SOFTWARE ENGINEERING CS 487
Participation-2

Name : Satyam Rajput (A20537375)
Email Id : srajput3@hawk.iit.edu

Participation-2

1) Design patterns are “common solutions” to “common problems”. Discuss the importance of the word “common” with respect to achieving software engineering goals.

Considering the **Design Pattern fundamentals and concepts** that are Adopted from the (building) architecture community

Common solutions to common problems
"Why do things all over again?"

- **Proven** and **tested advantages** are similar to those of **component reuse**.
- There are **significant upfront time savings.**
- The **potential benefit** of avoiding the fix-retest cycle is considerably bigger.

Example: **Observer Pattern** - Separate the display of an object’s state from the object itself.

Considering the factors concerning achieving software engineering goals some factors play a huge role in getting to a common solution to a common problem :

- **Reuse**

  Abstract reuse through design patterns, object-oriented design, and development.

  Reusable components and systems, including tailored COTS (Commercial Off-The-Shelf) solutions.

- **Configuration Management**

  Includes version control, system build management, and issue management

- **Requirements Satisfaction**

  • Non-functional system requirements are largely met through architectural design

  – Performance optimisation

  – Security

  – Safety

  – Availability

  – Maintainability

  • Trade-offs are likely necessary due to conflicting priorities and overlaps in design elements.

- **Design Decisions**
- **Architectural Views**

2) Discuss the role of communication protocols and state-transition diagrams in fulfilling functional requirements.

   Communication protocols play a crucial role in **fulfilling functional requirements** by **enabling the exchange of data and information between different system components**. Here's how they contribute to meeting functional requirements:

   Interaction Models

   In software engineering, interaction models are used to explain and **illustrate** how various system **components communicate** with one another and with other such users. They are necessary for understanding how actions and information move through a system.

   • Types of interactions to be modelled
   – **User inputs and outputs**
   – **Cooperating systems**
   – **System components**

• Use case modeling
– Each represents a **discrete external interaction**
– Very **high-level** system's functionalities
• Sequence diagrams
– Captures a **sequence of interactions**
– The **sequential flow** of a given use case

Example: **Use Case Diagram**, **Sequence Diagram**

Role of State-Transition Diagrams in Fulfilling Functional Requirements :

State-transition diagrams are used to **model the behaviour of a system or component in response to external events**, providing a visual representation of how the system transitions from one state to another.

Capture the system components and their inter-relationships

– **Class diagrams**

• OO classes and how they are associated

• Model the “real world”

– **Generalization**

• Use more general terms to avoid getting “caught” in the details

• Members have common characteristics

– **Aggregation**

• Building components from sub-components

Data-flow models: The way information moves through an organisation and how it is processed, consumed, and disseminated typically drives work.

• From what source is the information derived?

• Who is the owner?

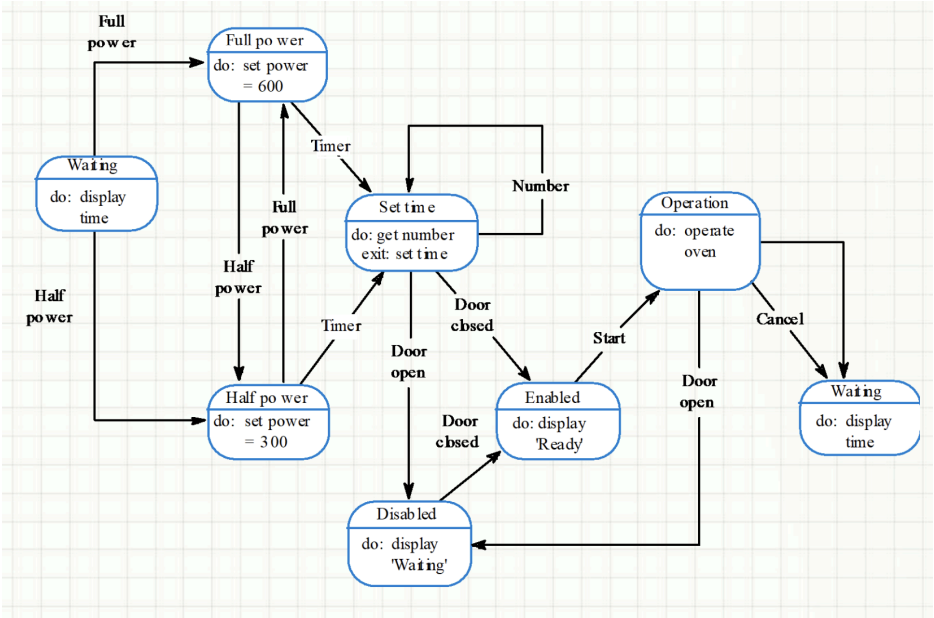
• How is it handled?, and so forth.

• Event-driven (state machine) models: In response to external stimuli, organisations transition between states.

• How will the system respond to various situations?

• What or who will initiate each event? and so forth.

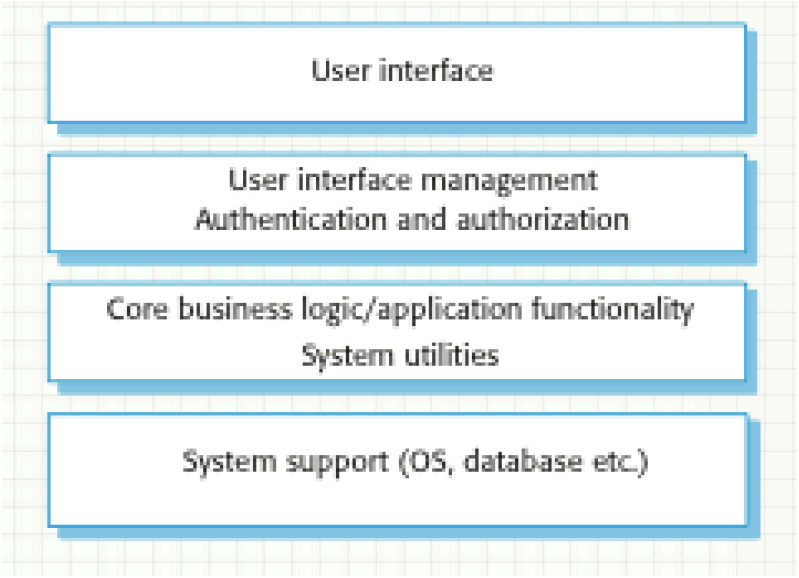
Example: **State Machine for a Microwave**



3) Discuss the role of security architecture and user experience design in implementing user categorization.

Security architecture makes sure that sensitive data and resource access are properly managed and regulated, it is essential to the implementation of user categorization.
Access Control
Authentication and Authorization
Data Protection

Considering the **Architectural Views** which helps Multiple perspectives to bring complexities into focus and **Architectural Patterns** like
Layered architecture
– Achieve separation and independence through layering
– Hierarchical organization
– Supports incremental development
Access Control
Authentication and Authorization

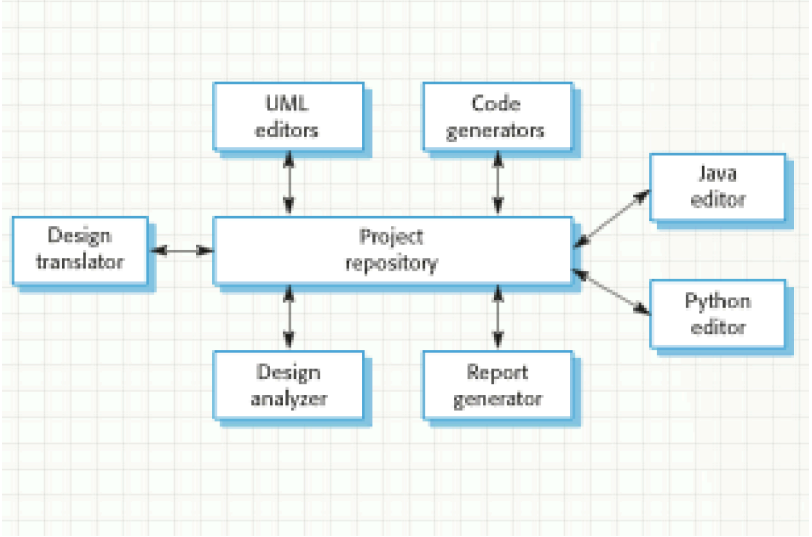


Repository architecture

– Support the exchange of information between sub-systems

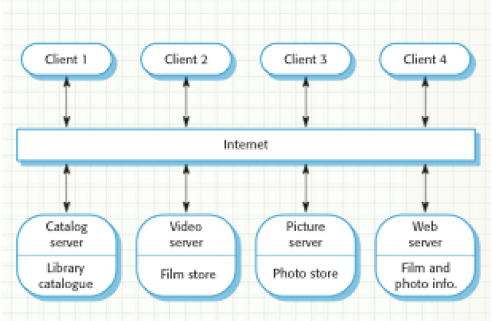
– Use a central repository to manage shared data

– Establish and maintain a separate database for each subsystem



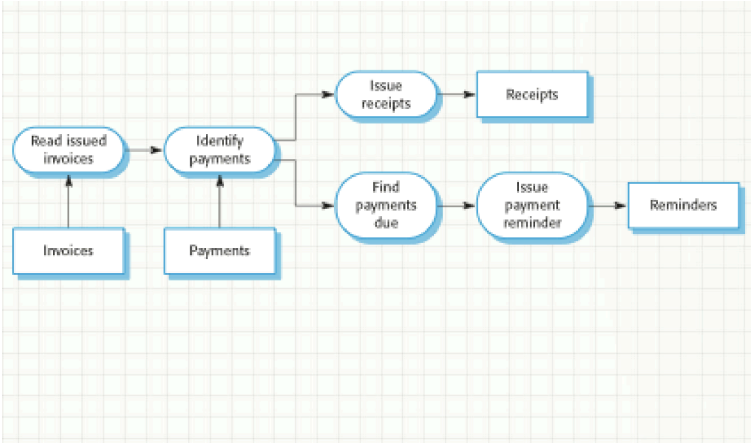
Client-server architecture

– Organized as a set of services and associated servers, accessed by clients “calling” the services



Pipe-and-Filter architecture

– Workflow
– Information is transformed as it flows through the system



Role of User Experience Design in Implementing User Categorization

In order to effectively integrate user categorization, user experience design must make sure that the process is visible, easy to use, and compatible with the requirements and expectations of various user groups.

User-Friendly Interfaces
Personalization
Clarity and Transparency
Accessibility

4) Discuss the role of context modeling and giving an automated system the ability to maintain awareness.

Context modelling is the process of **gathering** and **displaying** relevant **contextual data** that may have an impact on an **automated system's interactions and behaviour**. In order for the system to adjust and **react** to various circumstances, contextual awareness is essential.

According to **Modular Decomposition Styles**
Deciding how best to decompose sub-systems
down to modules
• Object-oriented decomposition
– Loosely coupled objects with well-defined interfaces

– Classes are templates with attributes and operations
– During execution, objects are instantiated from classes
• Function-oriented pipelining
– Data flow
– Inputs are processed by transformational functions to produce outputs
• The “real” world looks more like objects
– Therefore, OO works best for reuse
• However, work looks more like functions
– Therefore, FO provides the best immediate fit

Control Styles
• Deciding how best to control modules in operation
• Centralized control
– Design a sub-system whose primary function is to control the other sub-systems
– The vast majority of control is handled by this subsystem
• Event-based control
– Design each sub-system to “react” to events
– Events can come from other sub-systems or the environment
• Centralizing provides a single-point of design, implementation, etc. system focus
• De-centralizing will often be a more logical fit for modelling the “real” world

5) Compare the layout and flow of a retail store to a software system’s architecture.

Software System’s Architecture

A software system's architecture refers to the high-level design and structure of the system, encompassing components, their interactions, and the principles guiding their organization.

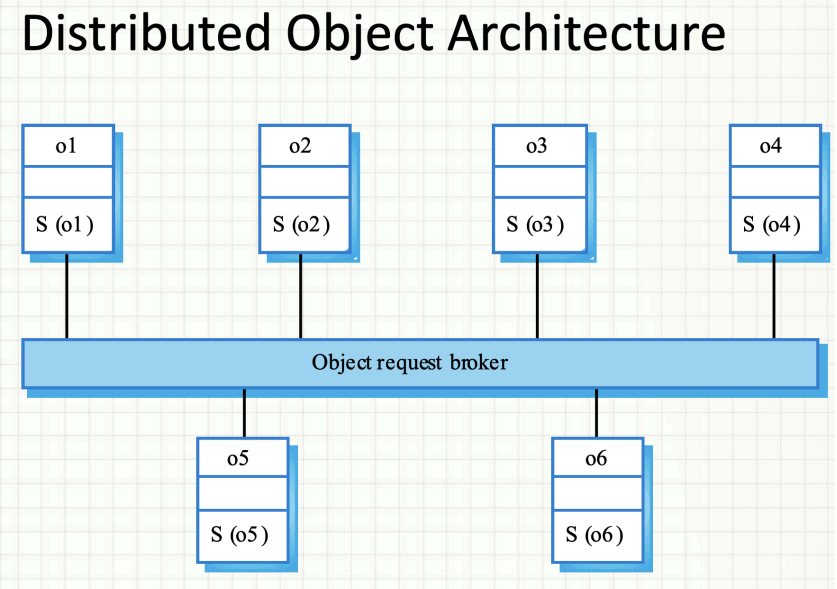
Component Organization
User Flow
User Interface Design
Data Management and Processing

Integration Points

Distributed Object Architectures

• Less restrictive than client-server in that all objects can offer services to all other objects

• Middleware, known as an object request broker, allow distributed objects to communicate across networked computers



Master-slave architecture

– Real-time systems requiring guaranteed response times

• 2-tier client-server architecture

– Centralized systems for security reasons

• Multi-tier C/S architecture

– To support high-volume transaction processing

• Distributed component architecture

– Supports combining resources from different systems

• Peer-to-peer architecture

– Servers “introduce” peers who then work together locally

Retail Store Architecture

The layout and flow of a retail store are designed to optimize the shopping experience for customers and the operational efficiency of the store.

Physical Classification: A retail establishment is often divided into discrete sections or divisions, each oriented to a certain category of

goods. This division improves the store's organisation and makes products easier for consumers to find.

Customer Flow: The layout of the store is intended to bring consumers along an established path, frequently passing by high-margin or highlighted items. The goal of this flow is to promote exploration and increase the possibility of making purchases.

Operational Zones: Behind the scenes, the store layout includes operational zones such as stockrooms, staff areas, and administrative offices, ensuring smooth functioning of the store.

Checkout Procedure: To ensure client comfort and speedy transaction processing, the checkout area is positioned strategically. For convenience, it is frequently positioned in the middle of the area or close to the store's entrance.