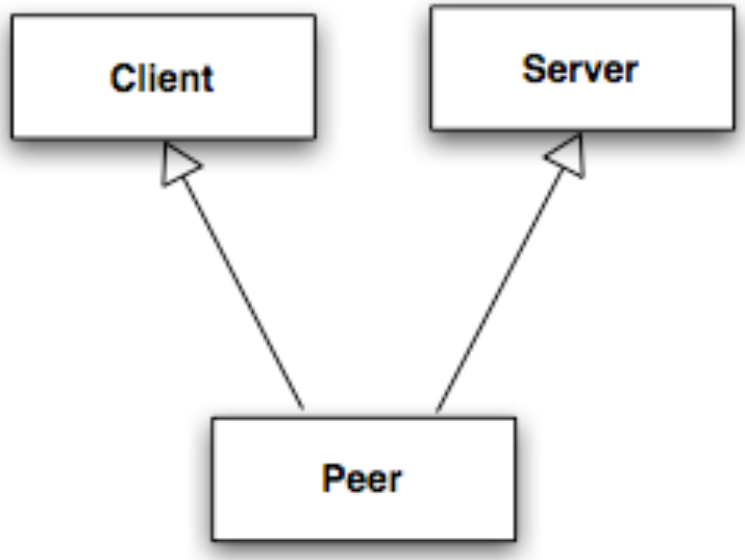
Peer-to-Peer Architectural Style

Generalization of Client/Server Architecture

Clients can be servers and servers can be clients

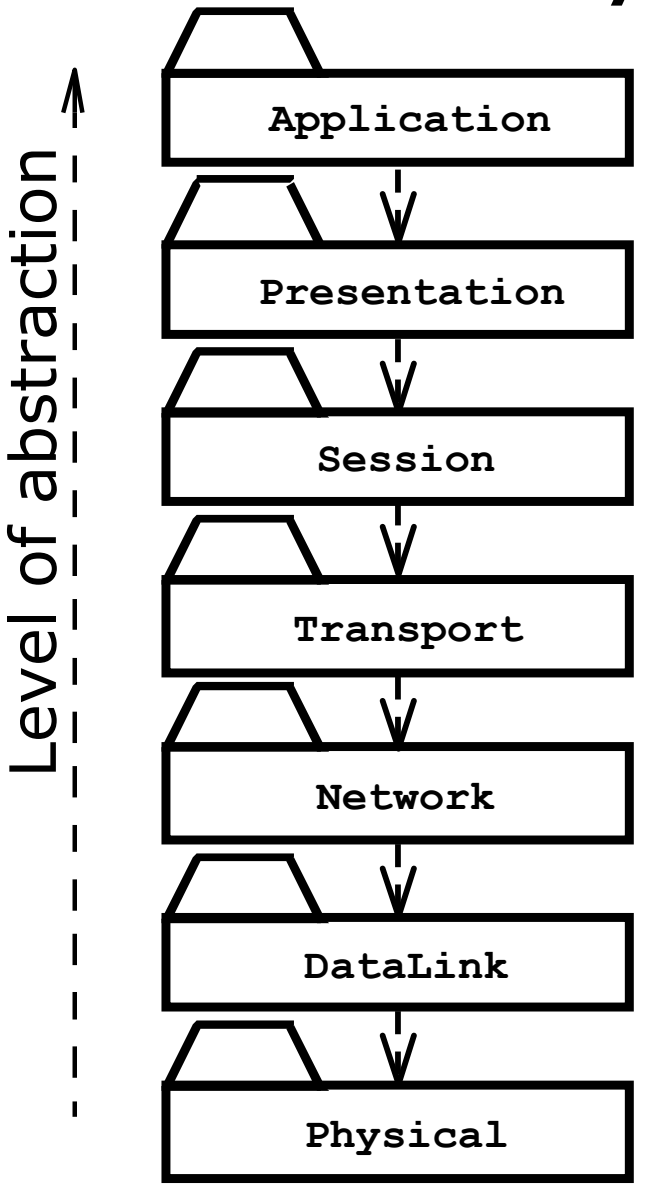
=> "A peer can be a client as well as a server".





Example: Peer-to-Peer Architectural Style

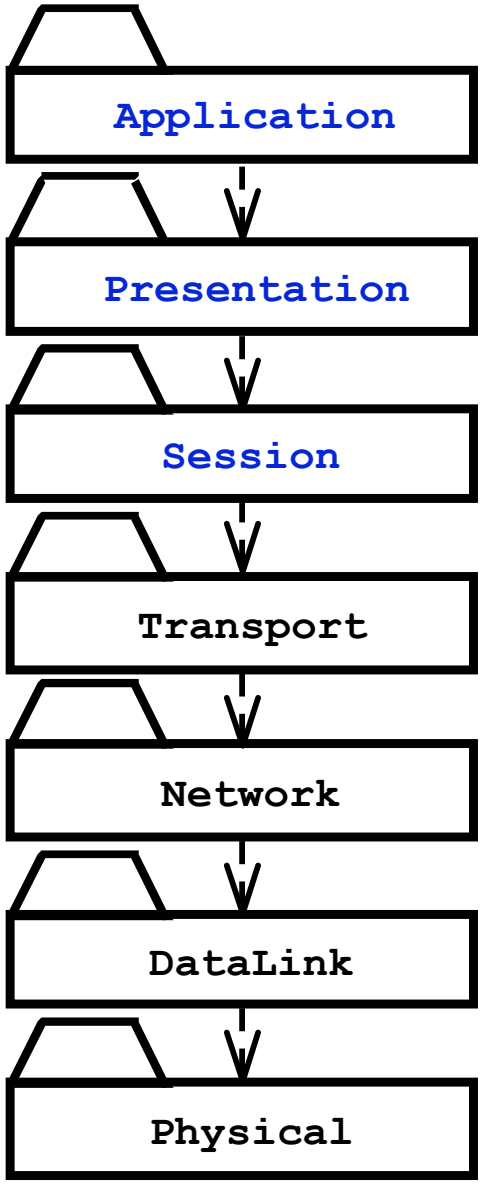
- ISO's OSI Reference Model
  - ISO = International Standard Organization
  - OSI = Open System Interconnection
- Reference model which defines 7 layers and communication protocols between the layers



OSI Model Layers and their Services

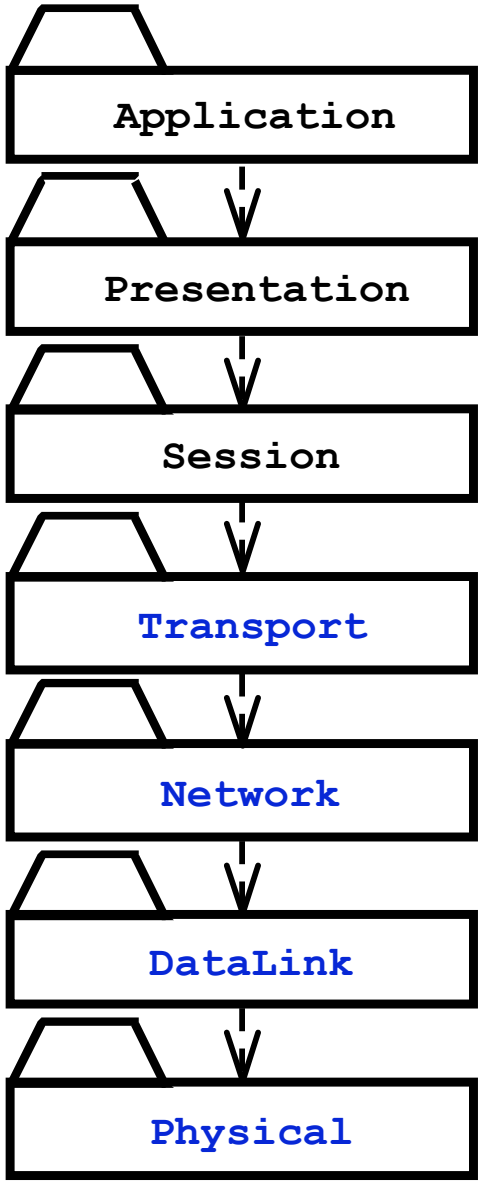
- The Application layer is the system you are building (unless you build a protocol stack)

  - The application layer is usually layered itself
- The Presentation layer performs data transformation services, such as byte swapping and encryption
- The Session layer is responsible for initializing a connection, including authentication

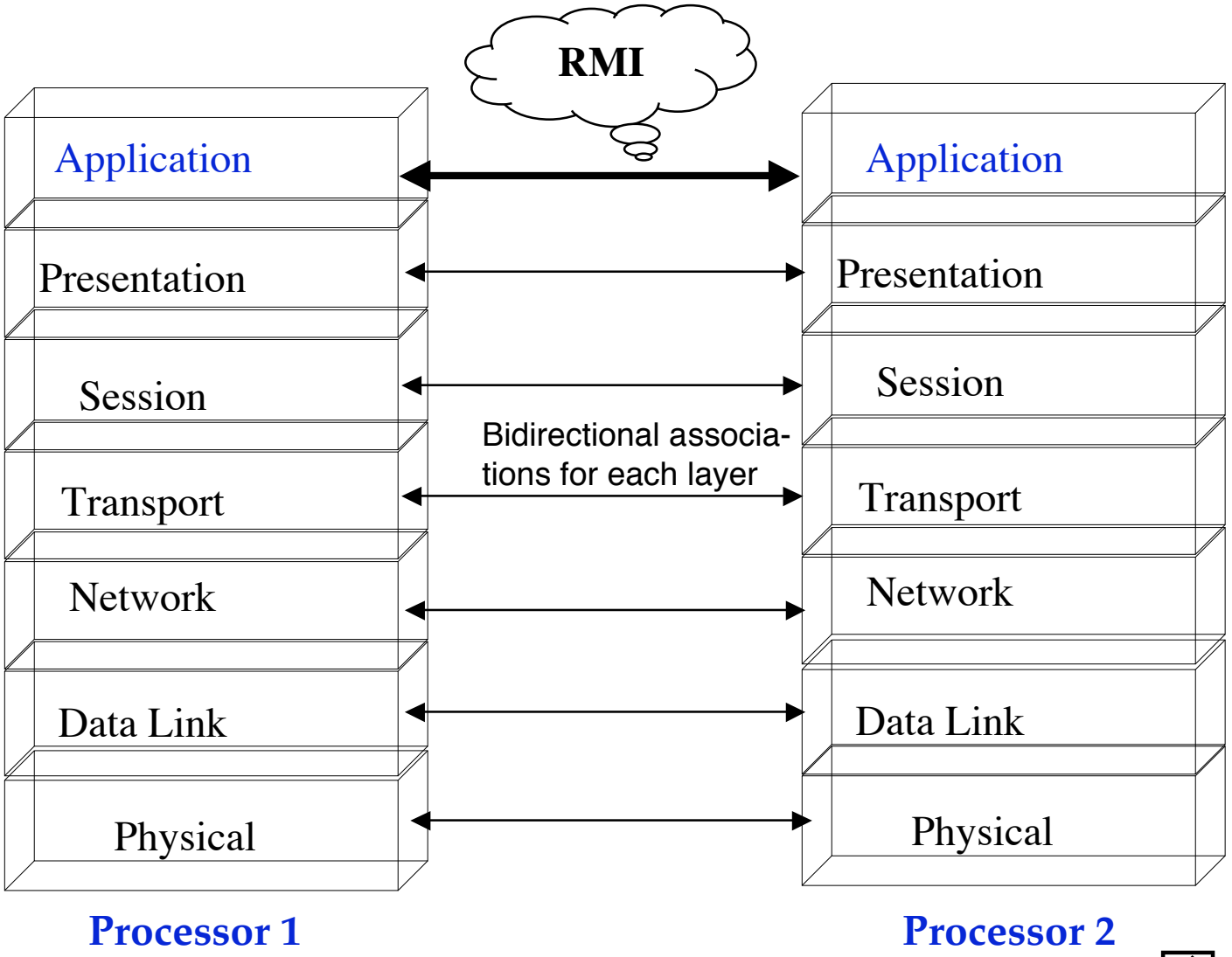


OSI Model Layers and their Services

- The Transport layer is responsible for reliably transmitting messages
  - Used by Unix programmers who transmit messages over TCP/IP sockets
- The Network layer ensures transmission and routing
  - Services: Transmit and route data within the network
- The Datalink layer models frames
  - Services: Transmit frames without error
- The Physical layer represents the hardware interface to the network
  - Services: sendBit() and receiveBit()

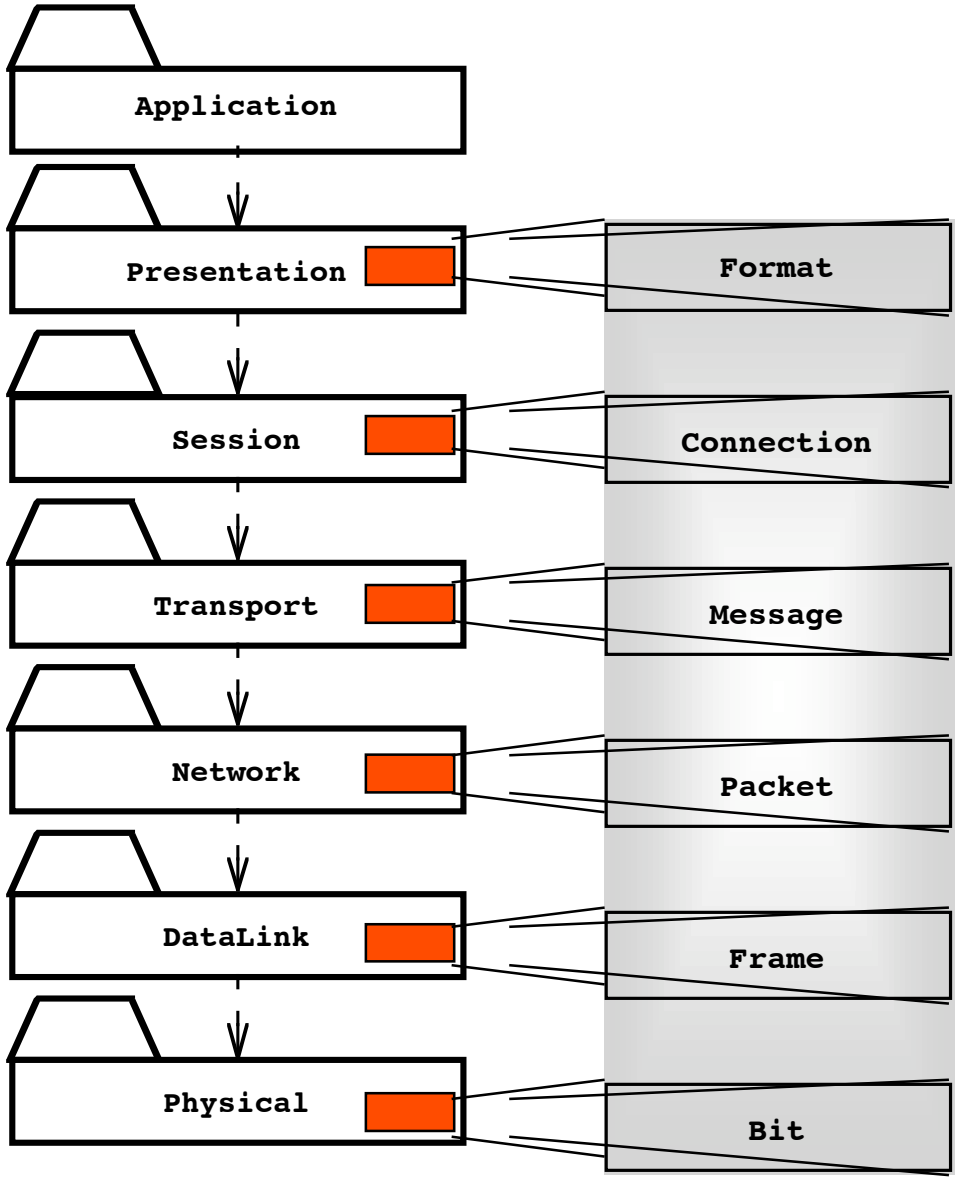


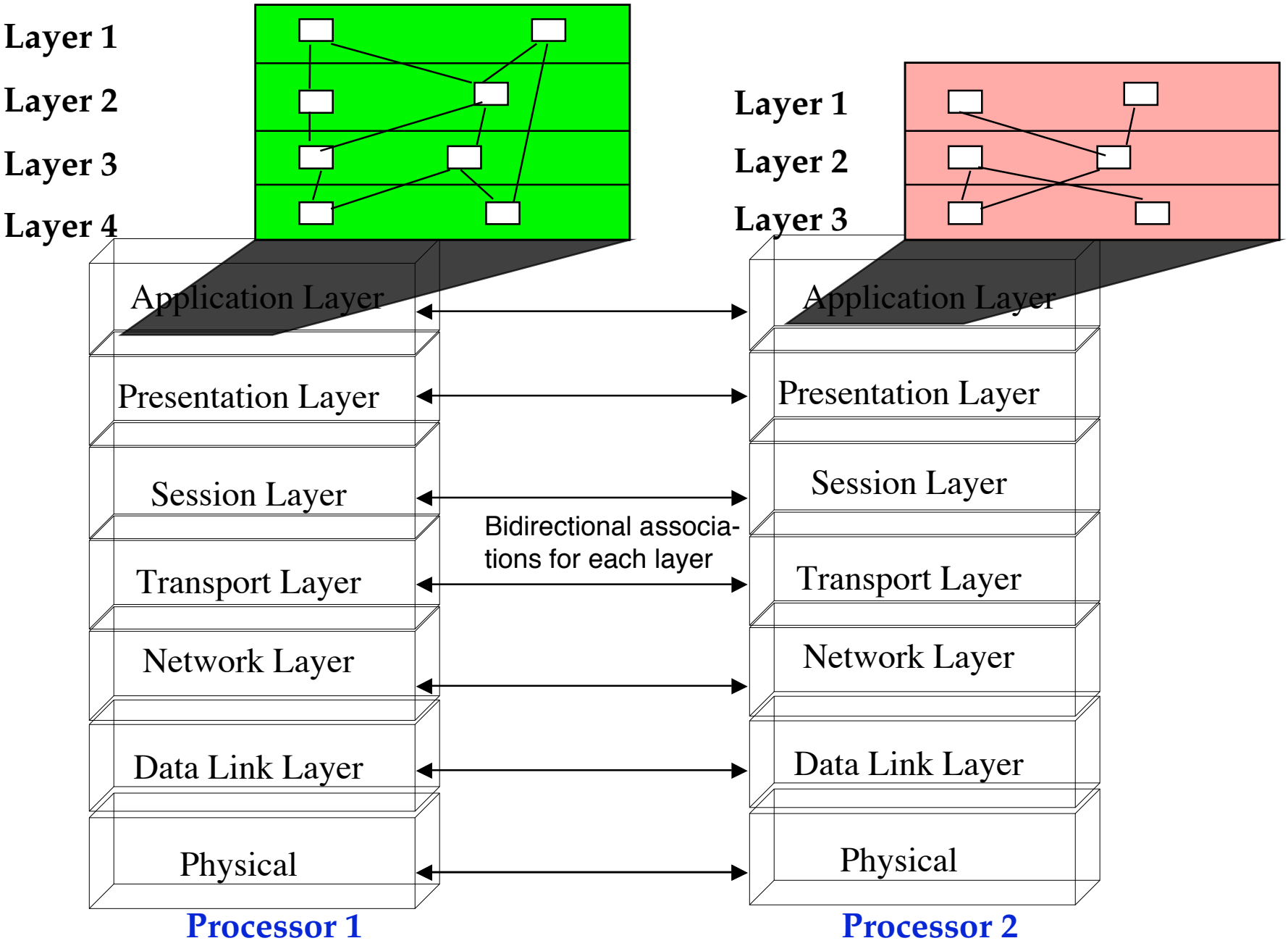
The Application Layer Provides the Abstractions of the "New System"



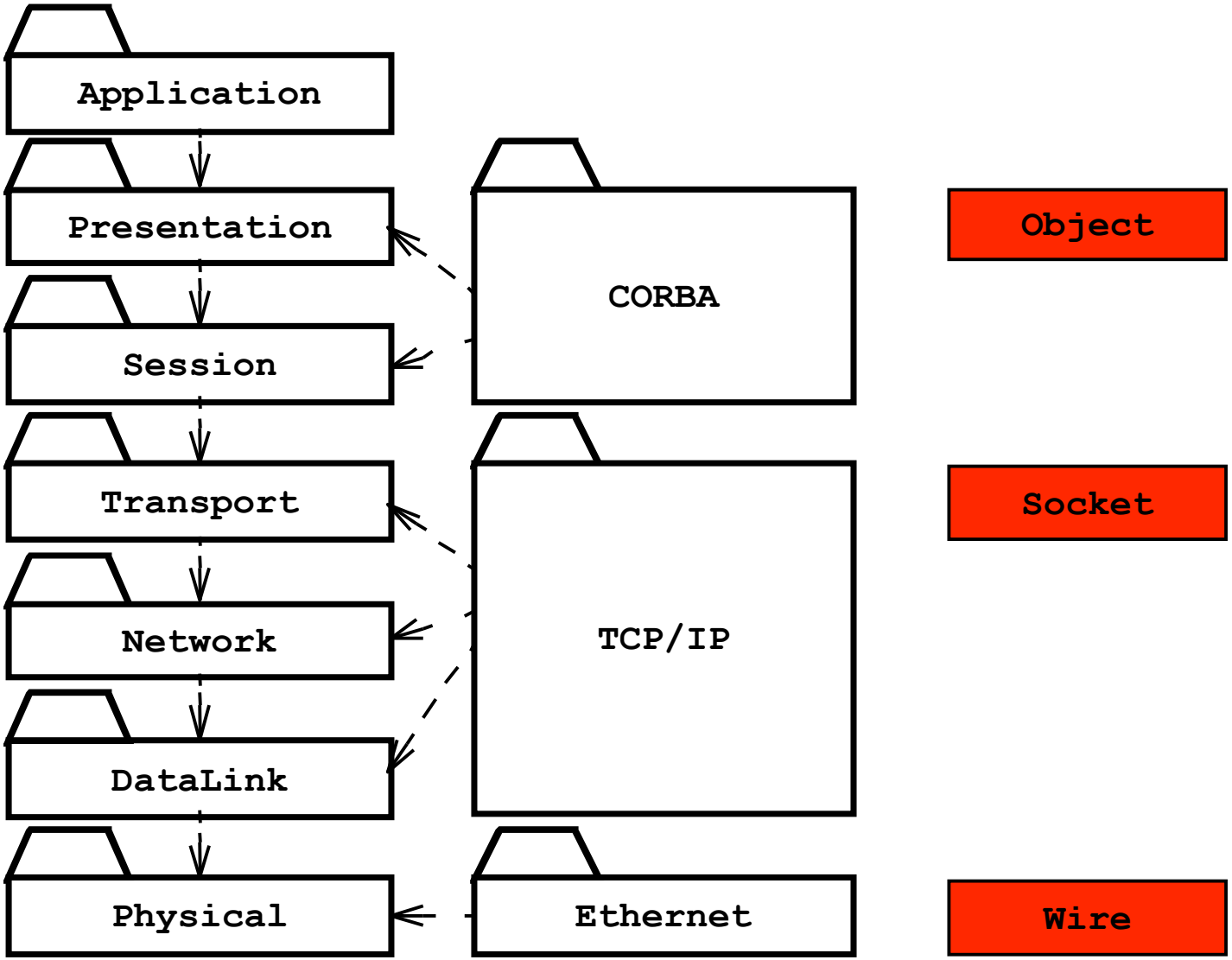
An Object-Oriented View of the OSI Model

- The OSI Model is a closed software architecture (i.e., it uses opaque layering)
- Each layer can be modeled as a UML package containing a set of classes available for the layer above



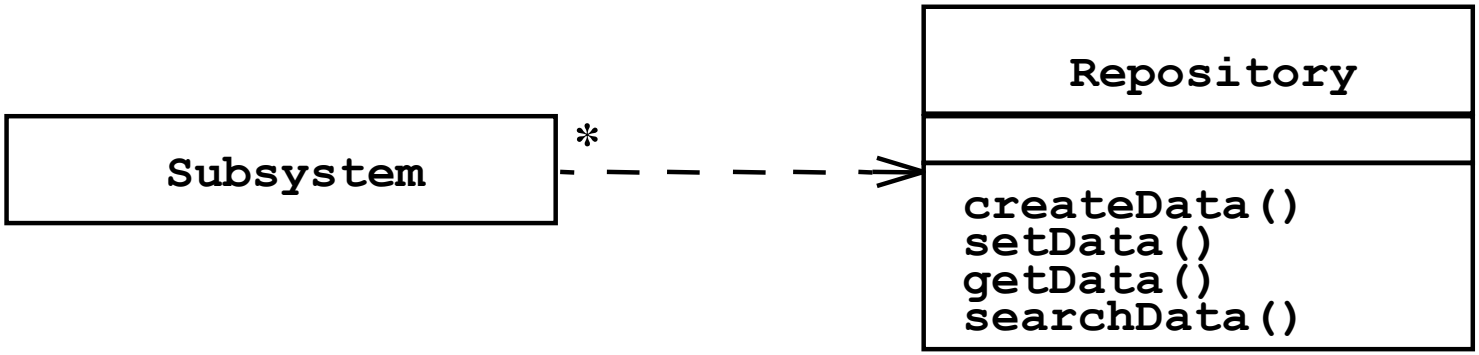

Layer 1
Layer 2
Layer 3
Layer 4
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical
Processor 1
Bidirectional associa-
tions for each layer
Layer 1
Layer 2
Layer 3
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical
Processor 2

Middleware Allows Focus On Higher Layers

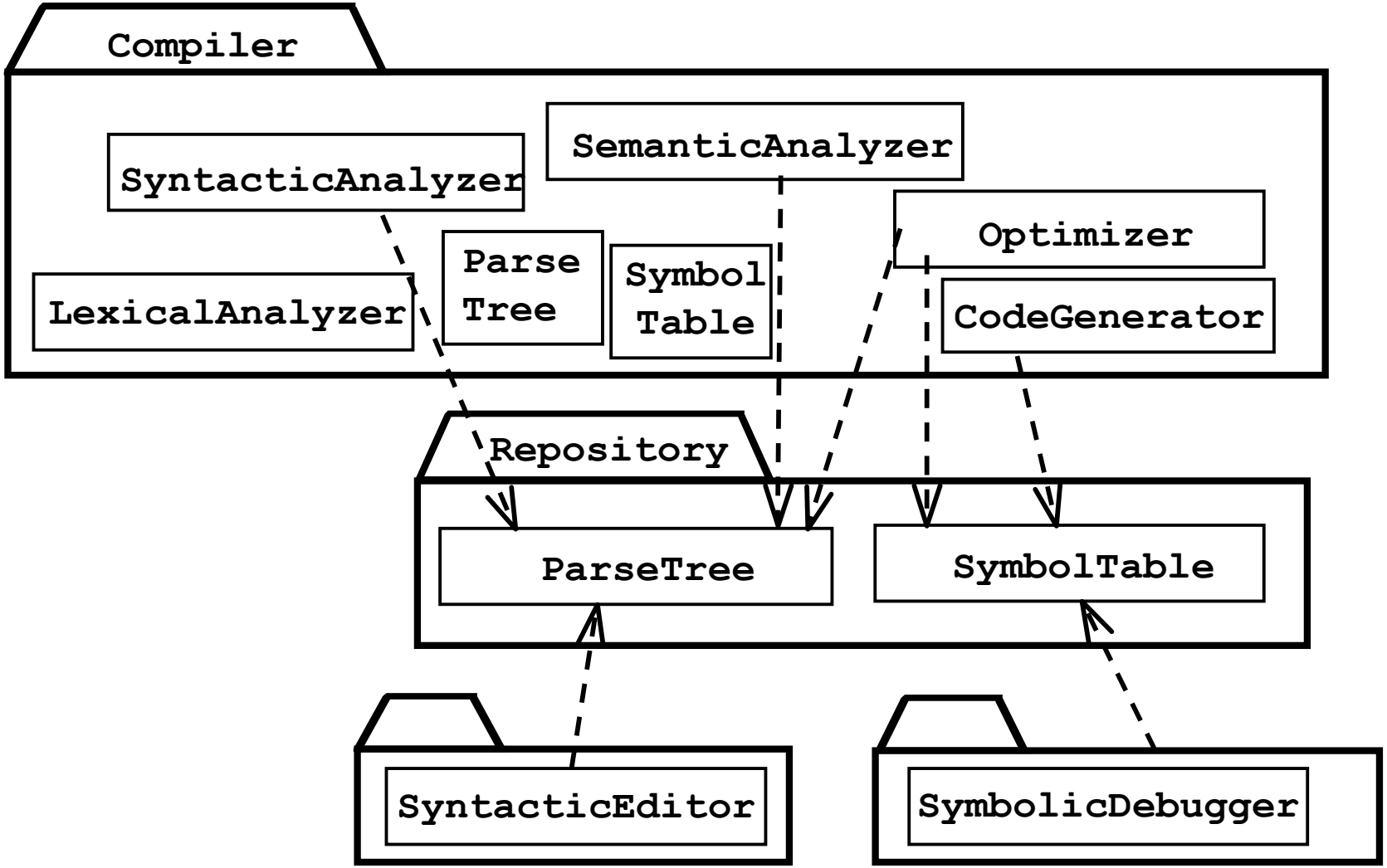


Repository Architectural Style

- Subsystems access and modify data from a single data structure called the repository
- Also called blackboard architecture
- 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)



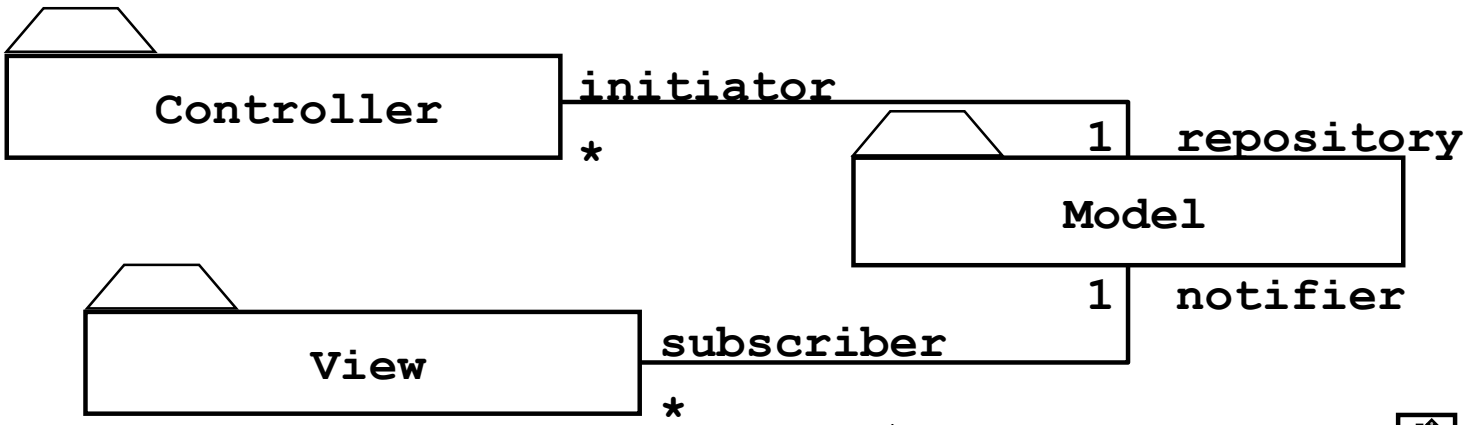
Model /View/ Controller Architectural Style

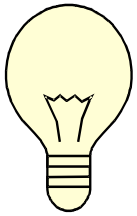
- Subsystems are classified into 3 different types

  Model subsystem: Responsible for application domain knowledge

  View subsystem: Responsible for displaying application domain objects to the user

  Controller subsystem: Responsible for sequence of interactions with the user and notifying views of changes in the model

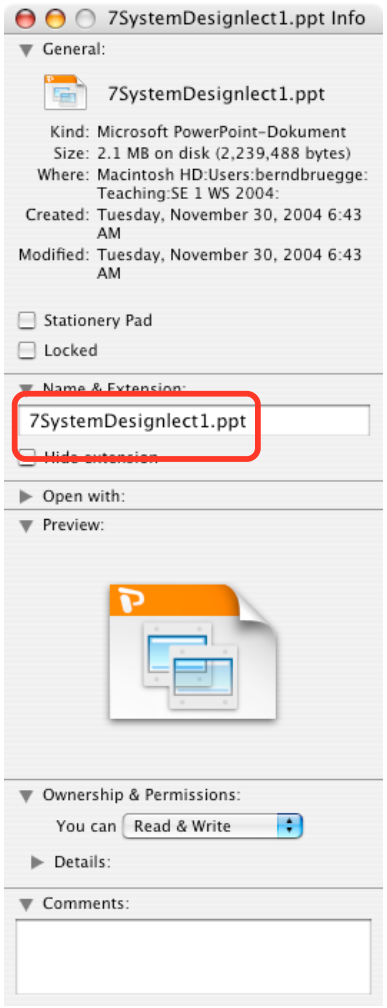


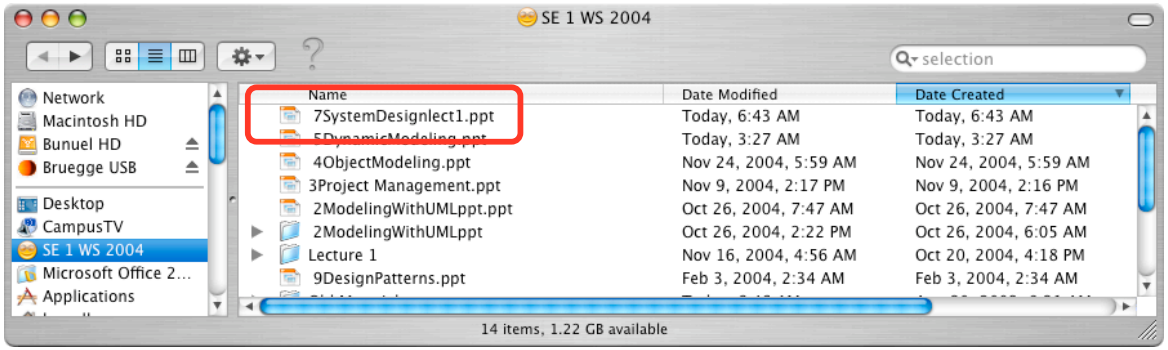


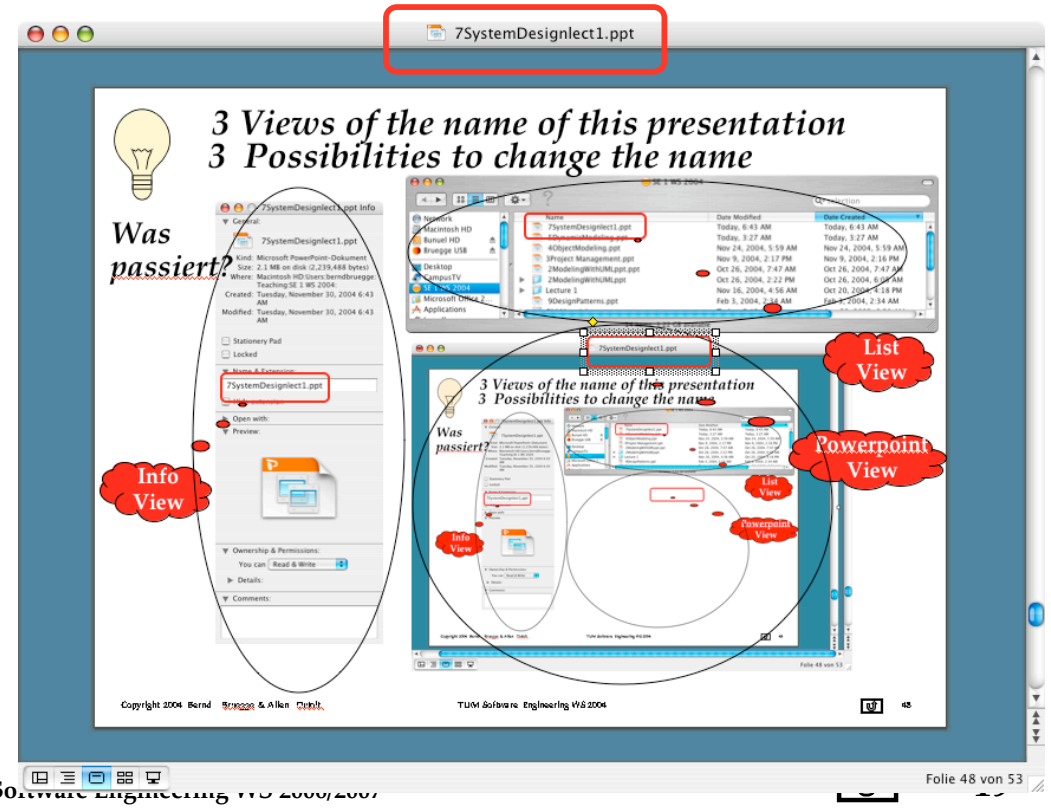
3 Views of the name of this presentation

3 Possibilities to change the name

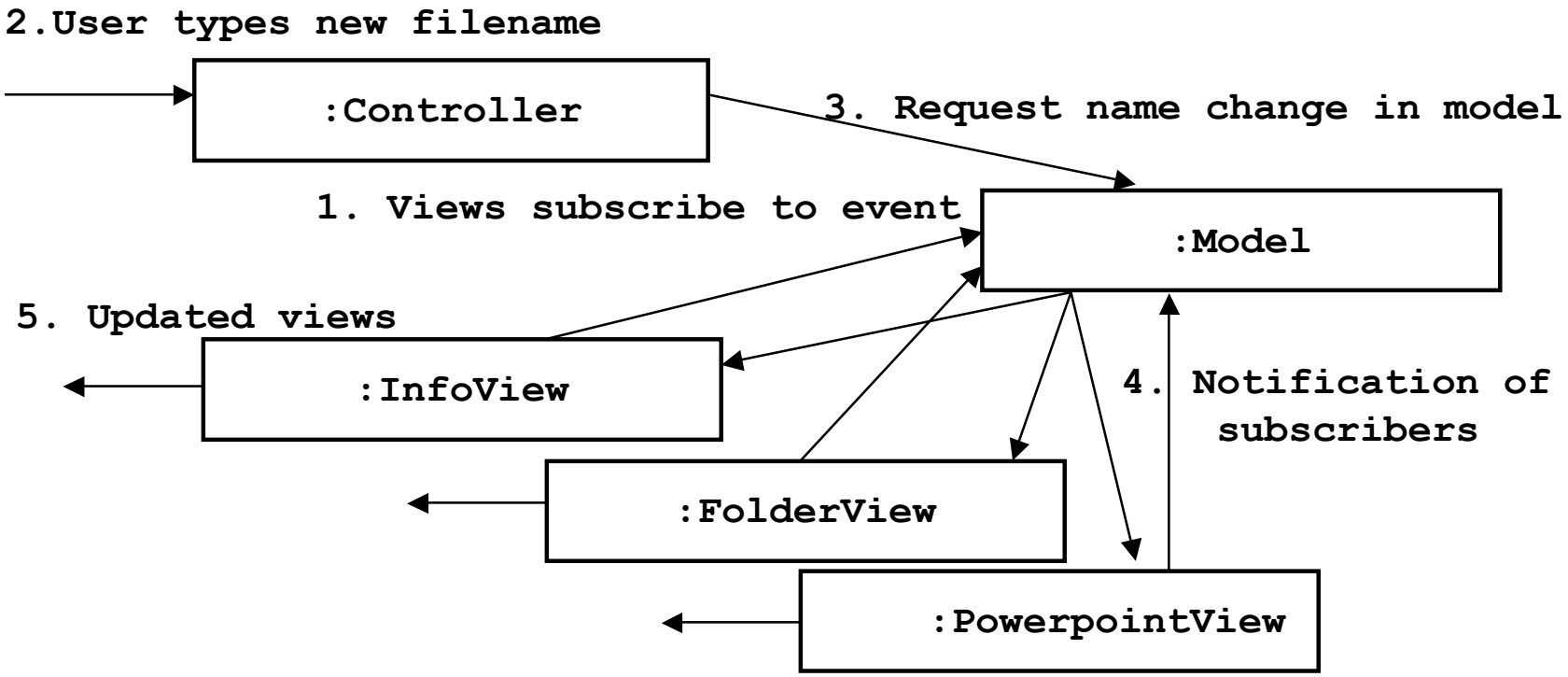
What happens?



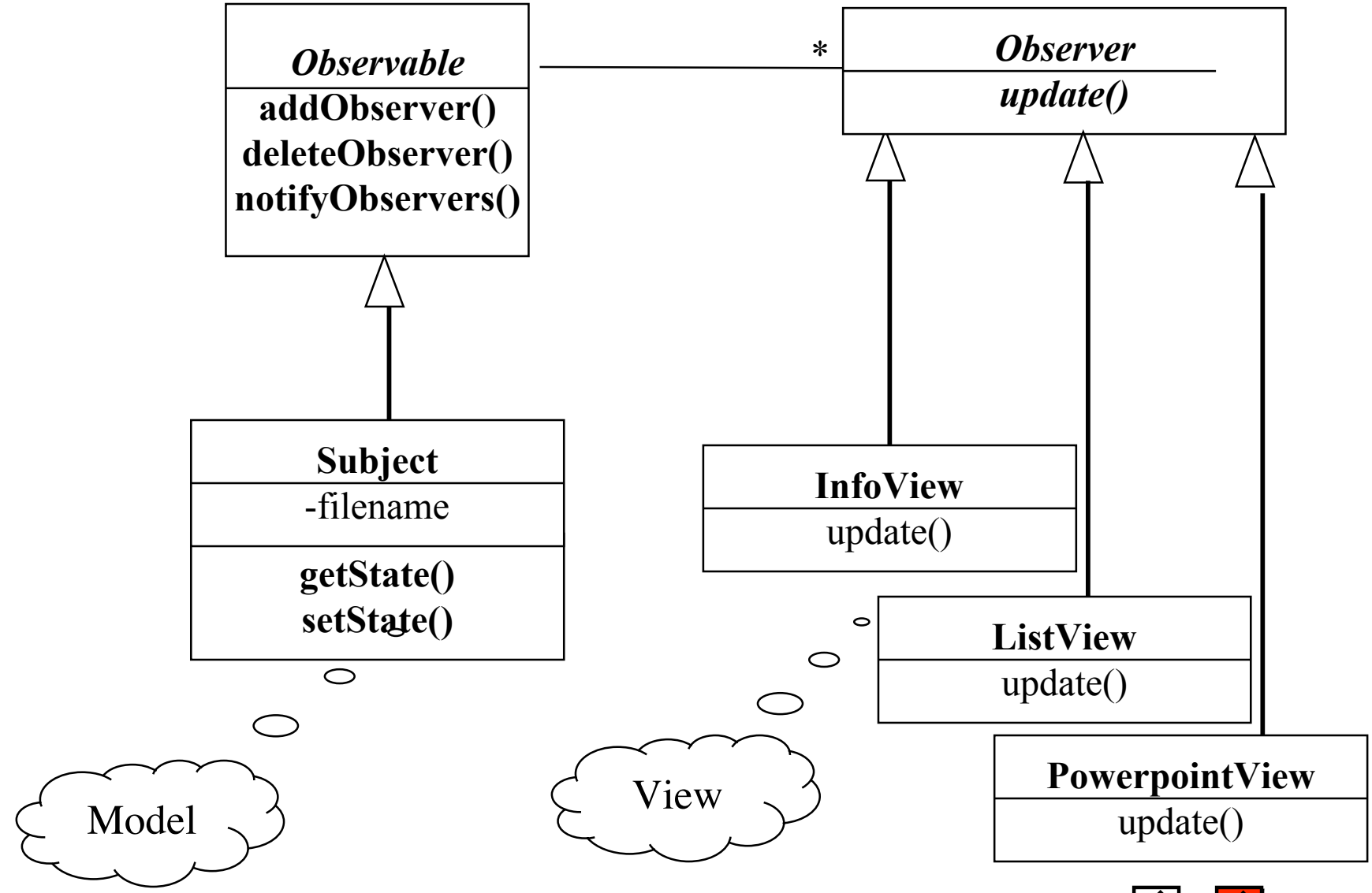




Sequence of Events (UML Collaboration Diagram)



Class Diagram („Observer Pattern“)



Additional Readings

- L.D. Erman, F. Hayes-Roth,
  - "The Hearsay-II-Speech- Understanding System: Integrating knowledge to resolve uncertainty", ACM Computing Surveys, Vol 12. No. 2, pp 213-253, 1980
- J.D. Day and H. Zimmermann,
  - "The OSI Reference Model", Proc. Of the IEEE, Vol. 71, pp 1334-1340, Dec 19983

Summary

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

Architectural Styles

Software Engineering I
Week 4, Lecture 3

Prepared by
Bernd Bruegge, Ph.D.
University Professor of Applied Software Engineering
Technische Universitaet Muenchen