UNIT-III
Design Engineering

Course Outcomes

- Understand software design principles
- Apply UML modeling techniques

Unit III Syllabus

- Design Process and Quality
- Design Concepts
- Design Model
- Software Architecture
- Data Design
- Architectural Styles and Patterns
- UML Diagrams

Introduction to Design Engineering

- Transforms requirements into a blueprint for software construction.

Design Process

- Requirements Analysis → Architectural Design → Interface Design → Component Design → Deployment

Objectives of Design

- Improve Quality
- Reduce Complexity
- Enhance Maintainability

Design Quality

- Attributes:
- • Correctness
- • Efficiency
- • Maintainability
- • Reusability

Design Concepts

- Abstraction
- Refinement
- Modularity
- Information Hiding
- Functional Independence

Abstraction

- Focuses on essential characteristics while ignoring unnecessary details.

Refinement

- Elaborates design details progressively.

Modularity

- Divides software into manageable modules.

Information Hiding

- Conceals internal details from other modules.

Functional Independence

- High Cohesion and Low Coupling.

The Design Model

- Data Design
- Architectural Design
- Interface Design
- Component-Level Design

Software Architecture

- Overall structure of software components and relationships.

Architectural Design Process

- System Structuring → Control Modeling → Modular Decomposition

Data Design

- Transforms information domain models into data structures.

Architectural Styles

- Layered Architecture
- Client-Server
- Repository
- MVC

Architectural Patterns

- Reusable solutions to common design problems.

Examples of Patterns

- Singleton
- Factory
- Observer
- MVC

Introduction to UML

- Unified Modeling Language for software visualization.

Conceptual Model of UML

- Building Blocks
- Rules
- Common Mechanisms

Basic Structural Modeling

- Represents static aspects of the system.

Class Diagrams

- Classes, Attributes, Operations, Relationships.

Class Diagram Example

- Student → Course → Faculty relationships.

Sequence Diagrams

- Show object interactions over time.

Collaboration Diagrams

- Depict object interactions and relationships.

Use Case Diagrams

- Represent system functionality from user's perspective.

Component Diagrams

- Show software components and dependencies.

Case Study: ATM System UML

- Use Case, Class and Sequence Diagrams for ATM operations.

Real-Time Example: E-Commerce Architecture

- Client-Server architecture with layered design.

Summary

- Review Design Engineering concepts and UML diagrams.