UNIT-3
SOFTWARE DESIGN

Structured Analysis/Structured Design (SA/SD)

1. Analysis and Design model -

   DDOD  ERD  DFD  PSPEC
   STD  CSPEC

   ANALYSIS MODEL

2. Design Process - Sequence of steps carried through which the
requirements are translated into a system or software model.

   Requirements Specification
   Architecture Design
   Software Specification
   Interface Specification
   Component Specification
   Data Structure Specification
   Algorithm Specification

   Design Products

3. Design Principles -
   (1) Design process should not suffer from “tunnel vision.”
   (2) Design should be traceable to the analysis model.
   (3) Design should not reinvent the wheel.
   (4) Design should “minimize the intellectual distance” between the
       software and the problem in the real world.
   (5) Design should exhibit uniformity and integration.
   (6) Design should be structured to accommodate change.
   (7) Design should be structured to degrade gently.
   (8) Design is not coding and coding is not design.
   (9) Design should be assessed for quality.
   (10) Design should be reviewed to minimize conceptual errors.
5. **Design Concepts** - Following issues are considered while designing the software:
   1. Abstraction
   2. Modularity
   3. Refinement

5. **Modular Design** - Use of modules

Various quality parameters for effective modular design are:
1. **Functional Independence** - Functional modules with single-minded effort.
2. **Cohesion** - Information hiding can be done. Perform only one thing.
3. **Coupling** - How modules are connected to each other outside use (e.g.,

   → Various types of Cohesion:
   i. Coincidently cohesive - set of tasks related to each other.
   ii. Logically cohesive - logically related to each other.
   iii. Temporal Cohesion - tasks executed in some specific time span.
   iv. Procedural Cohesion - specific order execution of processing elements.
   v. Communication Cohesion - sharing of data between processing elements.

   → Various types of Coupling:
   i. Data Coupling - done by parameter passing or data interaction.
   ii. Control Coupling - share related control data.
   iii. Common Coupling - common data or global data.
   iv. Content Coupling - use content of another module.

   → Fan-out - Number of immediate subordinates to a module.
   → Fan-in - How many modules directly control a given module.
   → Factoring - separation of function into new modules.

6. **Software Modelling** - In software architecture, the software model is designed and structured. The model is partitioned horizontally or vertically:

   1. **Structural Model** - Organizes collection of components.
   2. **Framework Model** - Reusable architectural design frameworks.
(3) Dynamic model - behaviour aspect of the system's environment
(4) Process model - business or technical process
(5) Functional model - functional hierarchy of the system:

```
          Decision maker
             /     /
           /     /
          /     /
         /     /
        /     /
       /     /
      /     /
     /     /
    /     /
   /     /
  /     /
```

Horizontal Partitioning

```
          Worker
             /     /
           /     /
          /     /
         /     /
        /     /
       /     /
      /     /
     /     /
    /     /
   /     /
```

Vertical Partitioning

(7) Function oriented design -
    Design of the software system is divided into various self-interacting units called functions.
    Various models that can be created in function oriented design are:
    (1) DFD, (2) Data Dictionary, (3) Pseudocode, (4) Structured chart/design.

    Structured charts/design -
    Basic element is module, consists of four attributes:
    Input and output, function, mechanics and internal data.

    Module = (Input, Output) + (Function, Mechanics) *
    Flag = Flag
    (Q, P) → (Q, P)

    Pseudocode -
    It is combination of algorithms written in simple language and programming language statements.

    Flowchart - Graphical representation of an algorithm.

    I/O symbol = Input/Output
    Process = (Box)
    Flow of information = (Arrow)

(8) Object-oriented design -
    It includes classes, objects, encapsulation, information hiding, inheritance, and polymorphism.

    Advantages - very flexible, easier to maintain and adapt.
    Disadvantages - very slow, require more memory.
Unified Modeling Language (UML) -

1. Use Case View -
   - Use Case Diagram
     - Describes an interaction between a user and a system.
     - Actor - An entity which interacts with the system.
     - Use Case - Represents the behavior of the system.

2. Structural View -
   - Class Diagram -
     - Used to capture the static view of the system.
     - **OBJECT NAME**
     - DATA MEMBERS
     - AND ATTRIBUTES
     - OPERATIONS OR
     - METHODS

3. Behavioral View -
   - Collaboration Diagram -
   - Sequence Diagram -
     - Sequence of events in which the object interacts with the other object is shown.

**Use** = ![Image]  
**Active Object** = ![Image]  
**Interaction** = ![Image]  
**Destruction** = ![Image]

- Activity Diagram -
  - Graphical representation for representing the flow of interaction within specific scenarios.
  - Join & Fork (Represents many activities) = ![Image]  
  - Branch & Merge = ![Image]  

- State Transition Diagram - (Dynamic models) -
  - Graphically represent the state of the system.
  - ![Initial State] - Initial State  
  - ![Final State] - Final State  
  - ![Simple State] - Simple State  
  - ![Compound State] - Compound State
(2) Data flow Architecture

(3) Call and Return Architecture

(4) Object oriented architecture

(5) Layered Architecture

(6) Architectural Views

(7) User Interface Design
- User Interface design principles:
  - Familiarity
  - Consistency
  - Irreversibility
  - Recoverability
  - User Guidance

- Three rules for ideal interface design (golden rules):
  1. Place the user in control.
  2. Reduce the user's memory load.
  3. Make the interface consistent.

- Design Metrics:

  - Architectural Design Metrics:
    1. Metrics by Cost and Glass:
      - Structural complexity:
        \[ S(k) = \frac{\text{fan out}}{\text{fan out} + \text{fan in}} \]
      - Data complexity:
        \[ D(k) = \frac{\text{tot var}}{\text{fan out} + 1} \]
      - System complexity:
        \[ S_y(k) = S(k) + D(k) \]

    2. Metrics by Fenton:
      - Page: \( n + c \) (node + edge)
      - Depth: longest path from root to leaf node
      - Width: maximum number of nodes at particular level
      - Edge to node ratio, \( r = e/n \)

    3. Metrics by using Design structure quality index (DSQI):
      - Value of indices range from 0 to 1 (F1 - F7)

- Object-oriented Design metrics:
  1. CK metrics suite:
     - Simplicity of design metrics
     - Weight methods for class (WMC), Depth of inheritance tree (DIT)
     - Number of children (NOC), Coupling between object classes (CBO)
     - Response for a class (RFC), level of cohesion in methods (LCOM)
(2) MOOD Metrics suite -
  - Two metrics: Method inheritance factor (MIF), Coupling factor (CF).
(3) horning and khid oo metrics -
  - Class metrics, Inheritance metrics, Class internals, Encapsulation.
  - Component level design metrics - Three measures/metrics are -
    Cohesion, Coupling, and complexity.
  - User interface design metrics -
    - Layout appropriateness -
      \[ \text{Cost} = \frac{1}{2} \sum_{i=1}^{n} \text{frequency of transitions} \times \text{Cost of transition} \]

(4) Cohesion Metrics -
  - Weak function cohesion = \( \frac{\text{Number of glue tokens}}{\text{Total no. of data tokens}} \)
  - Strong function cohesion = \( \frac{\text{Number of glue tokens}}{\text{Total no. of data tokens}} \)

\[ \text{Cost} = \frac{1}{2} \sum_{i=1}^{n} \text{frequency of transitions} \times \text{Cost of transition} \]