Overview of Architectural Design

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

April 3, 2025

Outline

Outline

Recall: Requirements

1. Discussed
   - Overview of requirement engineering
   - Agile vs. traditional (plan & document)
2. An agile approach of requirement analysis
   - Design user stories for/as requirements
   - In Behavior-Driven Development (BDD), map a user story to one or more scenarios
   - Question: How do we ensure that our "stories" are acceptable by the users? $\rightarrow$ acceptance tests?
   - Each scenario can be an acceptance test
3. Your project
   - User stories, storyboards, scenarios
   - How do we do acceptance testing here?
   - How do we do unit tests?

We begin with a design of the software when we start building the software based on the requirements

An Analogy: Building Houses

- What does "designing a house" mean?
- How do we design a house? Is it a necessary step?
- Are there certain properties common to groups of houses?
  - Swiss chalet vs. Suburban house vs. multiunit apartment buildings
- What skills should architects possess?

Designing Houses vs. Designing Software

Compare designing houses to software design

- Do you think software design is necessary?
- Are there properties that are exhibited by the architectural design?
- Who are software architects?

Think about the implications of the medium

- Software is developed on the same medium as the blueprint

Software Design

- Design starts mostly from/with requirements – evolving mostly from functionalities and other non-functional characteristics
  - In the waterfall model Design generally occurs after Requirements
  - In agile, design is performed during in each iteration
- To answer: How is the software solution going to be structured?
  - What are the main components – (functional composition) often directly from requirements' functionalities (e.g., use cases, user stories, scenarios)
  - How are these components related? – Possibly re-organize the components (composition/decomposition)
- Two main levels of design:
  - Architectural (high level) design
  - Detailed design
  - Different design concerns at different abstraction levels (e.g. classes vs. modules vs. entire system)
- How should we depict design – what notation/language?

Outline

Architectural Design

Highest level of design, closest to requirements

- Definition: The set of principal design decisions made during the development of the system and its subsequent evolution
- Purposes: Sustainability of the developed system
  - e.g. architect for performance, reliability, resiliency, availability, etc.
- Effective project communication
- Adequate basis for reuse

What is software architecture?

Software Architecture

- Every software system has an architecture.
- Structure(s) of the solution, comprising:
  - Major software elements
  - Their externally visible properties
  - Relationships among elements
- May have multiple structures – multiple ways of organizing elements, depending on the perspective (multiple architectural designs from different perspectives)
- External properties of components (and modules)
  - Component (module) interfaces
  - Component (module) interactions, rather than internals of components and modules

Architecture View and Viewpoints

- View – representation of a system structure, e.g.,
- 4+1 Views (by Krutchen)
  - Logical (OO decomposition – key abstractions)
  - Process (run-time, concurrency/distribution of functions)
  - Subsystem decomposition
  - Physical architecture
  - +1: use cases
- Other classification (by Bass, Clements, Kazman)
  - Module
  - Run-time
  - Allocation (mapping to development environment)
- Different views for different people

Depicting Architecture

Commonly use Boxes and Arrows

- Boxes: system components (or modules)
- Arrows: communication (of some type)

Example System Architecture

- Boxes = modules
- Arrows = communication Channels

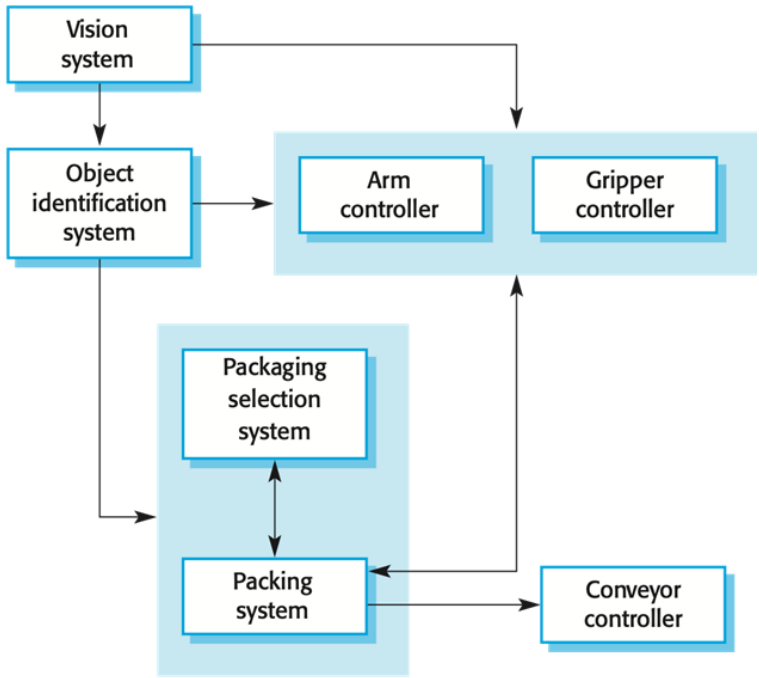
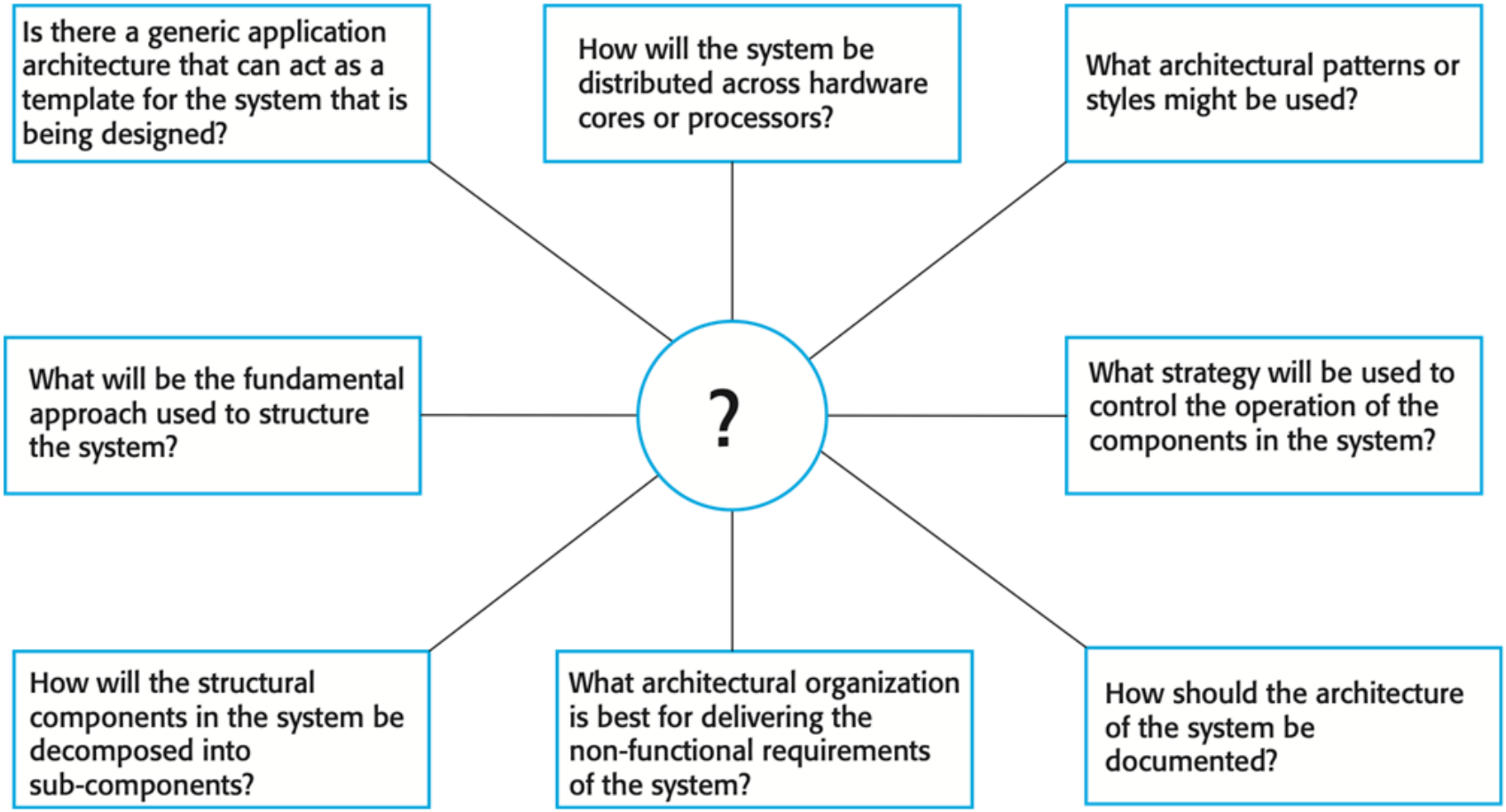


Image credit: Sommerville, 10th edition

Making Architectural Design Decisions



Impact of/on Implementation and Maintenance?

Prescriptive – What we plan to build?

- Usually some sort of model or document

Descriptive – What we end up with?

- As time goes on, prescriptive turns into descriptive
- Usually the structure of the code

Implementation has an influence, so there may not be an exact mapping

Architectural Degradation

Architectural drift

- Introduction of new principal design decisions into the descriptive architecture that were not included in the prescriptive architecture

Architectural erosion

- Introduction of new principal design decisions into the descriptive architecture that violate its prescriptive architecture

Outline

Architectural Design Patterns

- Patterns are a means of representing, sharing and reusing knowledge.
- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
- The grand tool: Refined experience
- Learn from success and from failure
- There is no virtue in reinventing the wheel, broken or otherwise

Some History of Design Patterns

Popularized by the Gang of Four book

- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. 1995
- Aimed at OO software implementation

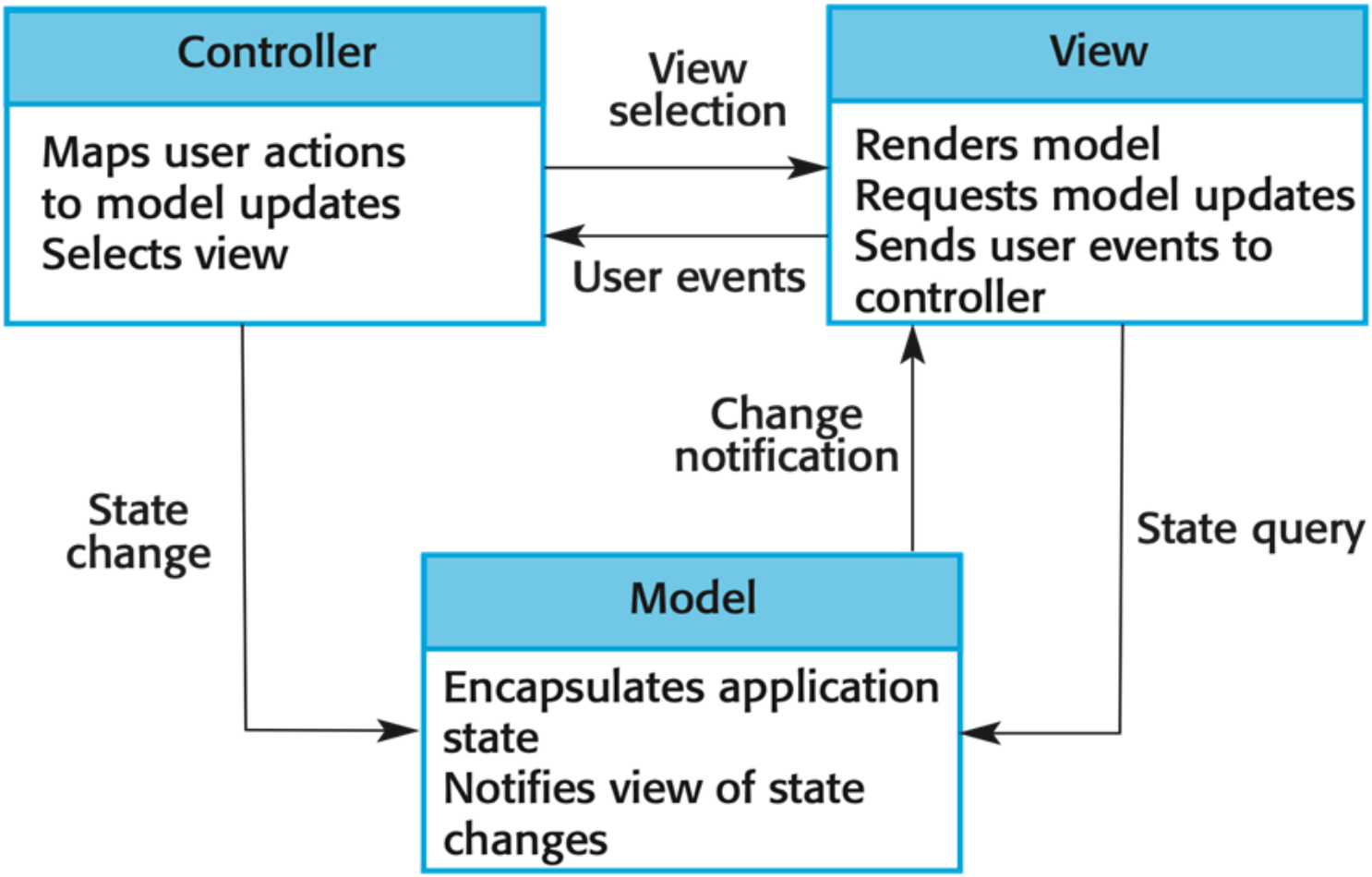
After the introduction of this book, all kinds of patterns conferences and books emerged

At present, we are discussing architectural patterns, not OO patterns

Architectural Design Patterns

- Model-view-controller (MVC)
- Layered design
- Distributed Layered – Client-server, Peer-to-peer
- Dataflow Pattern – Pipes and filters
- Implicit Invocation
- Event driven
- Database centric
- Three tier
- ...

Model-View-Controller (MVC)



Using MVC

Used in many places – generally in applications with a UI and data to show in that UI, e.g. web applications

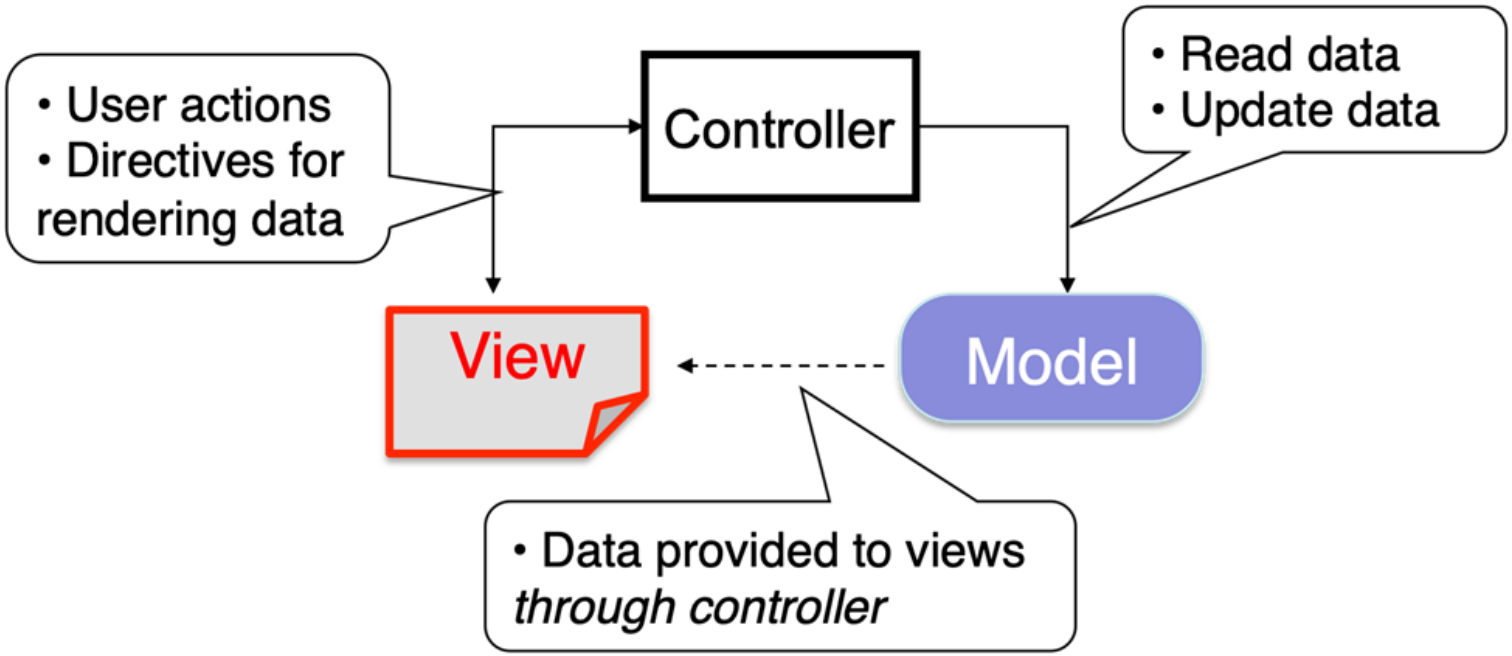
- Benefits
  - Separation of concerns
- Disadvantages
  - Extra code

Probably one of the best patterns for your mobile app

Alternative View of MVC

Goal: separate organization of data (model) from UI & presentation (view) by introducing controller

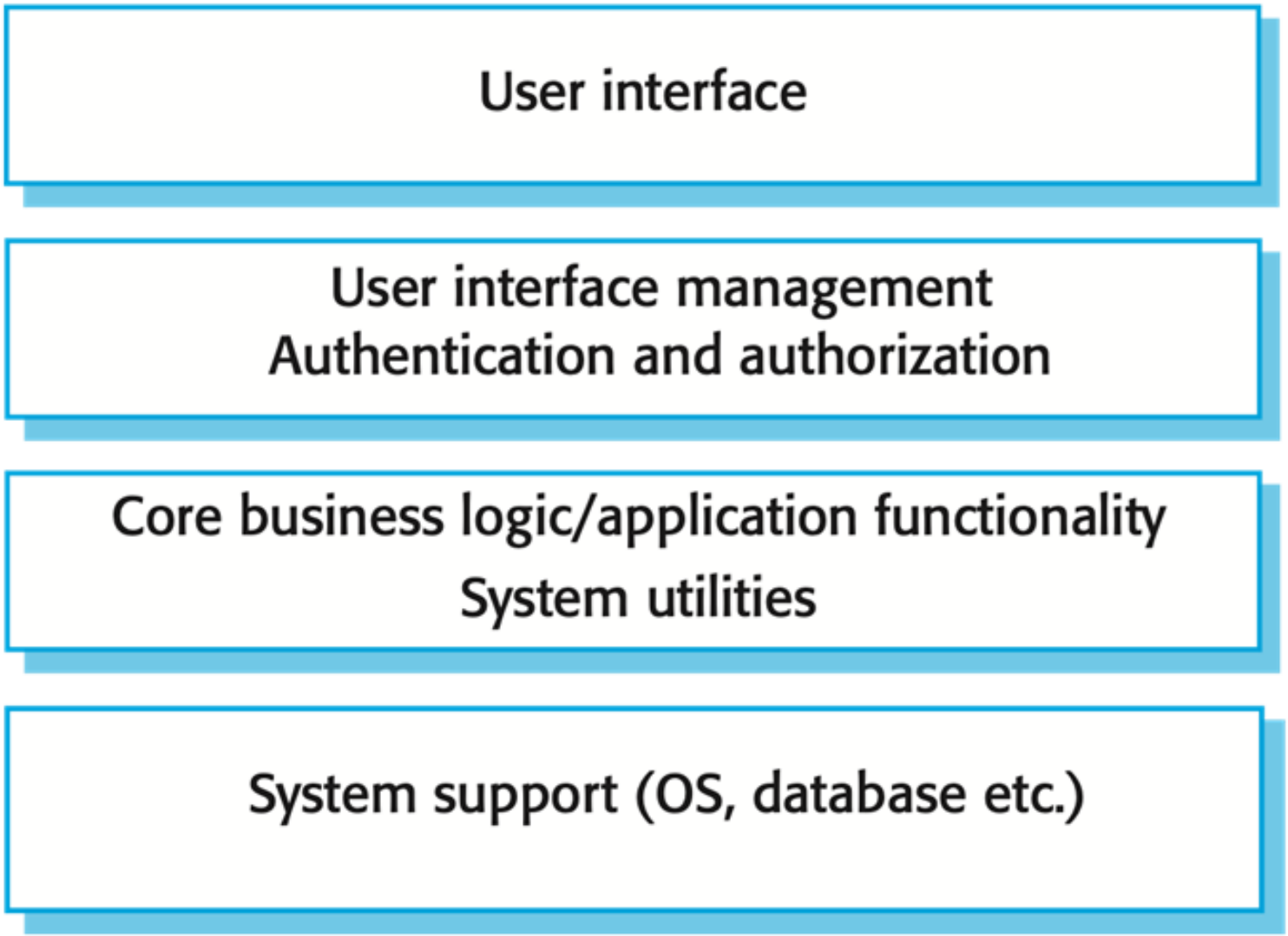
- mediates user actions requesting access to data
- presents data for rendering by the view



Mapping MVC to Android

- View = XML representation of UI
- Model = depends, e.g., (Data) Classes, SQLite DB, Web Tools
- Controller = Various listeners and callbacks on user behavior

Layered Design



Using Layered Design

Also used in many places, e.g. O.S.

- Benefits
  - Separation of concerns
  - Abstract away complexity into each layer
- Disadvantages
  - Requires good interfaces
  - Performance can be an issue when many layers

Distributed Layered Design

Distributed – assumes that each layer can exist on a separate machine

- Client-Server
  - Application split into client component and server component
  - Client issues request to server, gets response
  - Usually assumes that the number of clients >= number of servers
- Peer-to-Peer
  - Solves problem of "single point of failure/congestion" in Client - Server model
  - Adds layers of complexity in discovery, availability

Data Flow Pattern

Pipe and Filter

- The high-level design solution is decomposed into two "generic" parts: filters and pipes
  - Filter is a service that transforms a stream of input data into a stream of output data.
  - Pipe is a mechanism or conduit through which the data flows from one filter to another.
  - A basic example is the UNIX shell, e.g., `cat file.txt | wc`
  - Variants of this approach are very common
- When transformations are sequential, this is the Batch Sequential pattern
  - Transaction system (e.g. credit card transaction processing)
  - Not really suitable for interactive systems

Implicit Invocation

Callback driven

- Good example is Android application development
- Many UI driven systems are implemented in this way

Publish-Subscribe

- Subscribe to receive messages on a particular topic
- Publishers send messages on that topic
- Usually assumes that publishers and subscribers operate independently
- No shift of control to receive message

Future Topics on Design

- Services and service oriented architecture
- Software as a Service
- UML – language to depict design
- OO design

Outline

"Engineering Software as a Service" by Armando Fox and David Patterson (2nd Edition)
"Essentials of Software Engineering" by Frank Tsui, Orlando Karam, and Barbara Bernal(4th Edition)