Software Architecture

Thursday, November 15

Design Patterns

A general, reusable solution to a commonly occurring problem in a given context

Often have best practices associated with them

Architectural Patterns

Architectural Patterns are the fundamental **structural organization** for software systems.

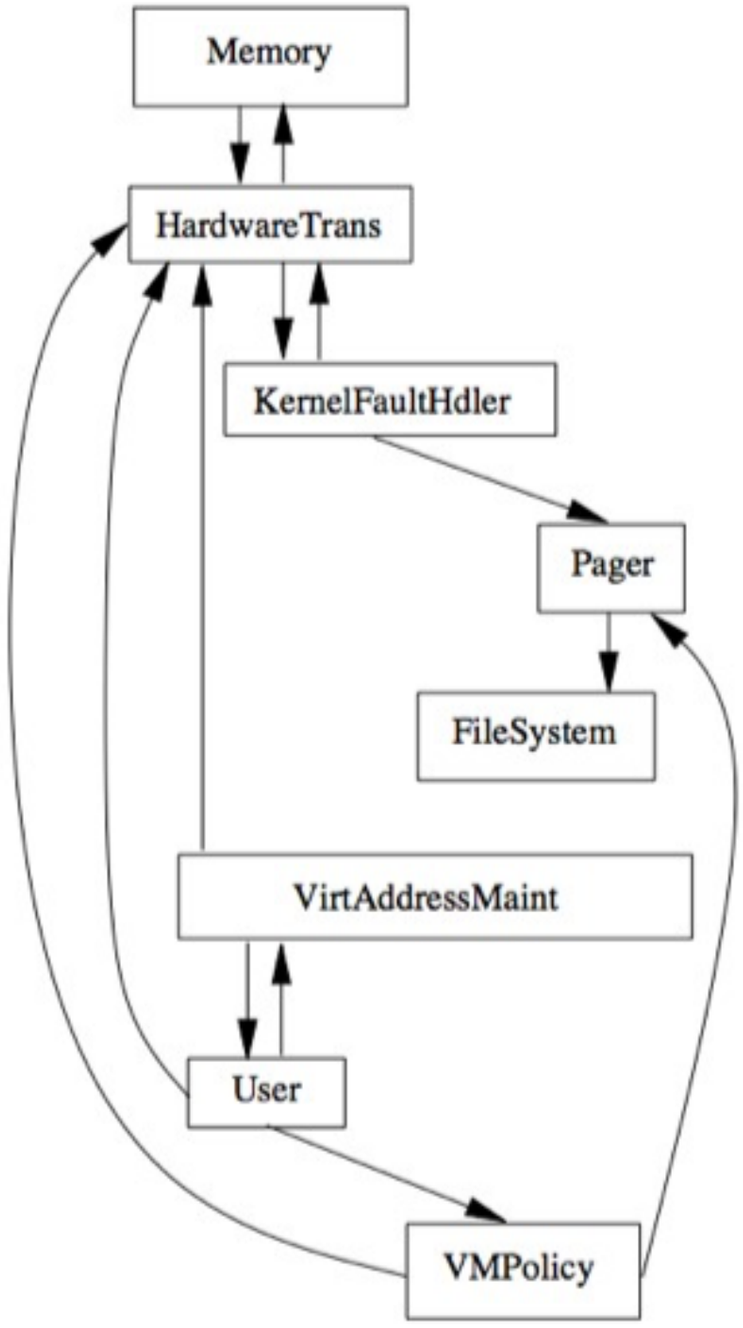
What is Software Architecture?

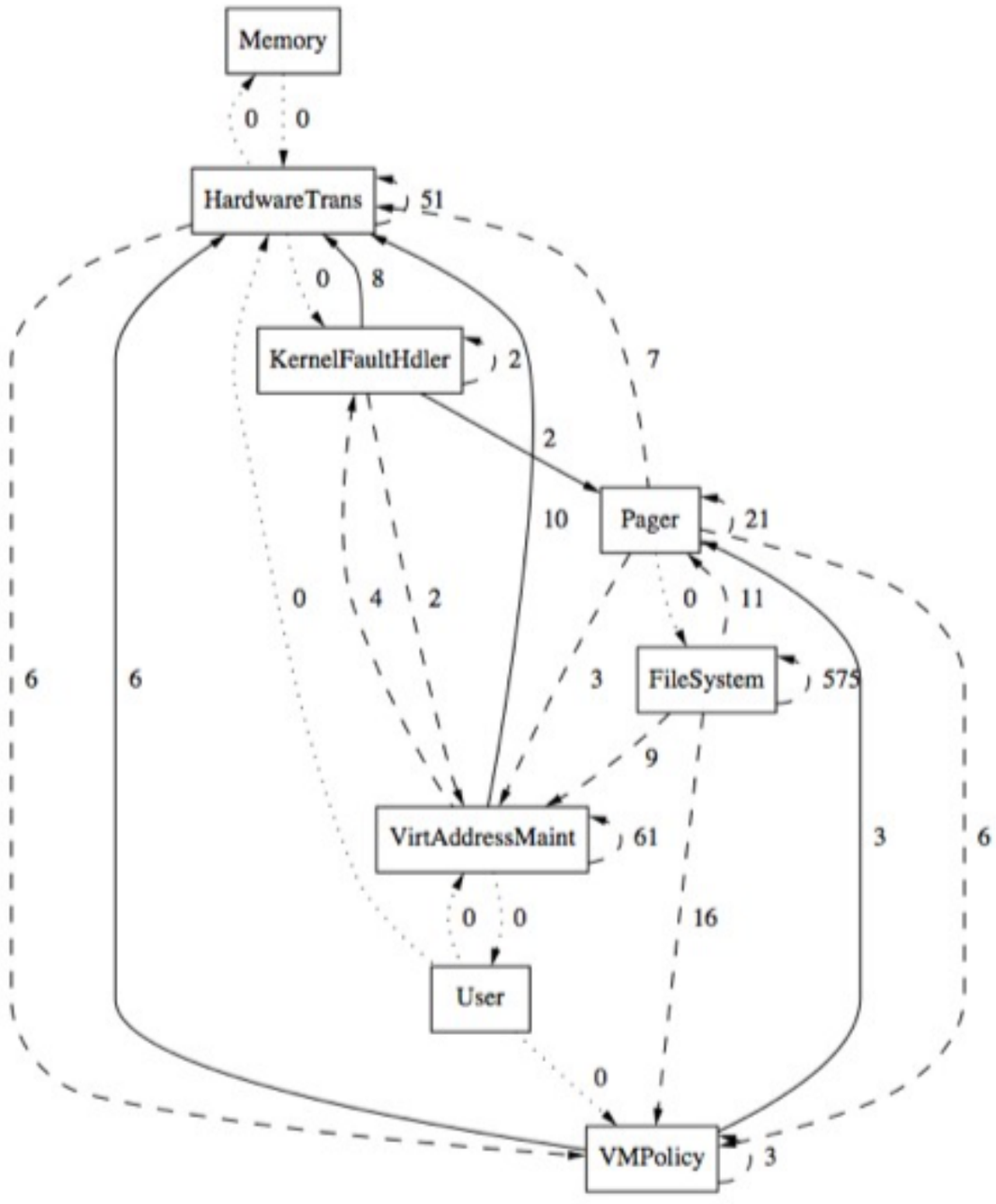
Architecture: shows pieces of a system & their relationships

Component: self-contained piece of a system, with clearly-defined interfaces and structure

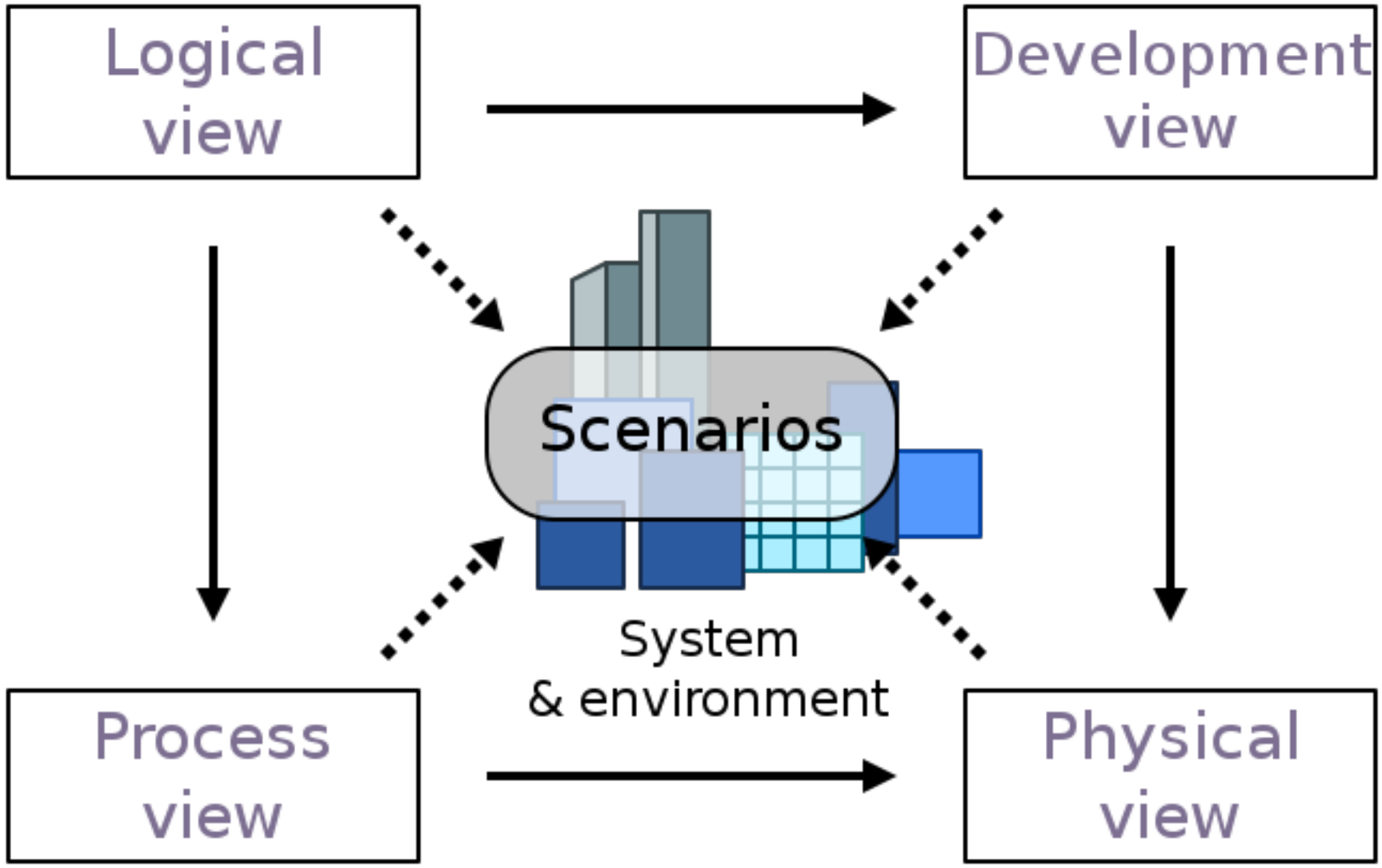
Connector: a linkage between components via an interface

Ideal vs. Real Architecture

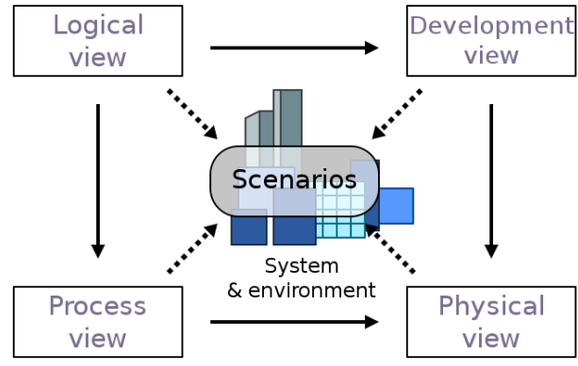




4+1 Architectural View



4+1 Architectural View



Logical View: concerned with the functionality the system provides to end users.

Process View: Focuses on the runtime behavior of the system. This includes the processes, and their communication

Development View: This is the system from the perspective of the developer. It has to deal with the management of the project, maintenance etc;

Physical View: How the system is deployed on actual physical hardware

Architectural Styles

Architectural Styles in Software

*An architectural style defines a **family of systems** in terms of a pattern of structural organization. More specifically, an architectural style defines a **vocabulary** of components and connector types, and a set of constraints on how they can be combined.*

— Shaw and Garlan

Thesis template vs.

Book chapter vs.

Novel vs. …

Architectural Styles

The Monolith

Client and Server

Micro-services

REST

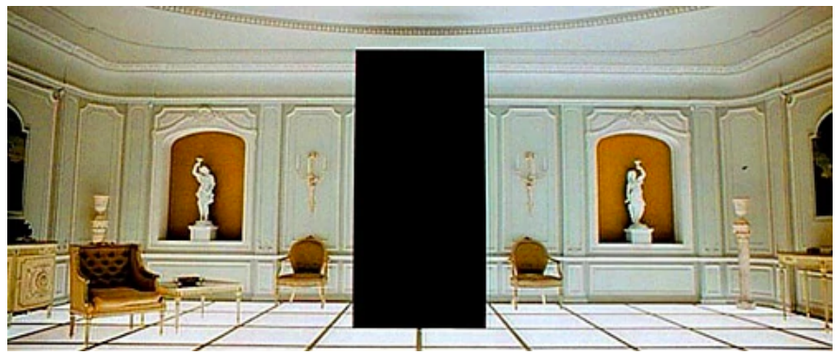
MVC

Layered

Event-driven

Pipe and Filter

The Monolith



A single-tier software application.

All components are combined into a single application.

Many projects begin this way.

Advantages & Disadvantages

Advantages & Disadvantages

Advantages:

Easy to deploy and test

Easy to develop

Disadvantages:

Reliability

Scalability

Maintenance

Client-server Architecture

Client-server Architecture

A client-server architecture distributes functionality of the system into services.

Client-server Architecture

A client-server architecture distributes functionality of the system into services.

Each service potentially delivered from a separate server.

Client-server Architecture

A client-server architecture distributes functionality of the system into services.

Each service potentially delivered from a separate server.

Clients use these service and access the server through the Internet.

Client-server Architecture

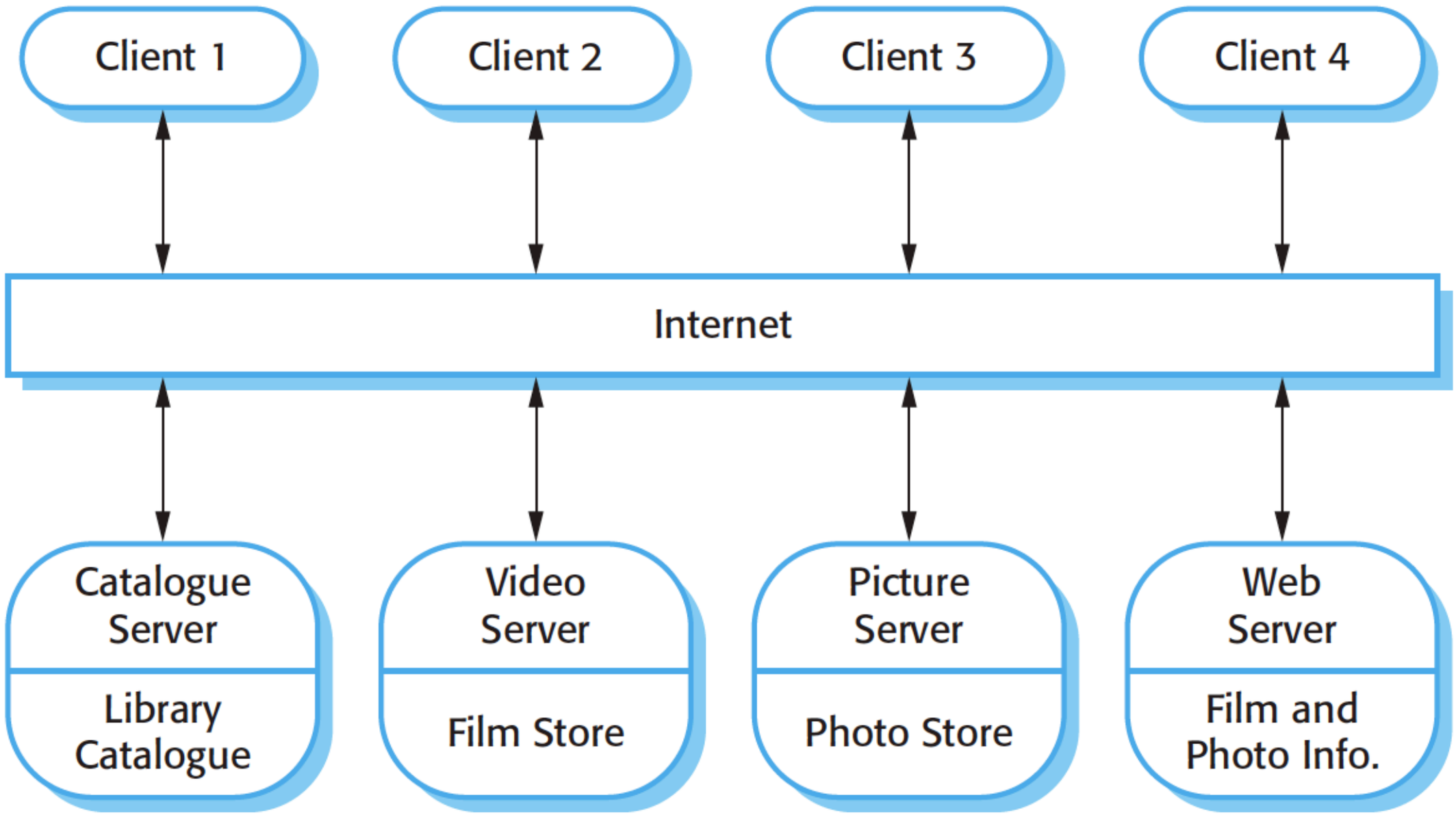
A client-server architecture distributes functionality of the system into services.

Each service potentially delivered from a separate server.

Clients use these service and access the server through the Internet.

Static structure and not the run-time organization.

Film and picture library



Client-server Architecture

Advantages

Servers distributed across a network

Makes effective use of networked systems. May require cheaper hardware

Easy to add new servers or upgrade existing servers

Disadvantages

Each service/server is single point of failure (susceptible to DOS attacks)

Performance depends on the network as well as the system

May require a central registry of names and services — it may be hard to find out what servers and services are available

Data interchange may be inefficient

For Amazon

How would the client server architecture look like?

What components would reside in the client? In the server?

Micro-service architecture

The extreme generalization of Client-Server

Instead of monolithic systems one has many concise services

A micro-service is a *“loosely coupled, reusable software component, which can be distributed.”*

Services use message based communication

Service discovery becomes a challenge

Advantages & Disadvantages

Advantages:

Good reliability

High scalability

Disadvantages:

Difficult testing

Deployment is harder

More difficult maintenance (across service boundaries)

RESTful Architecture

Inspired from the architecture of the largest distributed application ever: The Web

Stateless requests

Every resource has an individual URI

Uniform interface for all resources (GET, POST, PUT, DELETE)

The structure of a response is not specified

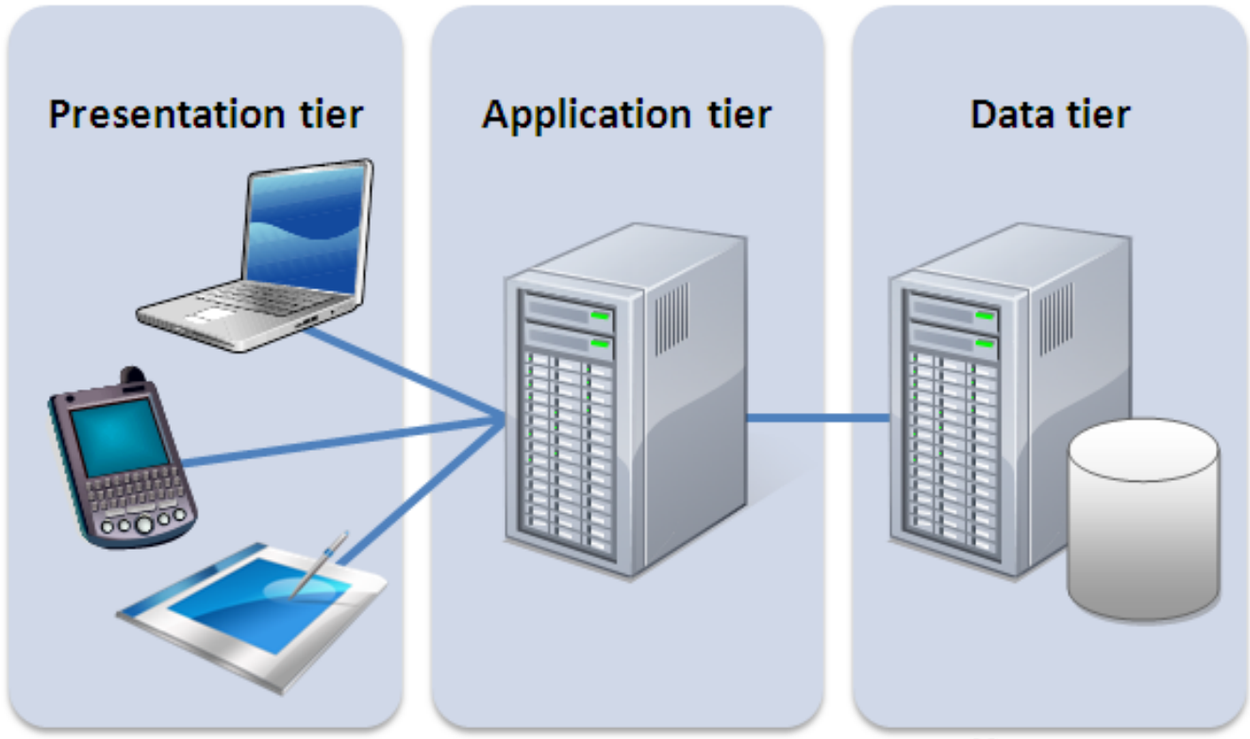
3-tier architecture

A client server architecture, where the functional process logic (business logic), data persistence (data access, storage management) and user interfaces are developed a 3 separate modules.

Presentation Layer: sends content to the browser/ user. You might use HTML/CSS, React, Angular etc.

Application Layer: process the business logic of the application; You might use Java, C#, Python etc.

Data Access Layer: a database management system that provides access to the data: MySQL, Postgres, MongoDB, etc.



Advantages & Disadvantages

Advantages:

Good Scalability

Good Flexibility

High Maintainability

Disadvantages:

More complicated deployment

The business logic is still a monolith.

Layered Architecture

Organizes a system into a subtasks, where each group of subtasks is at a particular layer of abstraction

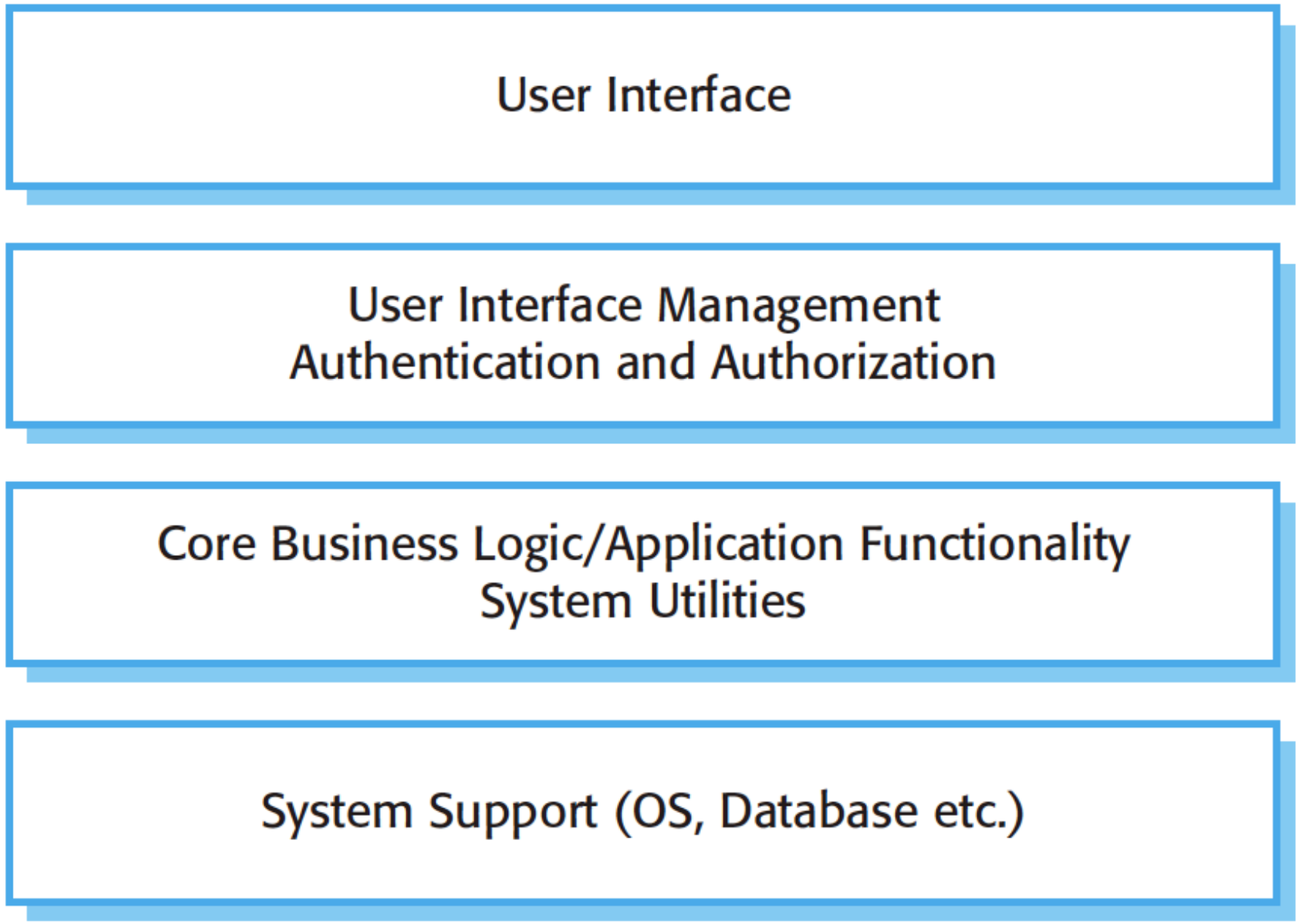
Each layer provides a set of services to the layer "above."

Normally layers are constrained so elements only see

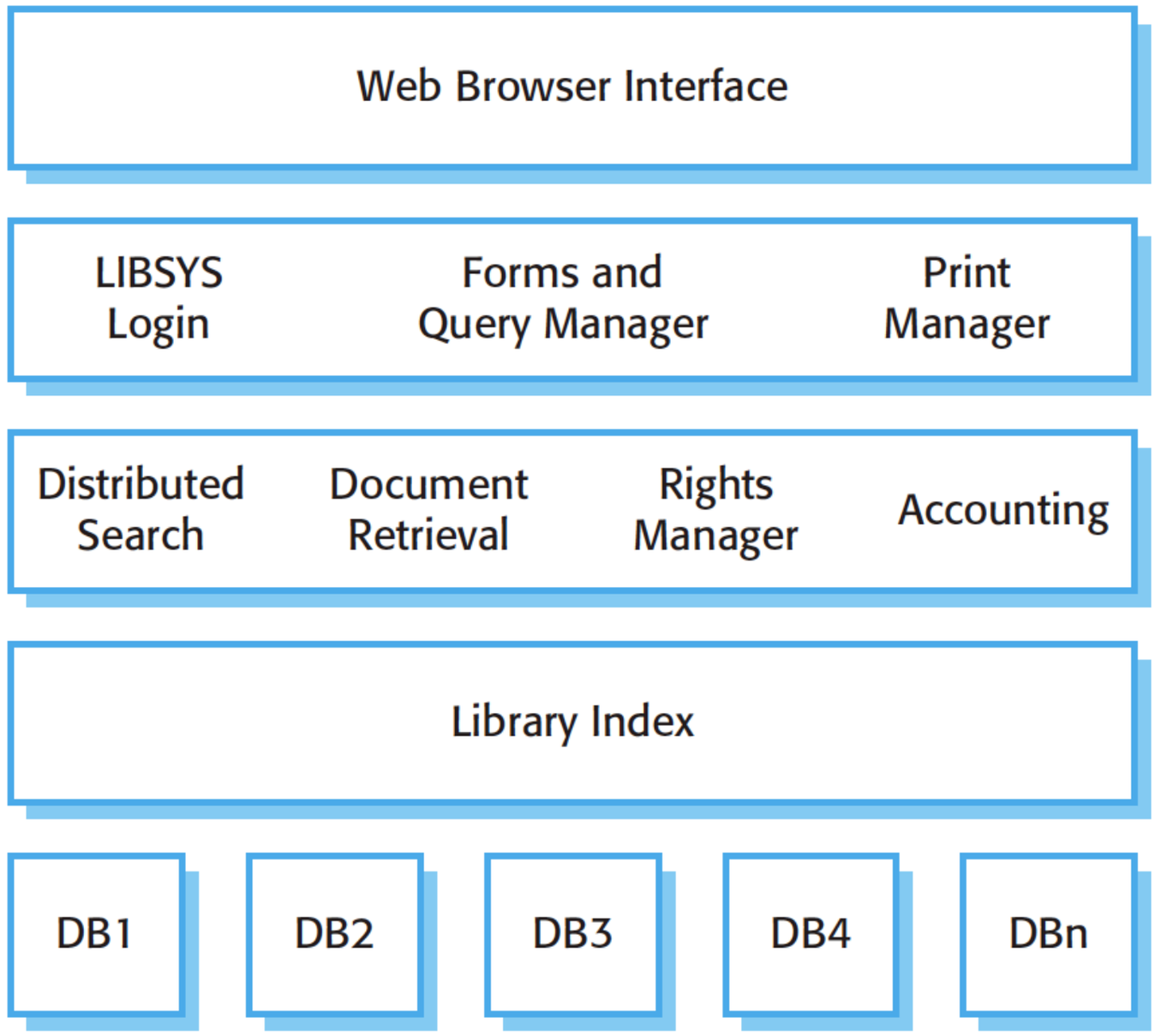
Other elements in the same layer, or

Elements of the layer below

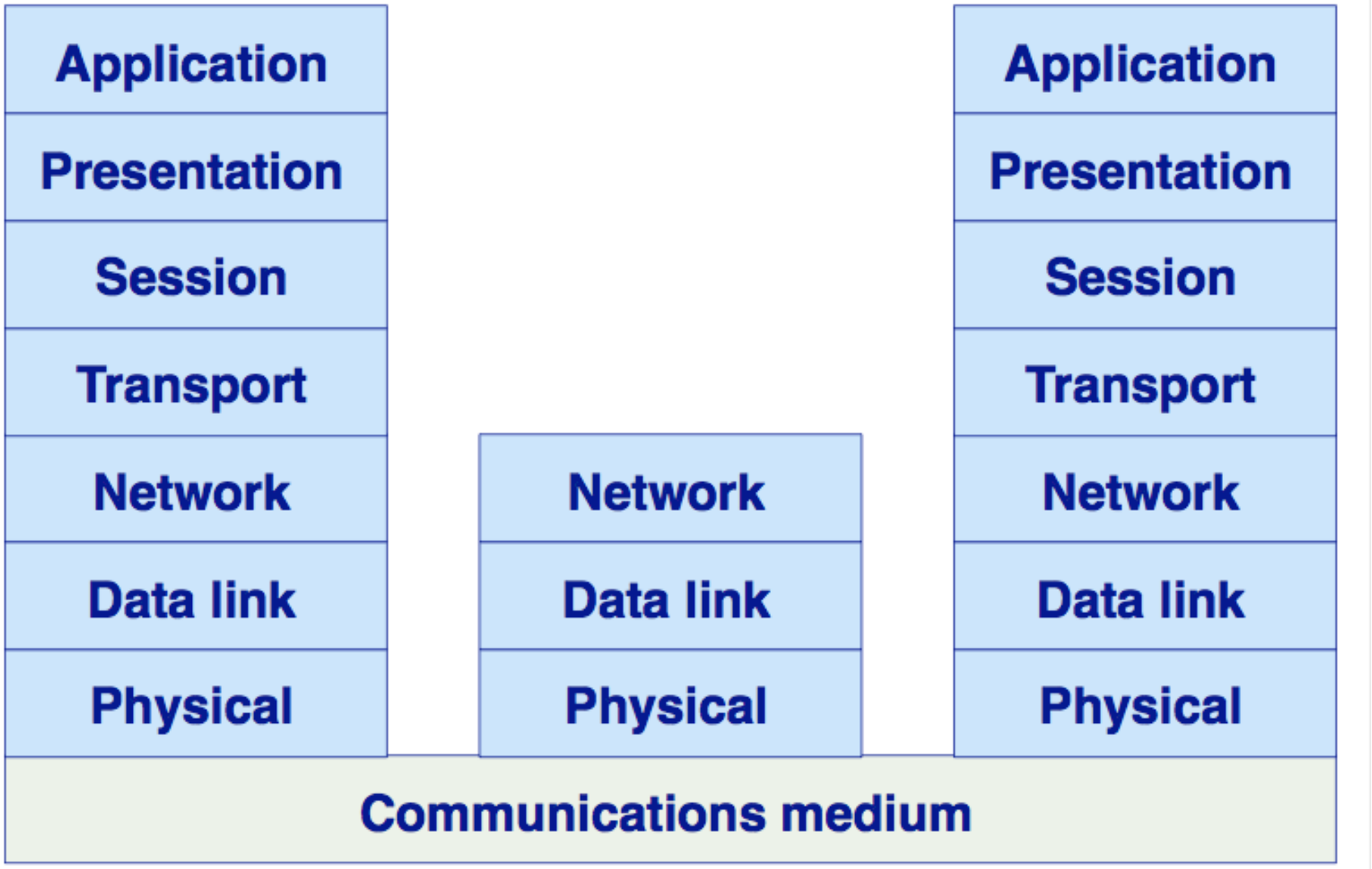
Generic Layered Architecture



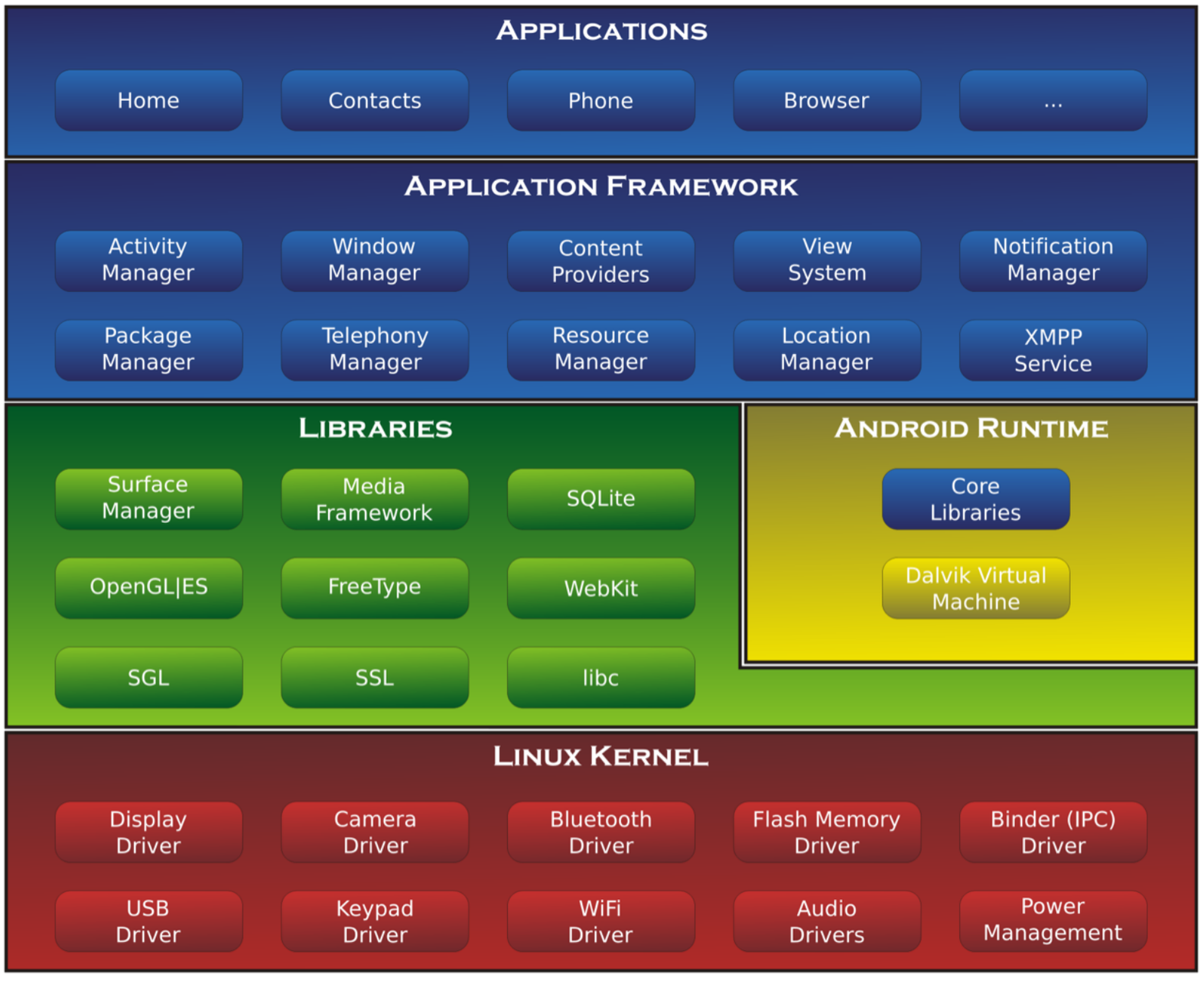
Online Library System



OSI reference model



Android Architecture



Advantages

Layers can be developed independently (and incrementally)

Makes reuse easier

Makes individual layers interchangeable (as long as interface is same)

Layer interactions clearly defined (through interfaces)

When layer interface change, only adjacent layers affected

Disadvantages

It’s hard to achieve a clean separation

Layers sometimes introduce unnecessary work

Performance can be impacted since each layer needs to interpret the service request

Event Driven Architecture

In an event-driven architecture components perform services in **reaction** to external events generated by other components

In **broadcast models** an event is broadcast to all sub-systems. Any sub-system which can handle the event may do so.

In **interrupt-driven models** real-time interrupts are detected by an interrupt handler and passed to some other component for processing

Event Driven Architecture

[Sommerville]

Advantages

Loose coupling

More responsive

Increased reliability

Asynchrony built in

Events distributed leads to timeliness

Disadvantages

Difficult debugging

Maintenance overhead (fewer build time validations)

Different understanding of events can lead to problems

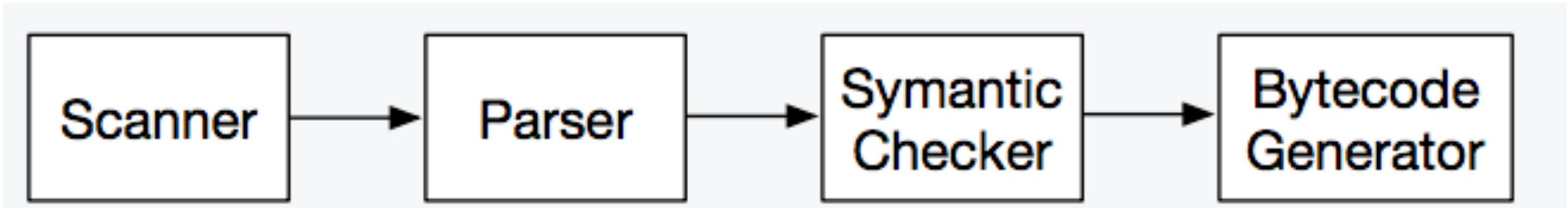
Pipe and Filter

Depicts Run-time organization of the system. Provides a structure for systems where each processing component is discrete and carries out one type of data transformation. Each step (filter) transforms the data and passes is on to the next step.

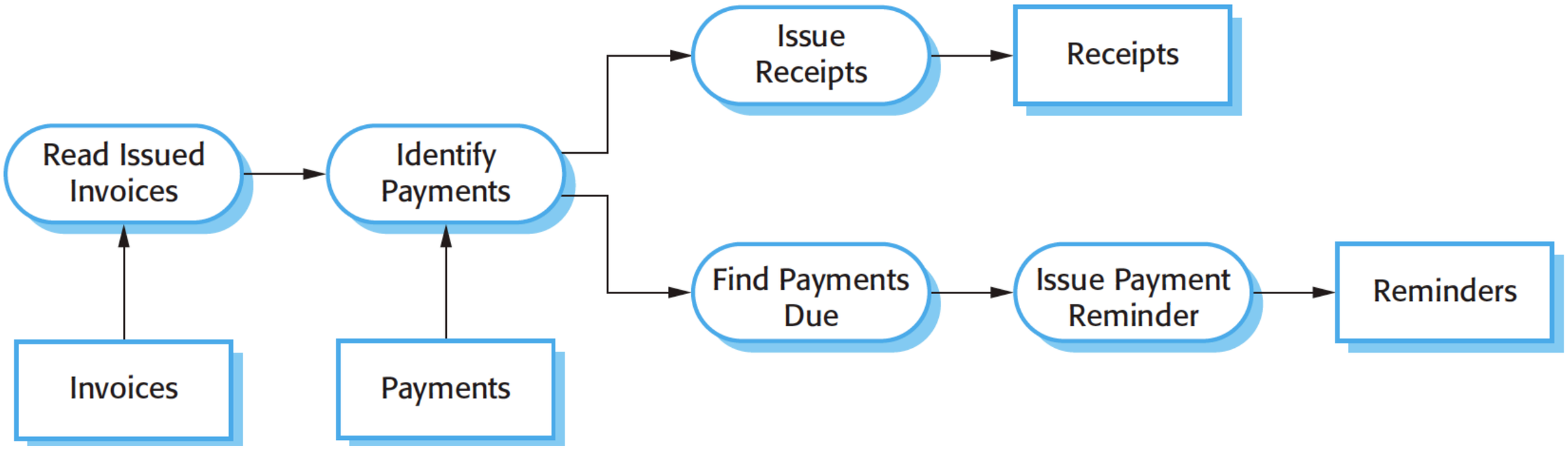
Can be sequential or parallel; single item or batch process

```
cat input.txt | grep "text" | sort > output.txt
```

Pipe and Filter



Invoice processing



Advantages

Easy to understand and matches business processes

Filters can be reused/ replaced

Ease of debugging

Can be sequential or concurrent

Disadvantages

Format of data transfer has to be agreed a-priori

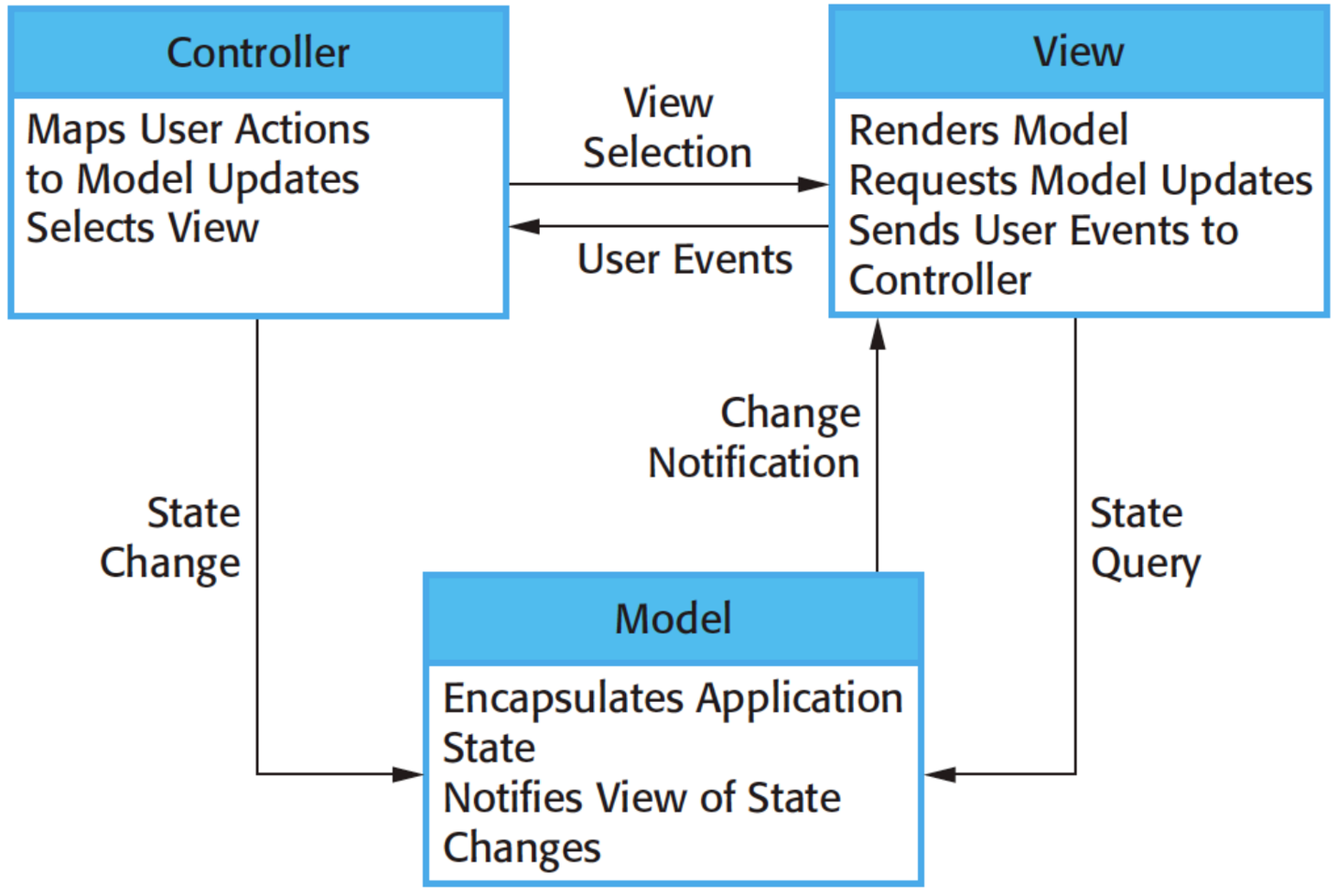
Each transformation must parse and serialize in the agreed form

Cannot reuse functional transformations that use incompatible data structures

Cannot (directly) share state between filters

Model-View-Controller

Separates information, presentation, and user interactions



Model

Manages the system data and associated operations on that data

Contains the Business logic. (application logic and structure)

View

Defines and manages how the data is presented to the user

Renders the model

Allows interaction with the user

Passes input to the controller

Controller

Manages user interaction and passes these to the view and the model

Receives input

Makes appropriate calls to the model

Updates the view

Model-View-Controller

Use when -

there are **multiple** ways to view and interact with the data

when future requirements for interaction and presentation of data are unknown

Advantages

Advantages

Clear separation of concerns

Allows data to change independently of its representation and vice versa

Multiple presentations to the same model

Single change to model updates all representations

Disadvantages

Disadvantages

Increased complexity, communication

Views & controllers are tightly bound

Common MVC frameworks

Ruby on Rails

Spring Framework for Java

Django for Python

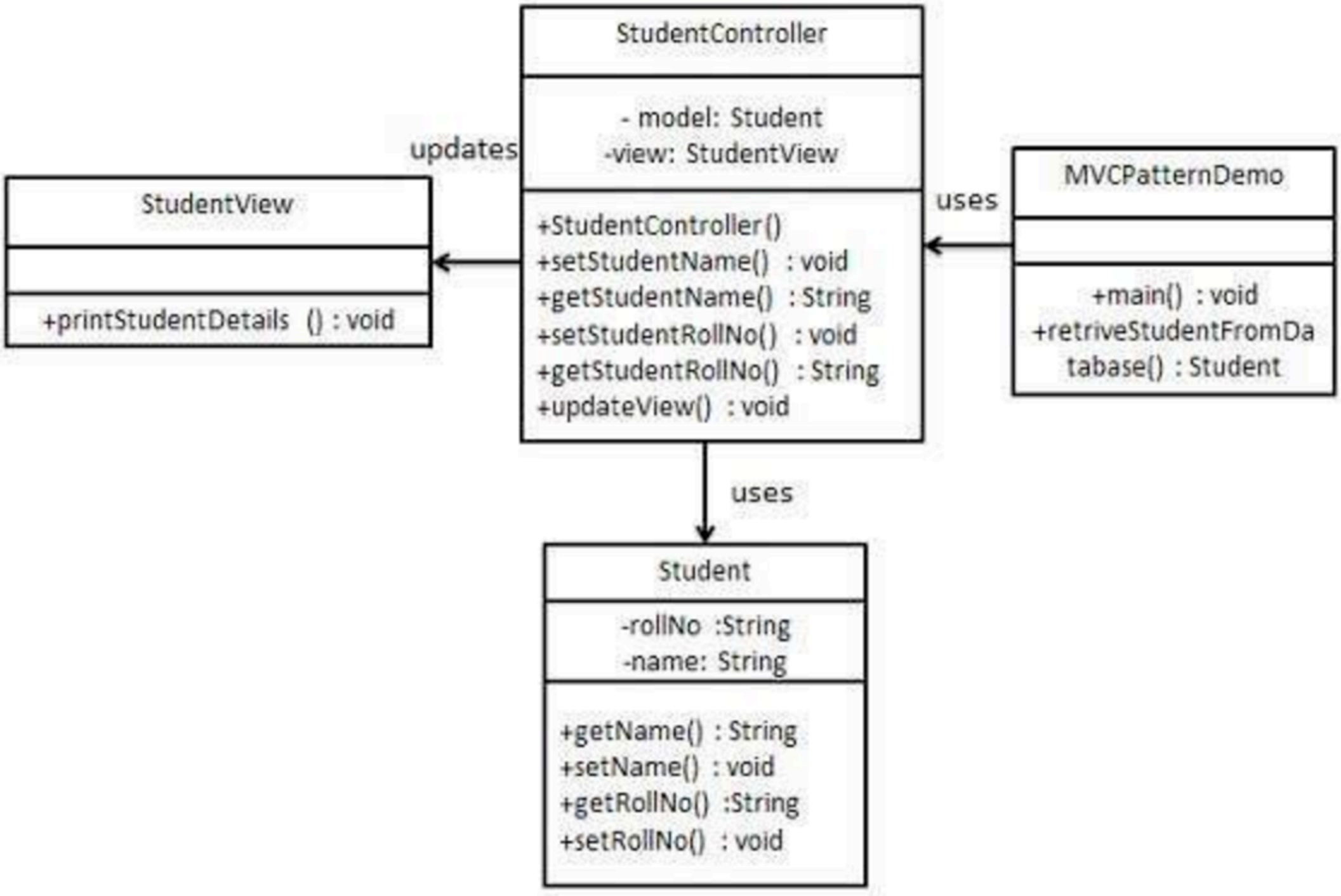
Google Web Toolkit for Java

AngularJS for Javascript

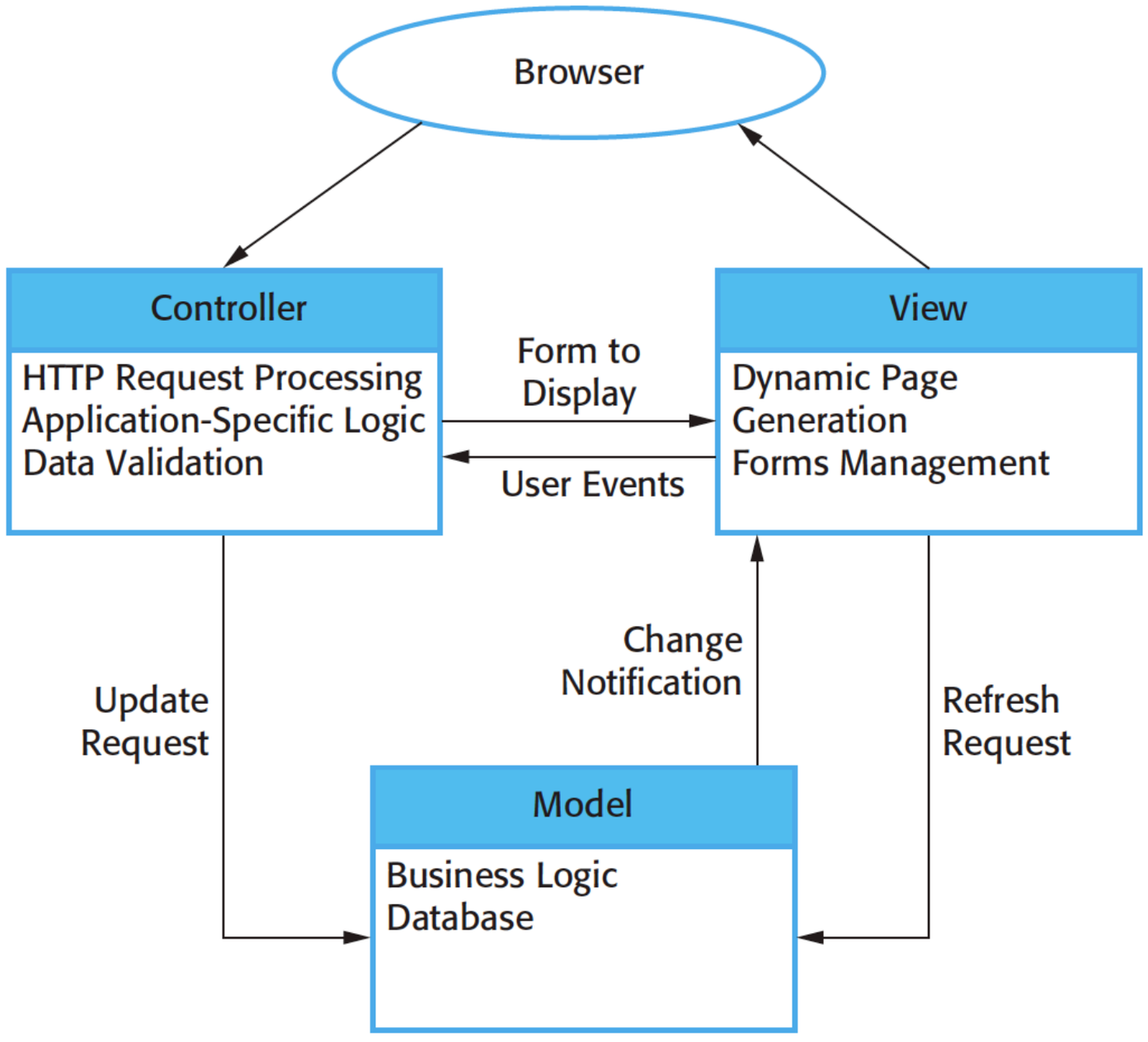
CodeIgniter for php

MVC – student record viewer

MVC – student record



Interaction in a (generic) web-based system



Bare Bones Facebook = BeaverBook

User profile (name, picture, status)

Wall that contains posts from your friends

Posts have poster's information, text, and comments

Comments have commenter's information, text

MVC for BeaverBook