Oregon State
University

Software Architecture

Today

- Discuss the mid-term course evaluation
- Another activity on Class Diagrams - they are important :-)
- Architecture
- Sprint 3 discussion

Mid Term Course Evaluation

- Overall
  - 130 responses
  - On most questions > 70% are “completely/agree” + neutral
  - Large class - so, I am trying to pace it to the majority

Oregon State University

Sprints

- **Expectations**
  - We are releasing the Sprints on Mon so that you can read it at home
  - I will spend 10 min in class on the Sprint
  - You need to plan the sprint around Tue so if you have any questions, we can discuss
  - 4 pair of eyes think the rubric and expectations are clear
    - so, please help us in helping you - ask questions in class, ask questions on Canvas

Oregon State University

Introduction to (new) topics/ slides

- **Pace and amount of detail on slides - mixed bag**
  - To reach to majority, I will pause after the slides to see if people are still taking notes
  - Use Doc Cam as white board
  - The readings are available on the schedule. Read **BEFORE** class
  - I will post the DRAFT slides before class… note they are in DRAFT stage at that point
  - When doing in-class exercises
    - I will spend more time with the problem statement
    - I will provide the "syntax" (e.g., for the diagram) on the doc cam – if I forget, remind me

Oregon State University

Reading…Reading….Reading

- Most people were unaware of it, so not everyone is doing it
- **VERY important** - slides are signposts - the readings are what will make things clear
- Read BEFORE class
- Readings are going to be used for exams.
  - And to get you in the habit of reading it….we will start to use it for the quiz too

Oregon State University

Examples

- The readings have more examples. And pointers to more examples
- We will go in depth today (class diagram) and next class for MVC architecture
- Because of class time limit - I can't cover more examples
  - But, I will make more available to you
  - On the sheet of page being passed, tell us what specific topic do you want more examples for
  - Have created all the UML diagrams for Library system

Real World application

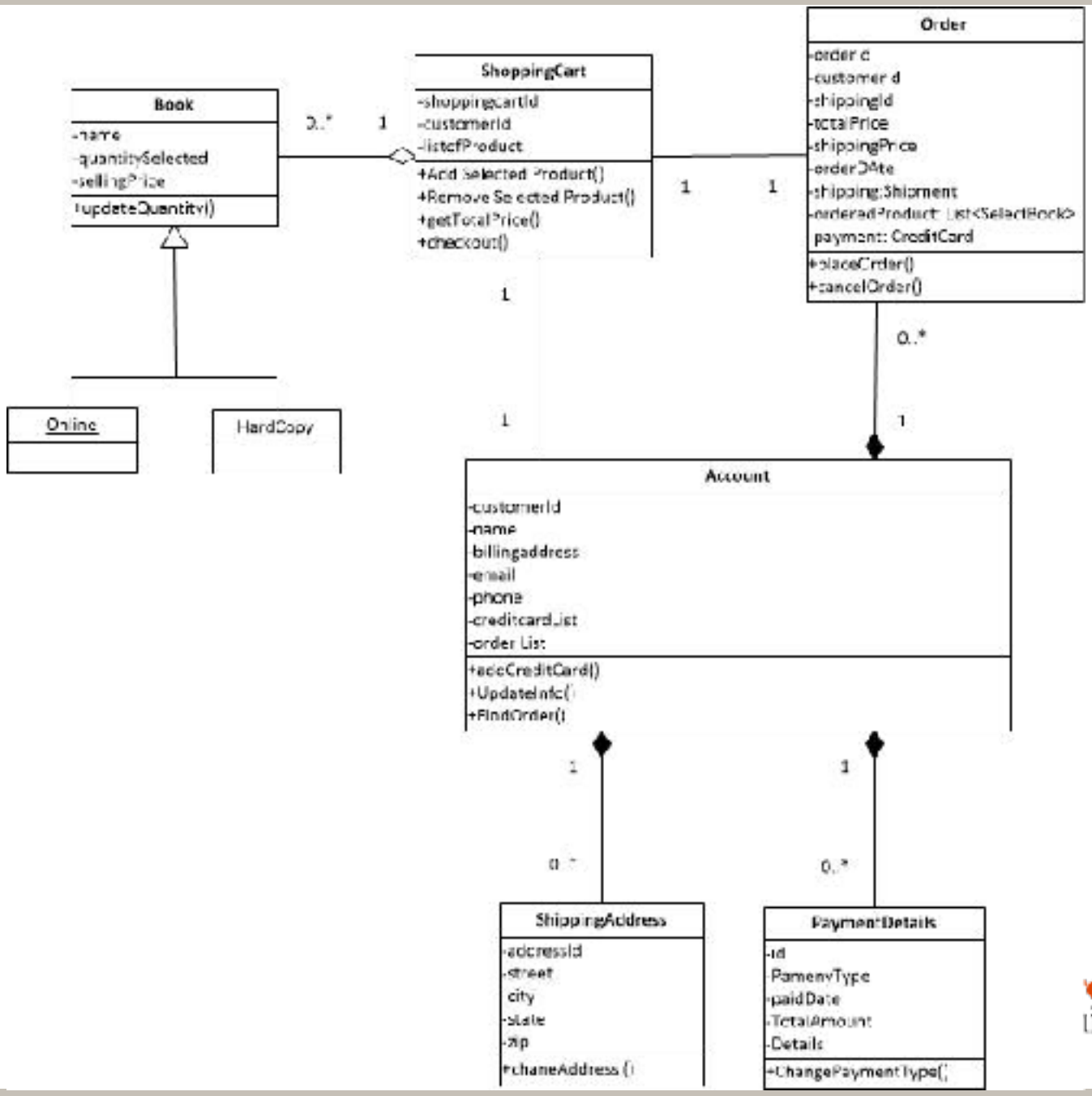
- You are learning a lot that has direct real world application
  - Git
  - GitHub
  - Workflow - pull request, comment - refactoring
  - Working on an existing code base
  - The activities and Sprints allow you to learn by doing

- We will have the interviews where you can ask these accomplished industry folks any questions
  - Career path
  - How certain concepts/topics are dealt with in industry

Oregon State University

Class Diagram Exercise

- Book buying in Amazon system
  - Books are written by authors
  - Books can be online or hard copy
  - Books are put in a shopping
  - User has account
  - An account can have multiple shipping addresses (any number)
  - An account can have multiple payment types (max 3 types)
  - Account stores orders (made)


Book
-name
-quantitySelected
-sellingPrice
+updateQuantity()
Online
HardCopy
0..*
1
ShoppingCart
-shoppingcartId
-customerId
-listofProduct
+Add Selected Product()
+Remove Selected Product()
+getTotalPrice()
+checkout()
1
1
Order
-order c
-customer d
-shipping Id
-total Price
-shipping Price
-order Date
-shipping:Shipment
-ordered Product: List<SelectBook>
payment: CreditCard
+placeOrder()
+cancelOrder()
0..*
1
1
1
Account
-customerId
-name
-billingaddress
-email
-phone
-creditcard_list
-order List
+addCreditCard()
+UpdateInfo()
+FindOrder()
1
0..*
1
0..*
ShippingAddress
-addressId
-street
-city
-state
-zip
+changeAddress()
PaymentDetails
-id
-PamenyType
-paidDate
-TotalAmount
-Details
+ChangePaymentType()

Architecture

Patterns

- A general, reusable solution to a commonly occurring problem in a given context
- Often have best practices associated with them

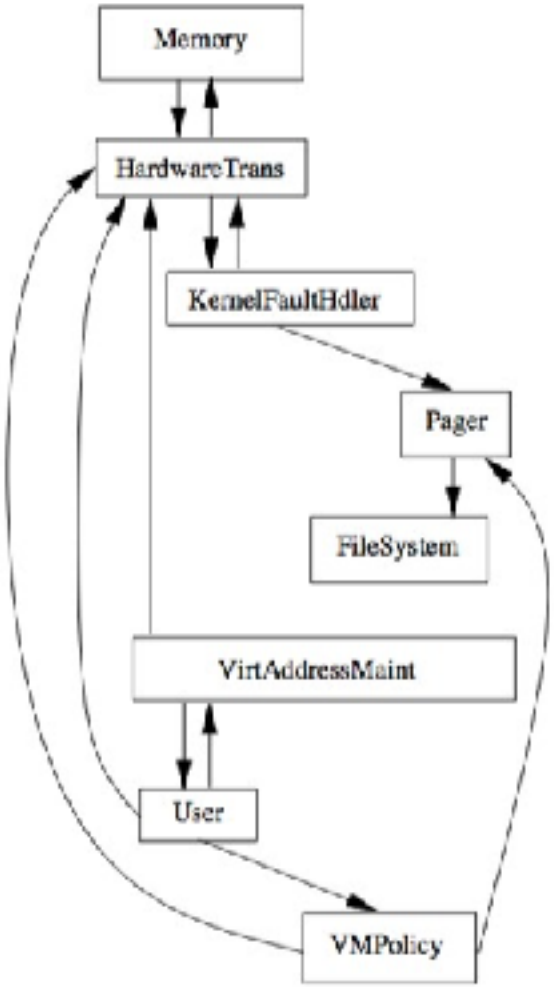
Patterns

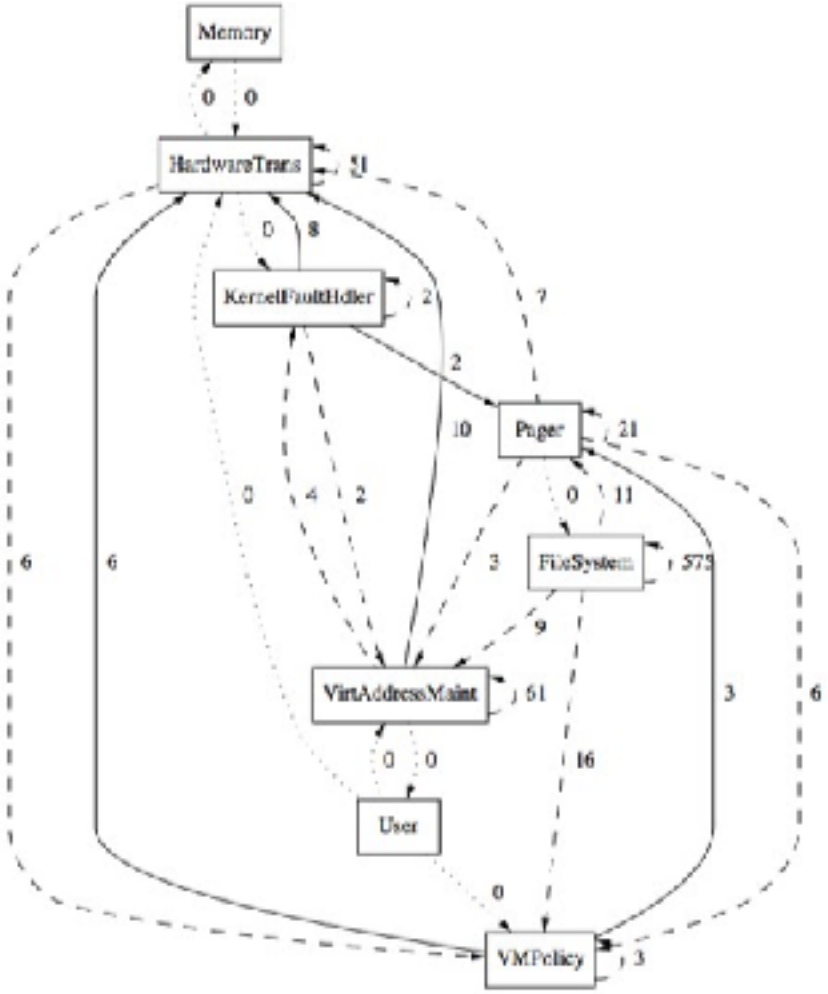
- Architectural Patterns Fundamental structural organization for software systems
- Design Patterns Solves reoccurring problems in software construction

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





How is architecture usually specified?

Oregon State University

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. …

Oregon State University

Architectural Styles

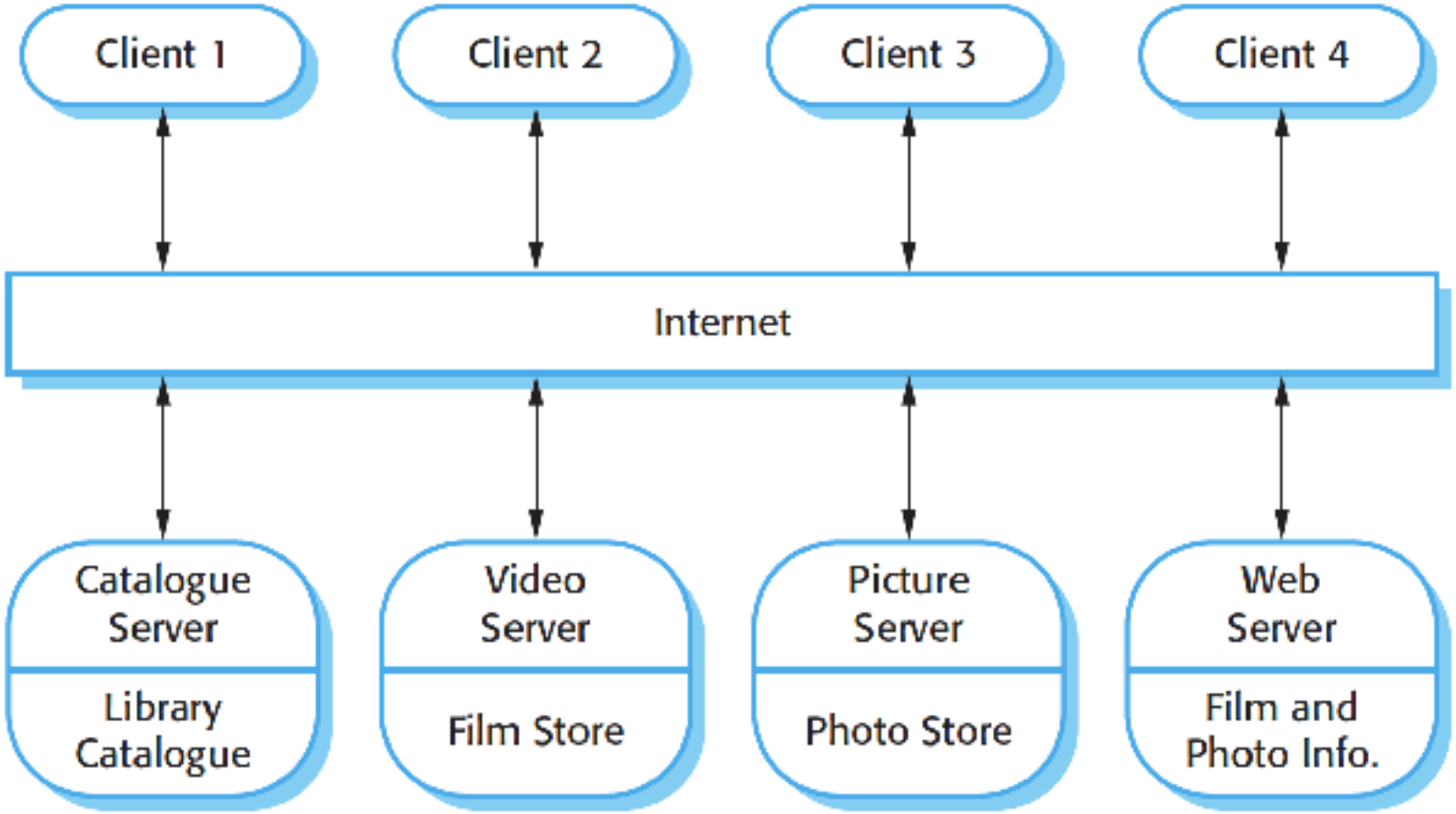
- Client and Server
- Service-Oriented
- REST
- MVC
- Layered
- Event-driven
- Pipe and Filter

Client-server Architecture

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



Oregon State University

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

Service Oriented Architecture

- The extreme generalization of Client-Server
- Instead of monolithic systems one has many concise services
- A Service is a "*loosely coupled, reusable software component, which can be distributed*"
- Services use message based communication
- Service discovery becomes a challenge

Oregon State University

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

https://restfulapi.net/

Oregon State University

For Amazon

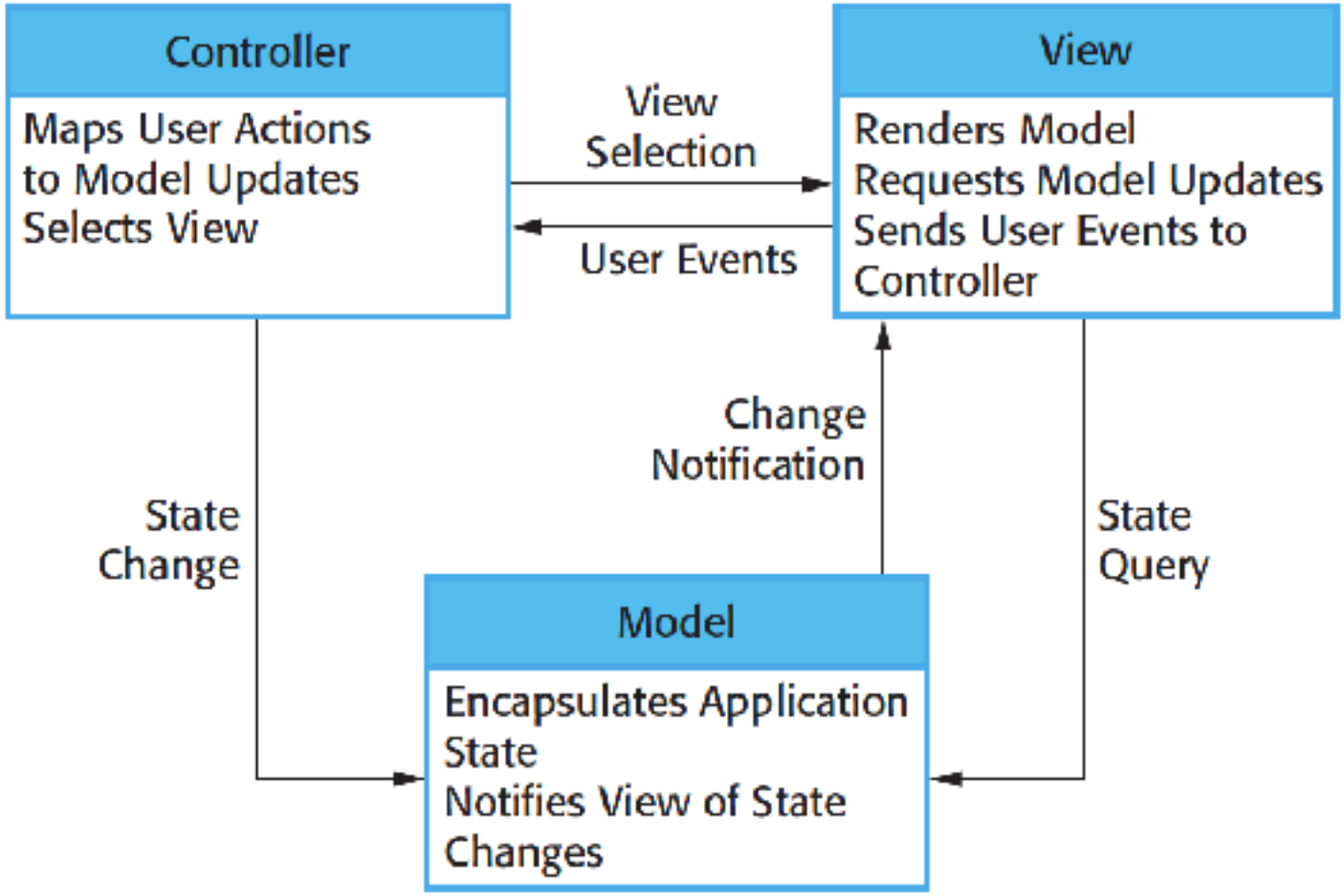
- How would the client server architecture look like?
  - What components would reside in the client? In the server?

Quiz

Oregon State University

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

Model-View-Controller

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

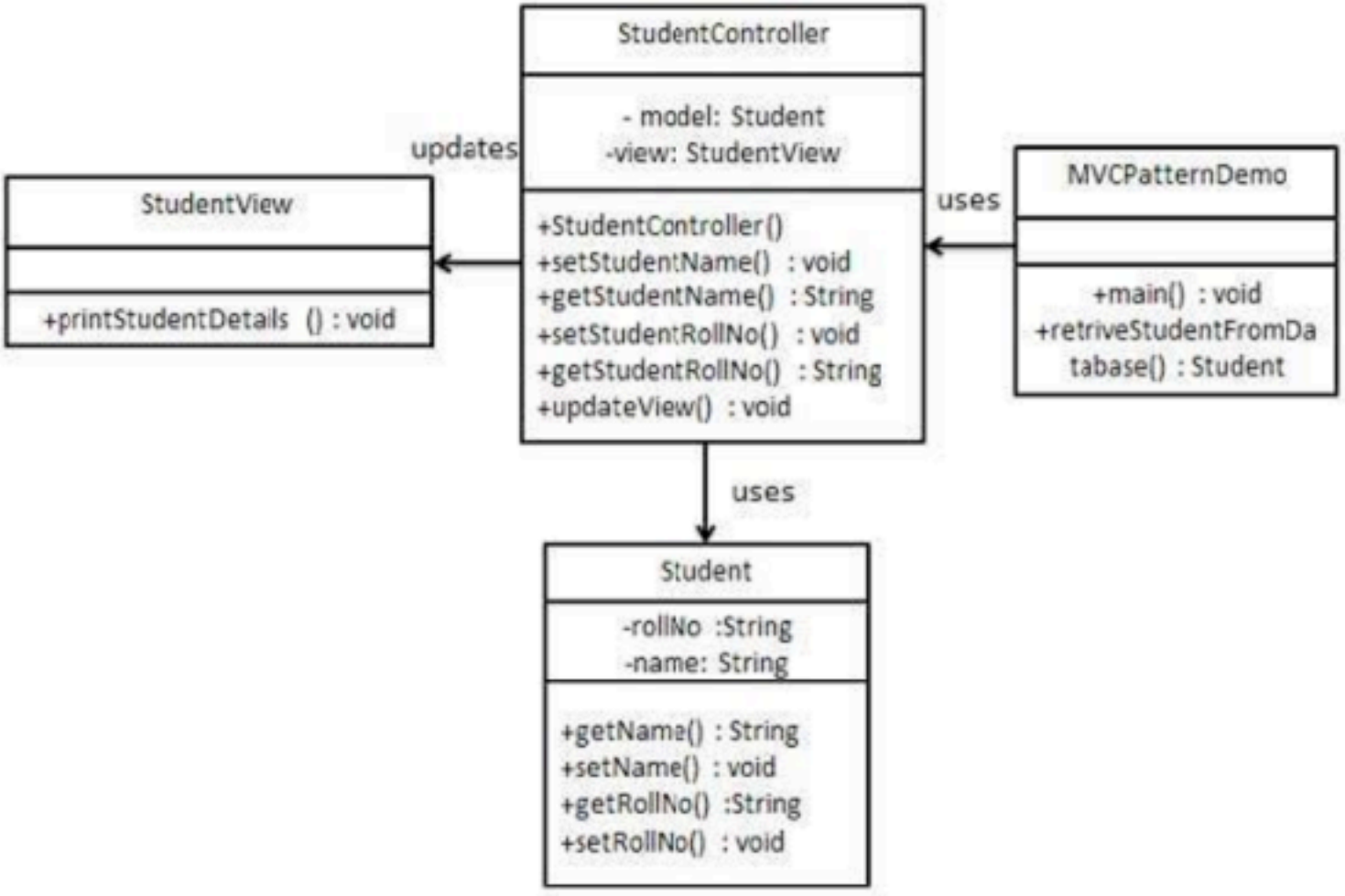
- Increased complexity, communication
- Views & controllers are tightly bound

Oregon State University

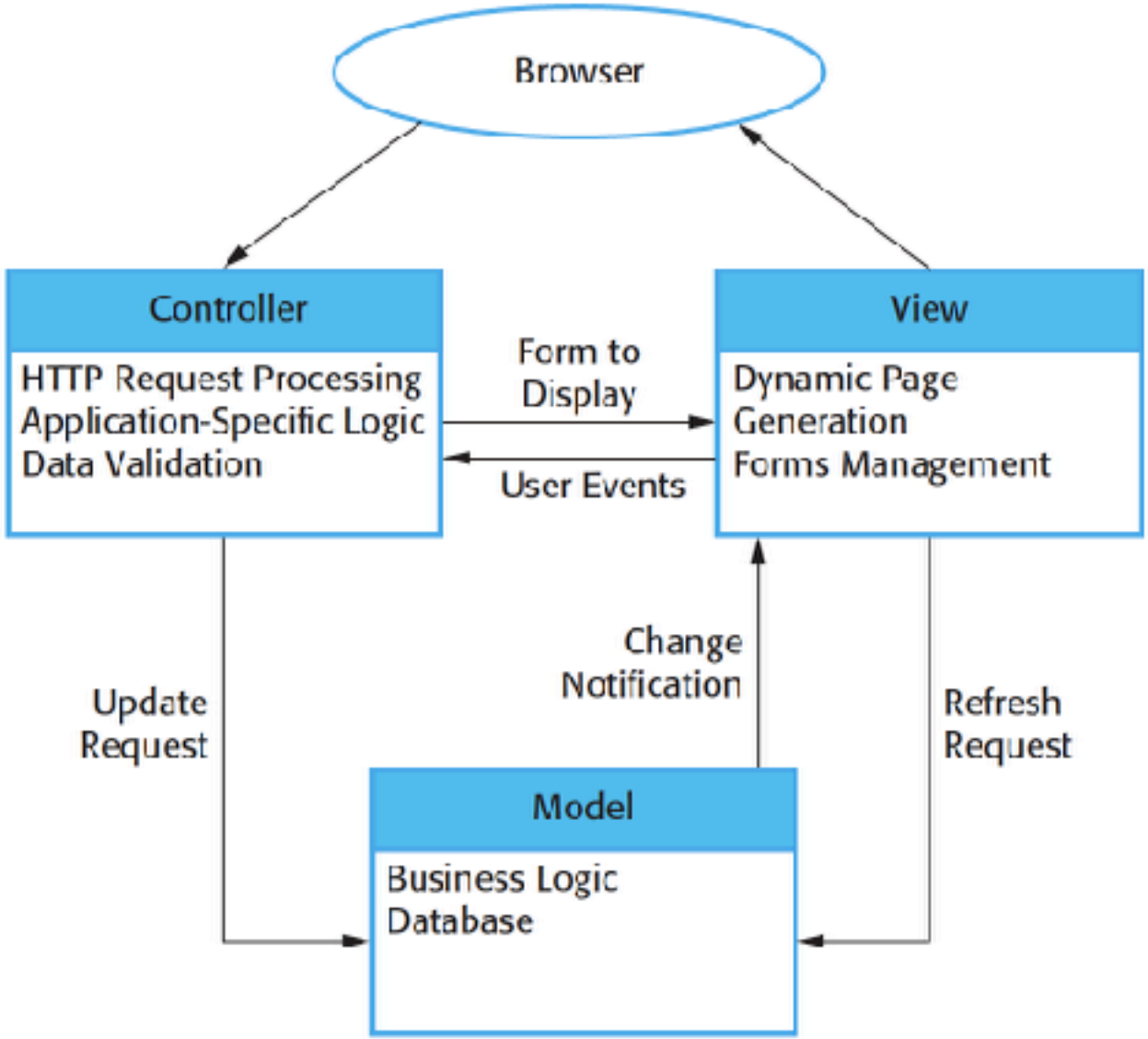
Common MVC frameworks

- 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



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

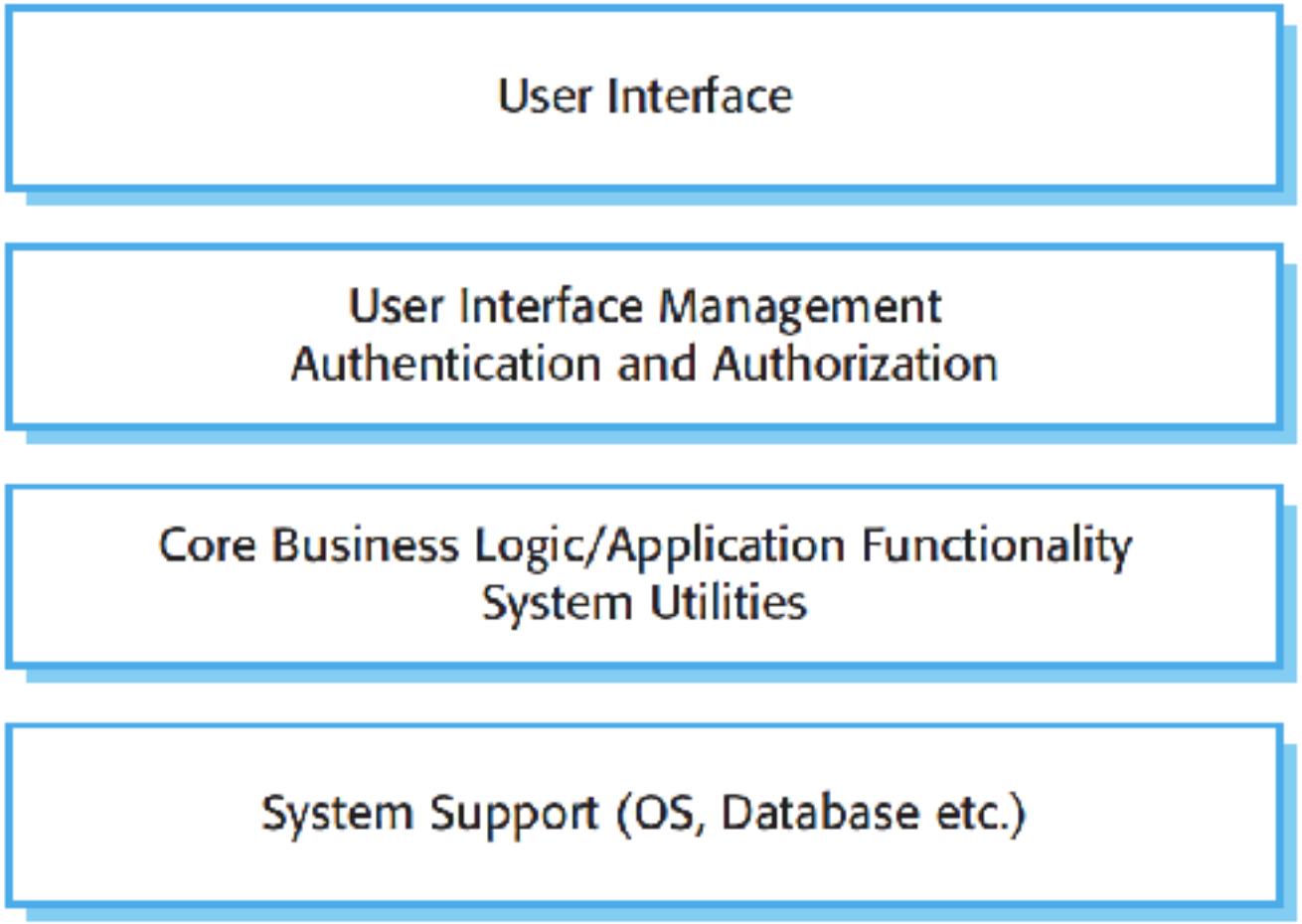
Layered Architecture

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

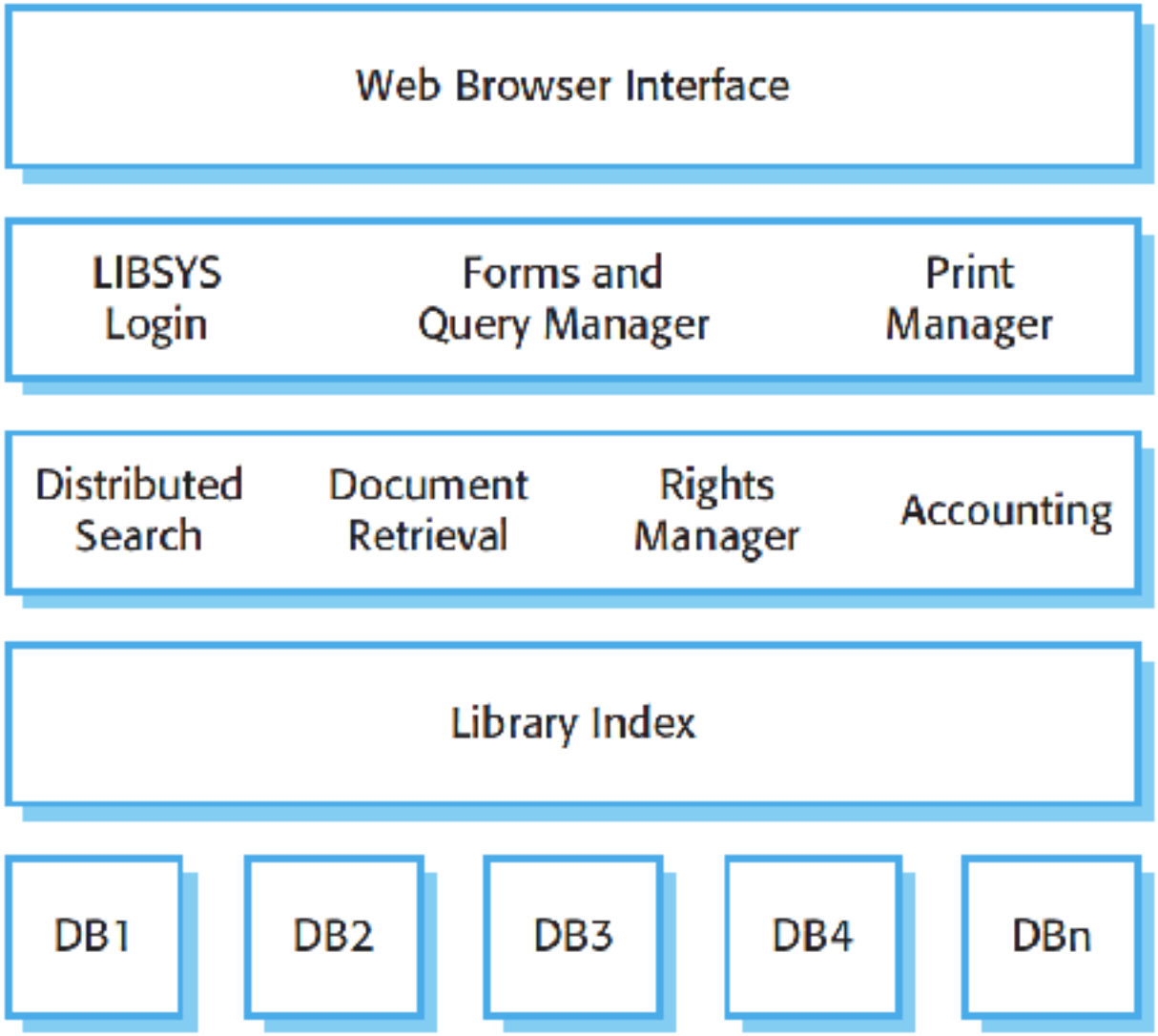
- Each layer provides a set of services to the layer "above".
- Normally layers are constrained so elements only see
  - Other elements in the same layer, or
  - Elements of the layer below

Oregon State University

Generic Layered Architecture

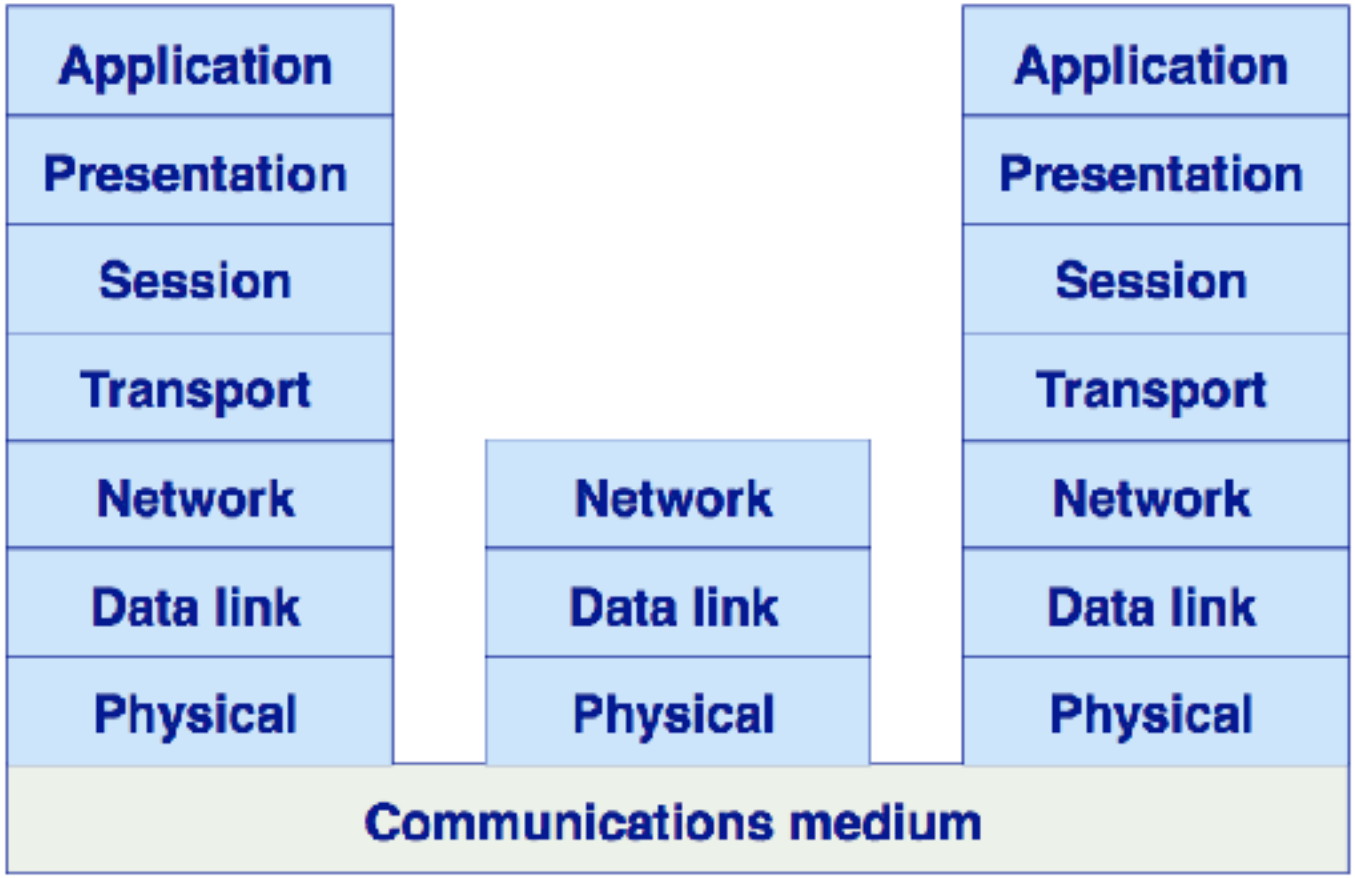


Online Library System

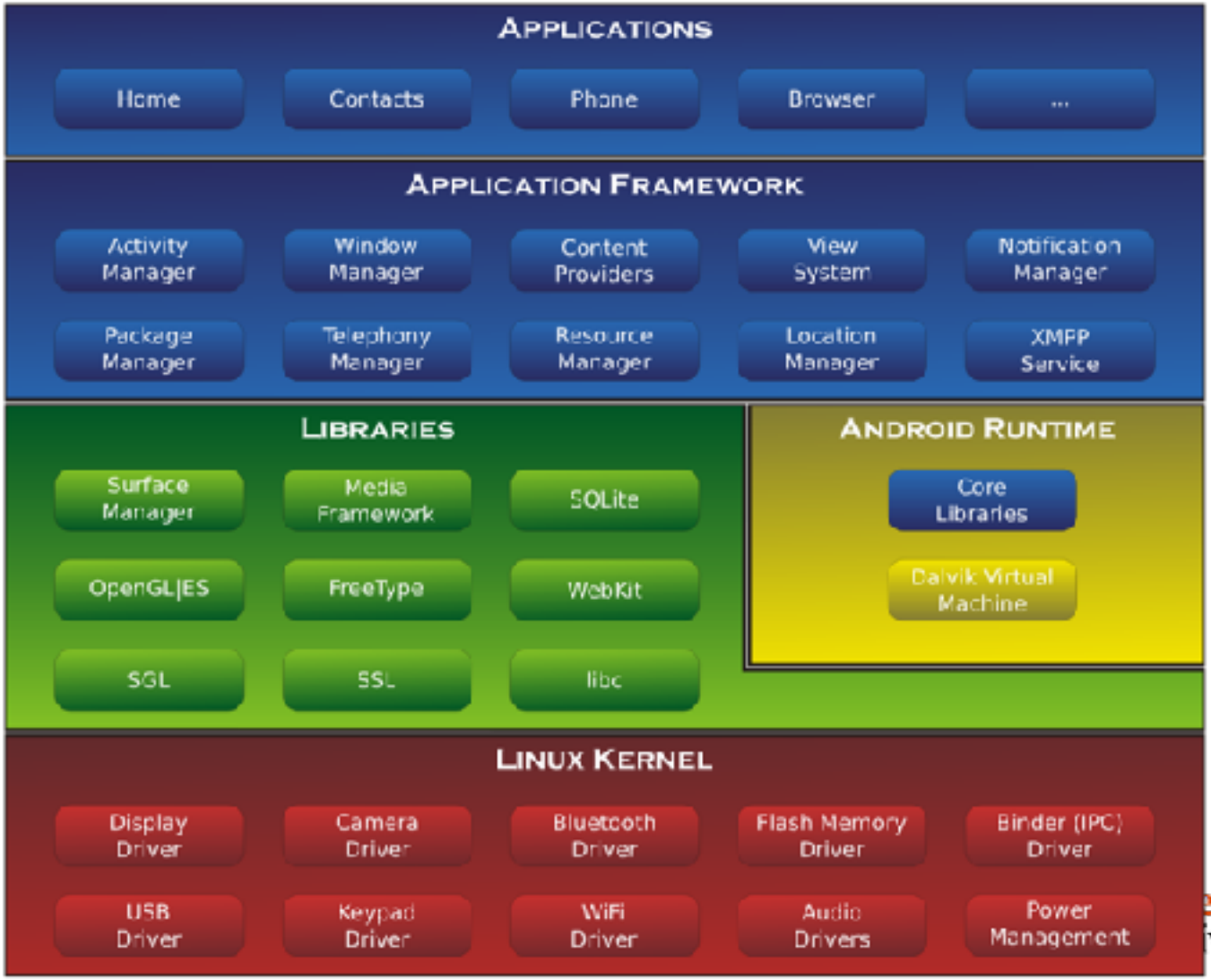


Oregon State University

OSI reference model



Android Architecture



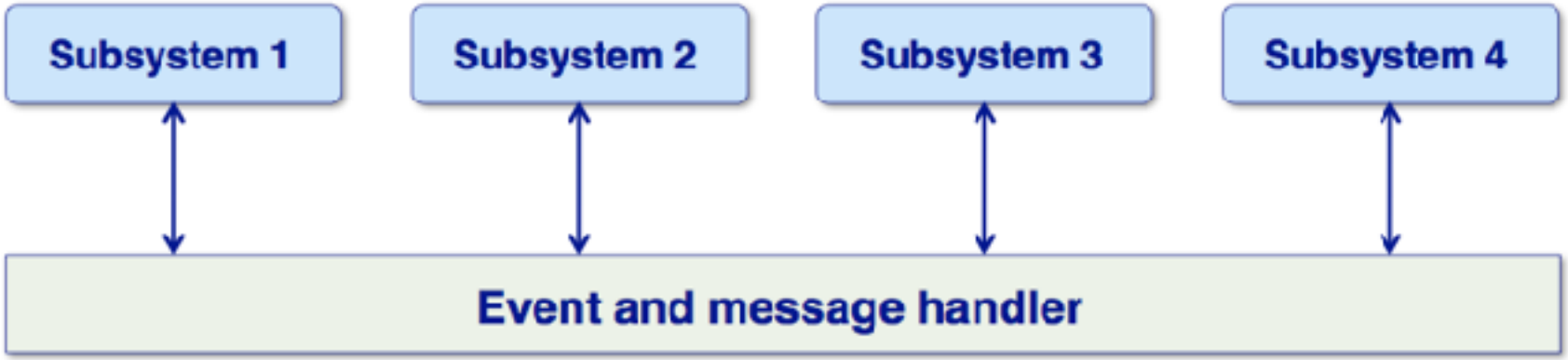
Advantages

- 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



[Sommerville]

Oregon State University

Event Driven Architecture

Advantages

- Loose coupling
- More responsive
- 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

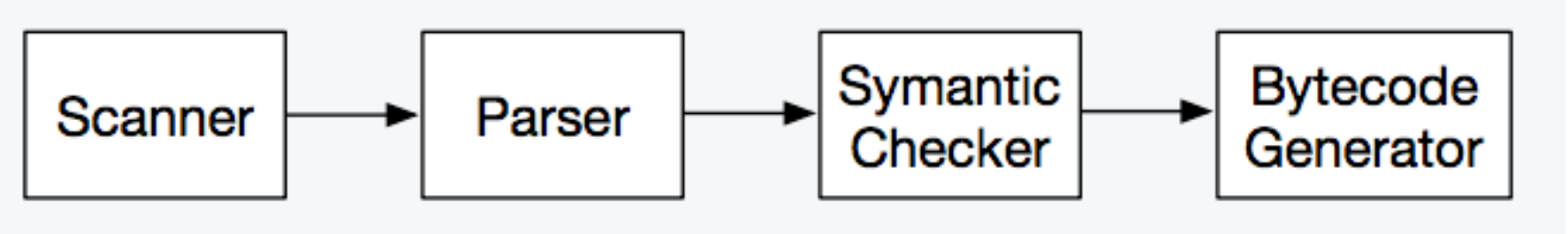
Depicts Run-time organization of the system. Provides a structure for systems where each processing component is discrete and carries out one tope of data transformation. Each step (filter) transforms the data and passes is on to the next step.

Can be sequential or parallel; single item or batch process

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

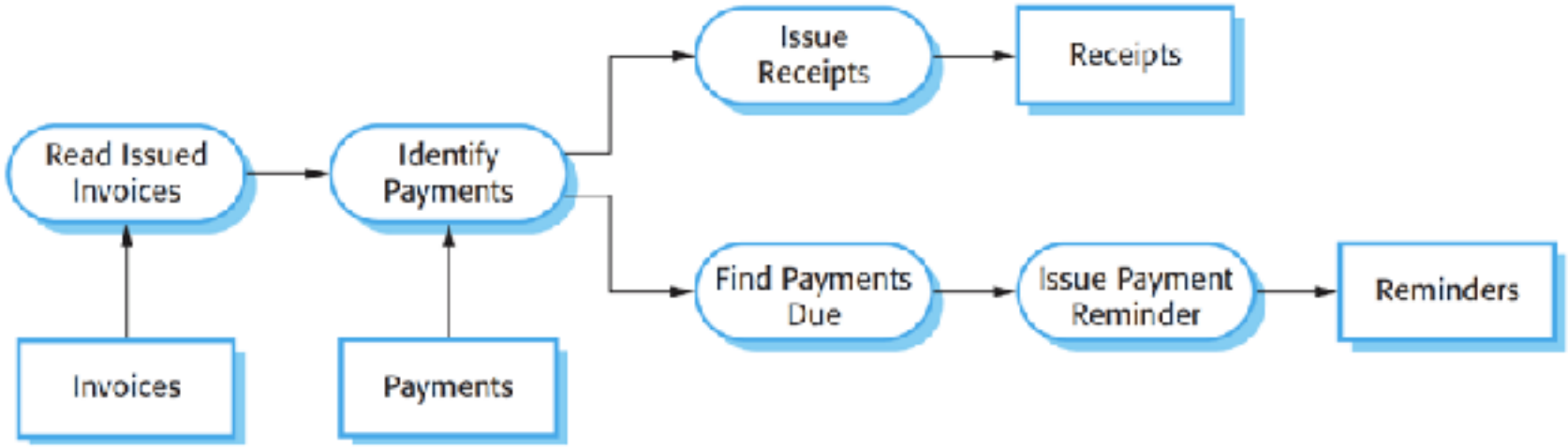
Oregon State University

Pipe and Filter



Oregon State University

Invoice processing



Pipe and Filter

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 its input/ unparse its output in the agreed form
- Cannot reuse functional transformations that use incompatible data structures
- Cannot share state between filters
- Cannot share state between filters

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

Oregon State
University