CP317B Software Engineering

Architecture design, part-1 – week 3-1

Shaun Gao, Ph.D., P.Eng.

Agenda

- Review week 2
- Introduction to high-level design
- Software Security
- User interface
- Architecture design
  - Monolithic
  - Client/server
  - Component-based
  - Service-oriented
  - Event-driven
  - Data-centric
  - Distributed
- Summary

Review week 2

- MOSCOW method
- FURPS and FURPS+ methods
  - Functional requirements, Non-functional requirements
- Requirement elicitation
  - Six techniques: communication, task analysis, domain analysis, brainstorm, prototype, observation
  - Alex F. Osborn four rules about brainstorm
- Requirements documentation
- Requirements validation and verification
  - Verification vs. validation

Introduction

- Design
  - Design is a problem-solving process whose objective is to find and describe a way to implement the system's functional requirements while respecting the constraints imposed by non-functional requirements (for example, performance, security, and etc.) and while adhering to general principles of good quality.
- High-level design(HLD) – architecture design
  - HLD provides a view of the system at an abstract level. It shows how the major pieces of the software product will fit together and interact with each other.
  - HLD describes an architecture that would be used for developing a software product.

Architecture design - project

- Think-Pair-Share

- Question: Architecture design of group project

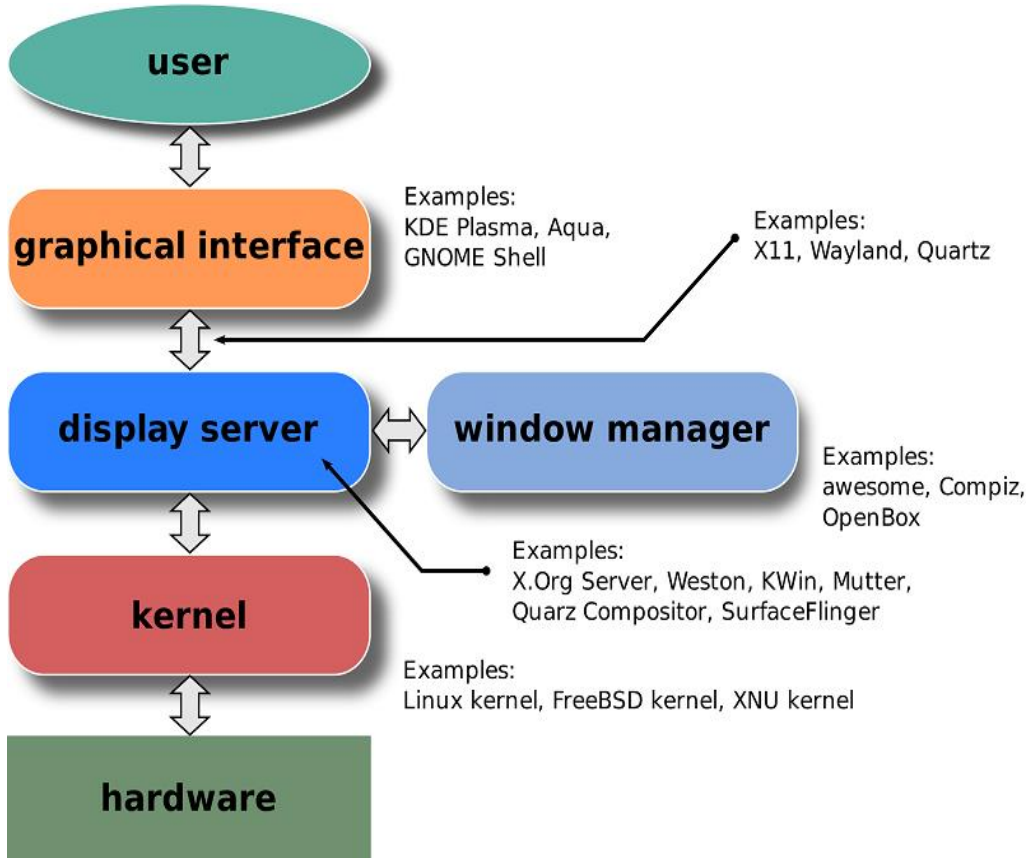
  - Design and develop a software application to read two text files and format the data output to a new file (Note: the format of input file and output file is described in Appendix A). The design should contain both architecture design and detailed design.

High-level design considerations

- Reliability, software product continue working when receiving any inputs.
- Quality,
- Software Security
  - **Software security** is an implemented idea/technique that protects a software system against malicious attack and other hacker risks so that the software system continues to function correctly under such potential risks.
- Software security should be considered at system design level
- Software vulnerability is the main cause of cyber security.
  - Buffer overflow attacks
  - SQL injection
  - Cross-site Scripting
  - …

High-level design considerations – cont.

- User interface
  - User interface is a (visual) part of a computer application or operating system through which a user interacts with a computer or a software system.
  - Examples of user interfaces
    - Keyboard, mouse
    - Touch screen, voice recognition,
    - Windows login screen

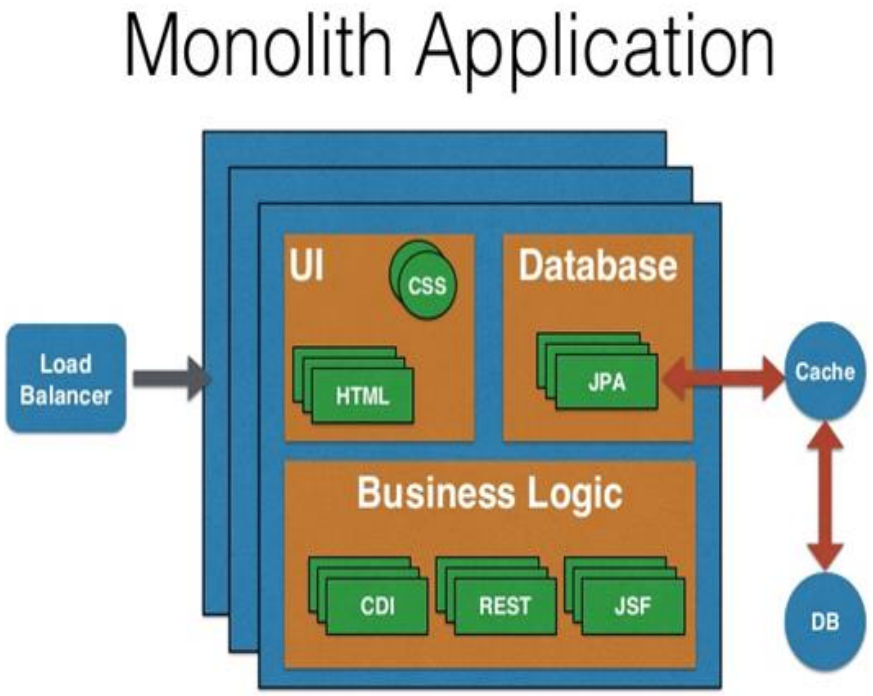


Architectural design

- What is Architecture?
  - Architecture is a high-level model of a thing
- Architectural design of software engineering
  - Architectural design is the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a software system.
- Models of architecture design
  - Monolithic
  - Client/server
  - Component-based
  - Service-oriented
  - Data-centric
  - Event-driven
  - Distributed

Architecture design

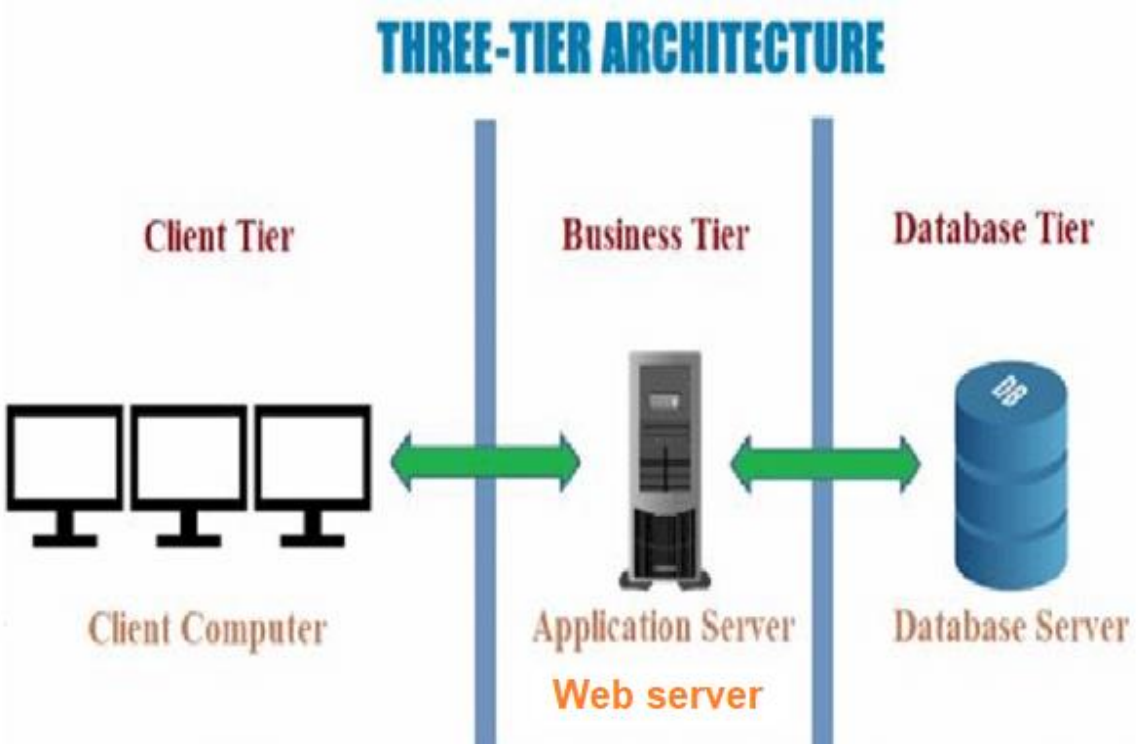
- Monolithic architecture (MA)
  - MA is to put all functions into a single program that does everything.
  - Advantage:
    - No need for communication across networks
  - Disadvantage:
    - Lack flexibility and scalability
- Examples:
  - Software used in current washing machine
  - Software used in current home appliances



JPA: Java Persistence API
CDI: Contexts and Dependency Injection
REST: Representational State Transfer
JSF: JavaServer Faces
CSS: Cascading Style Sheets

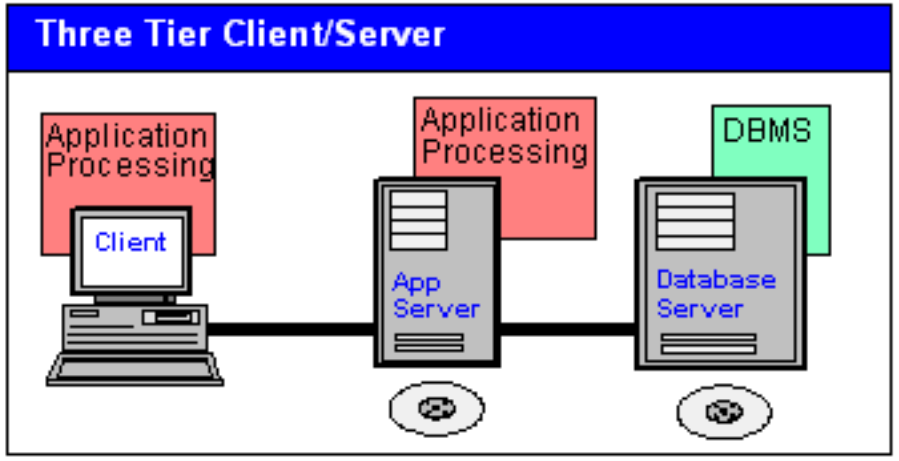
Architecture design – cont.

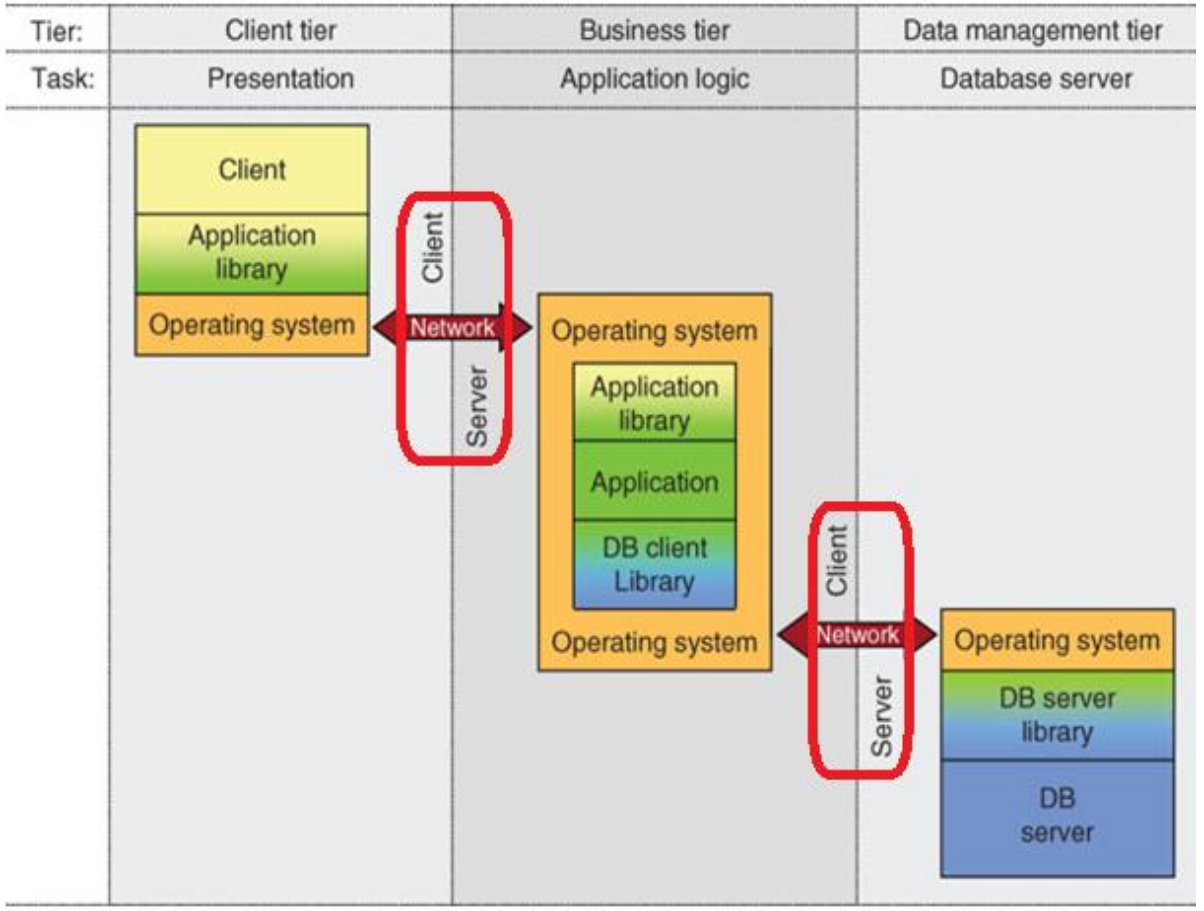
- Client/server architecture
  - Client/server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client.
  - Advantage:
    - Scalability and flexibility
  - Disadvantage:
    - Security



Architecture design – cont.

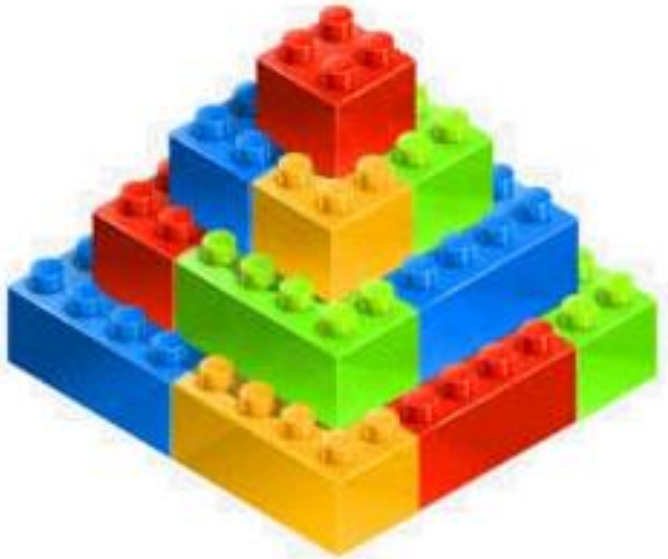
- Client/server architecture-cont.
- Client/server architectures have multiple tiers.
- For example: web app, DNS, …





Architecture design – cont.

- Component-based architecture
  - A component is a software object, intended to interact with other components, encapsulating certain functionality or a set of functionalities.
  - Component-based architecture focuses on the decomposition of the design into individual functional or logical components that represent well-defined communication interfaces containing methods, events, and properties. It provides a higher level of abstraction and divides the problem into sub-problems, each associated with component partitions.
- Decomposition



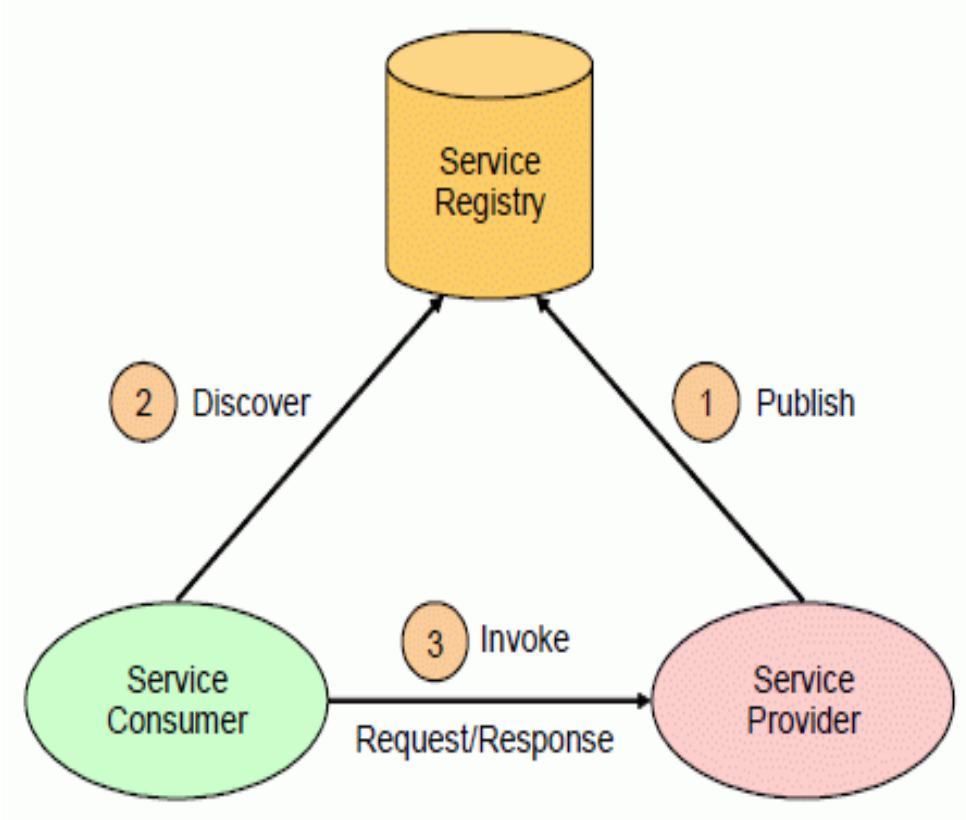
Architecture design – cont.

- Component-based architecture –cont.
  - A component is viewed as a functional element or a module of a software product.
- Example:
  - ATM:

  - Advantage:
    - Easier for understanding, shorten production time
  - Disadvantage:
    - Lack flexibility and scalability

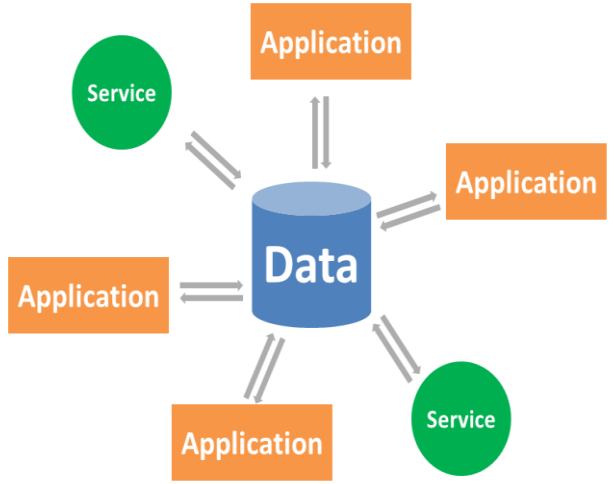
Architecture design – cont.

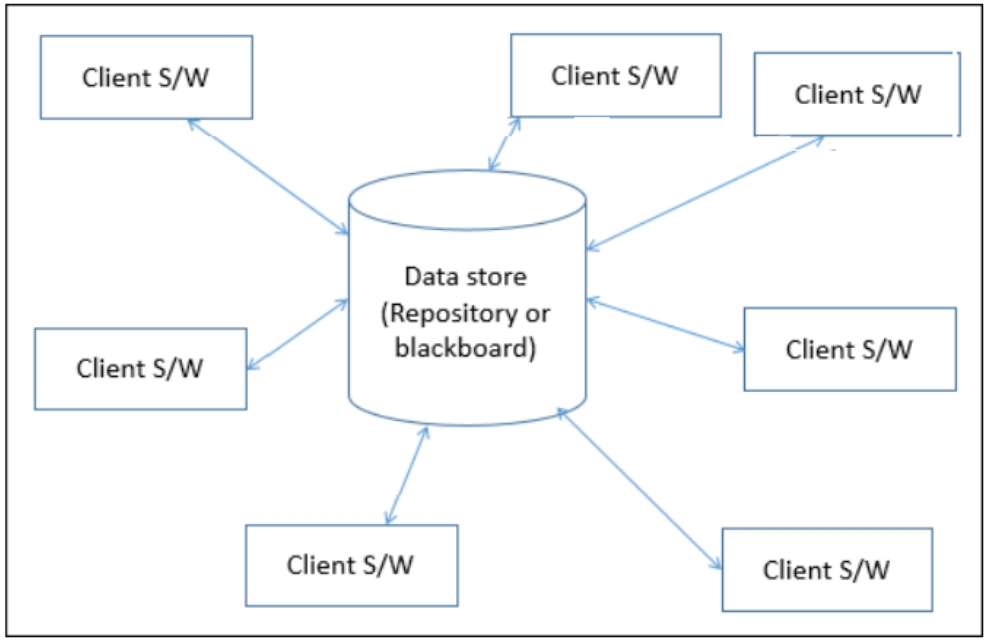
- Service-oriented architecture
  - Service-oriented architecture is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.
  - Advantage:
    - Services are easily maintained; reliability; availability; scalability
  - Disadvantage:
    - Increased overhead. E.g. customer needs
- Examples:
  - Cloud computing: Software as a service; platform as a service
  - Symmetric key management systems



Architecture design – cont.

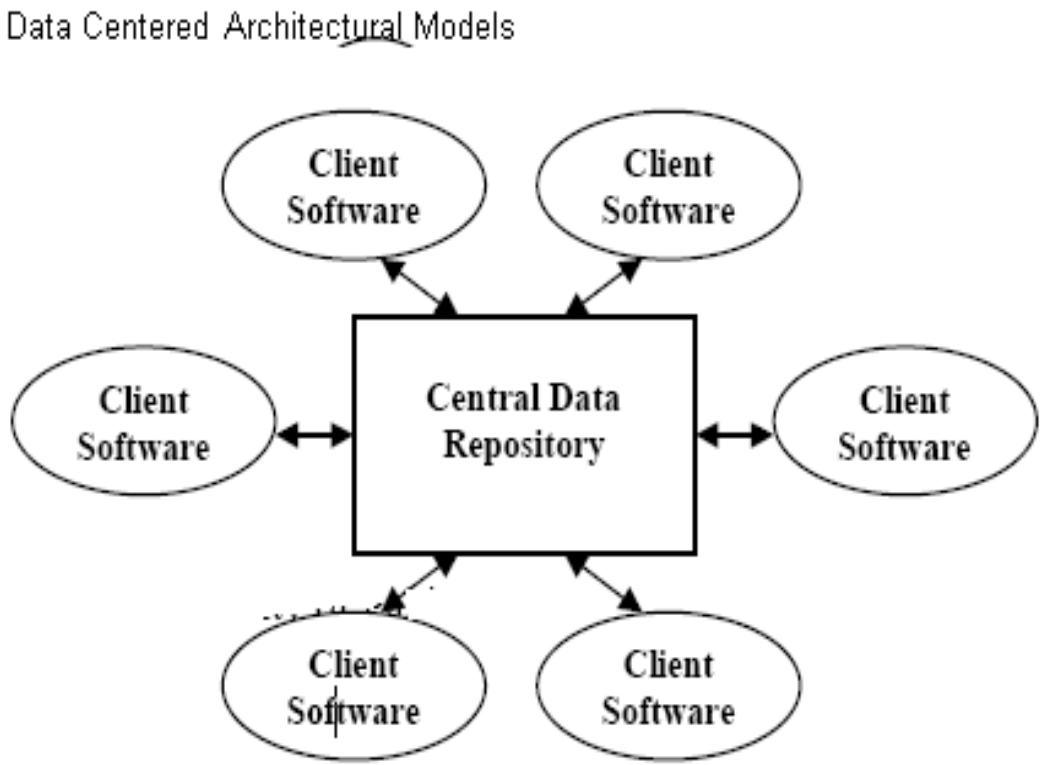
- Data-centric architecture
  - Data centric architecture is a software design model where data is the primary and permanent asset, and applications come and go.
  - Database plays an important role
  - Advantage:
    - Simplicity; security; integrity management
  - Disadvantage:
    - performance
- Example: Data Warehouse
- Question?
  - Data-centric architecture vs. client/server

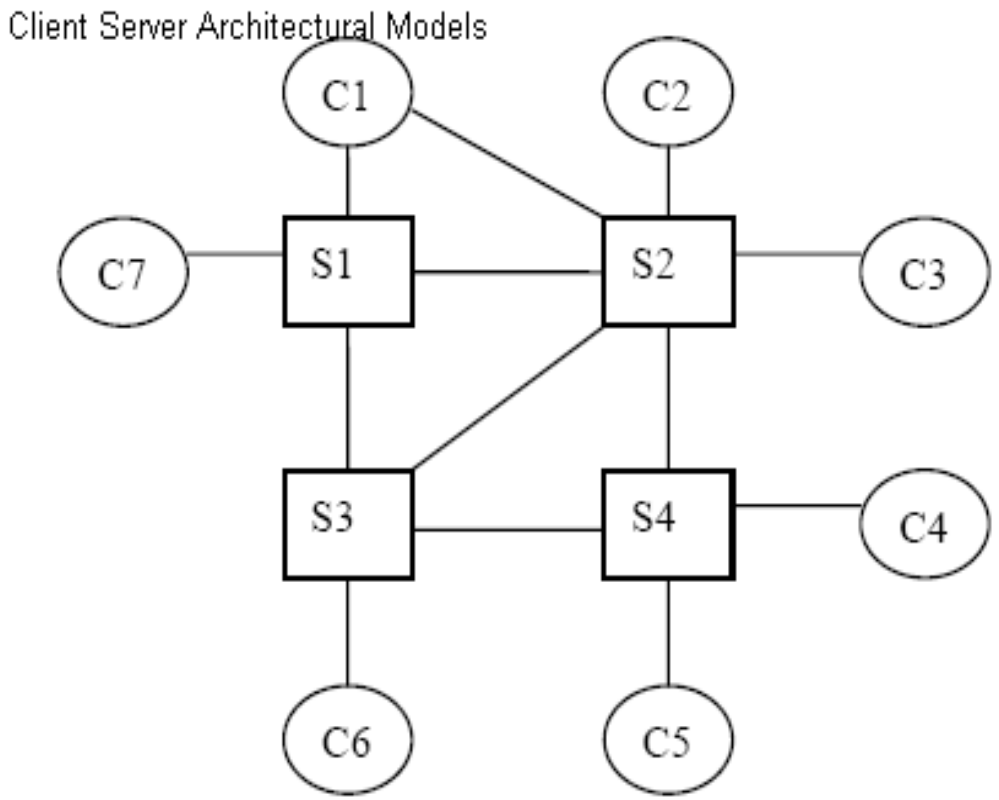




Architecture design – cont.

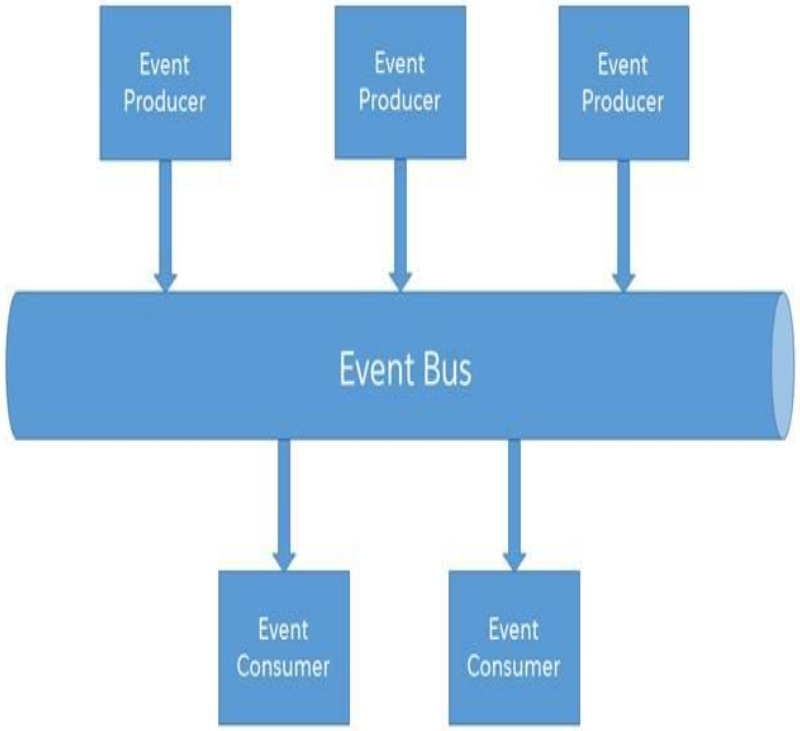
- Data-centric architecture vs. Client/server architecture





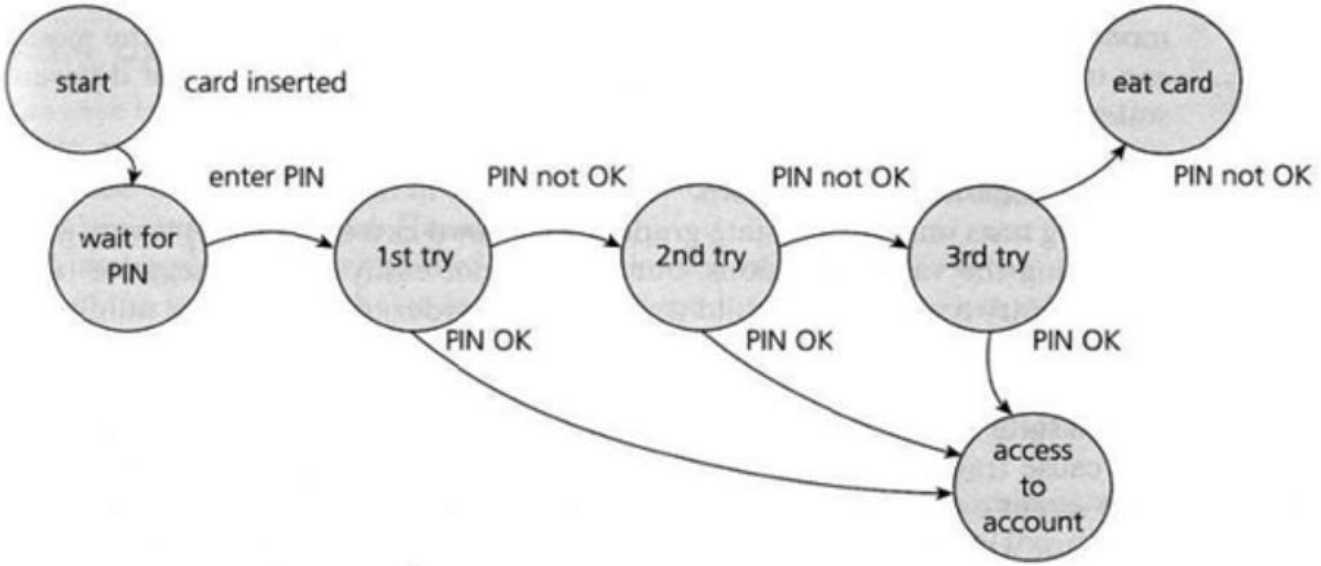
Architecture design – cont.

- Event-driven architecture (EDA)
  - Event: an occurrence that may trigger a state transition.
  - EDA is a software architecture pattern that the behavior of the software system based on events.
  - Two types of events
    - Hardware vs. software
  - Advantage:
    - Good performance
  - Disadvantage:
    - Complexity (cause race condition)



Architecture design – cont.

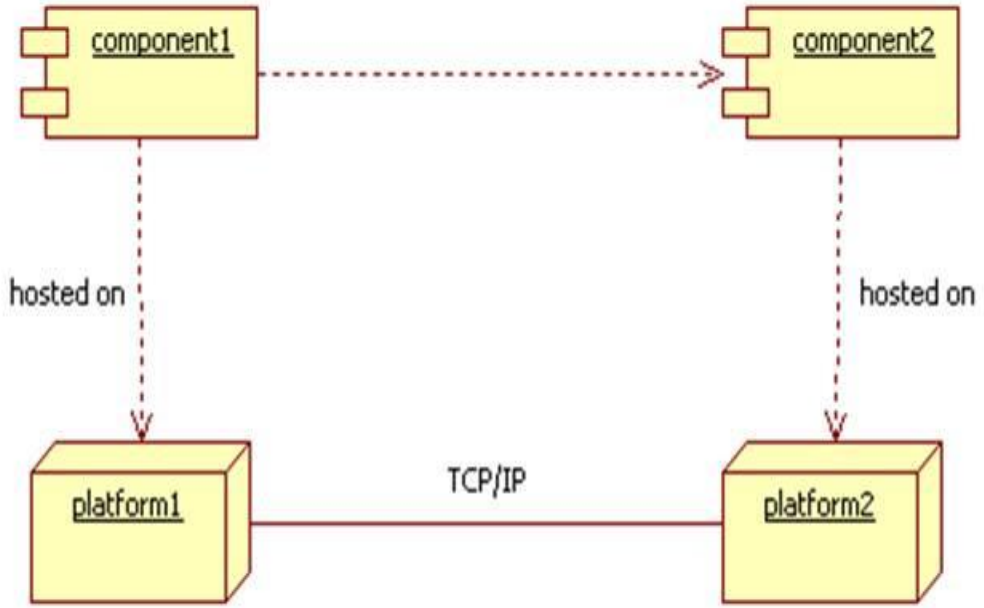
- Event-driven architecture (EDA) example – State transition diagram



	Event 1 (Insert card)	Event 2 (Enter Pin)	Event 3 (Pin OK)	Event 4 (Pin not OK)
S1:Start	S2	-	-	-
S2:Wait for Pin	-	S3	-	-
S3: 1st try	-	-	S6	S4
S4: 2nd Try	-	-	S6	S5
S5: 3rd Try	-	-	S6	S7
S6: access to account	-	-	-	-
S7: eat card	S1	-	-	-

Architecture design – cont.

- Distributed architecture
  - In distributed architecture, components are presented on different platforms and several components can cooperate with one another over a communication network in order to achieve a specific objective or goal.
  - Advantage:
    - Resource sharing; scalability
  - Disadvantage:
    - Complexity; security
- Examples:
  - MongoDB – non-SQL DB
  - Block Chain



Think – Pair – Share

- Architecture design
  - Monolithic
  - Client/server
  - Component-based
  - Service-oriented
  - Data-centric
  - Event-driven
  - Distributed
- For the group project, which architecture design is more suitable?

Summary

- Software Security
  - Security should be considered at system design phase
- User interface
  - Concept
- Architecture design
  - Monolithic
  - Client/server
  - Component-based
  - Service-oriented
  - Data-centric
  - Event-driven
  - Distributed

Announcement

- 25% of you have found a group for the group project
- Please let me know if you need help.