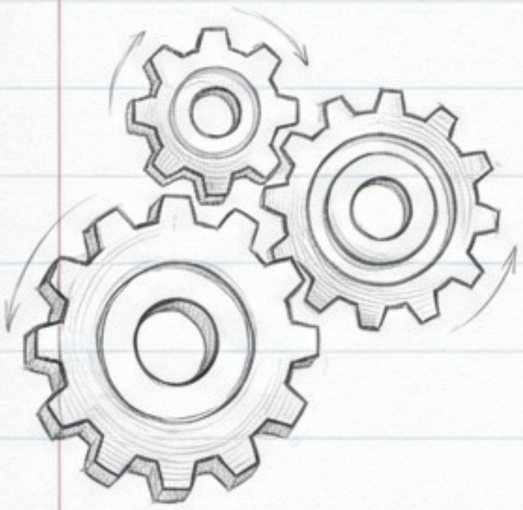
CSA 7105 T Software Engineering

Software Engineering - Detailed Notes (BCA)



Notes created by Kamal Kishor (HandNotes)

1. Introduction to Software Engineering

What is Software?

Software is a set of programs, procedures, and related documentation that tells a computer what to do.

Components of Software

- 1. Programs – Instructions written in programming languages
- 2. Data – Information used by programs
- 3. Documentation – User manuals, technical documents, help files

What is Software Engineering?

Software Engineering is the application of engineering principles to the design, development, testing, deployment, and maintenance of software.

IEEE Definition: Software Engineering is the application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software.

Why Need SE & Characteristics

Why Software Engineering is Needed?

- ✓ Software projects are large and complex
- ✓ Cost of failure is very high
- ✓ Need for quality, reliability, and maintainability
- ✓ Helps in project planning, risk management, and documentation

Software Characteristics

Software is:

- Developed, not manufactured
- Does not wear out
- Custom-built
- Highly complex
- Easy to copy but hard to design

The Evolving Role of Software

Earlier software	→	Modern software
Small		Web & mobile based
Standalone		Distributed systems
Less user interaction		AI-based systems
		Cloud applications
		Real-time systems

2. Software Process

A software process is a structured set of activities required to develop a software system.

Basic Process Activities:

1. Requirement analysis
2. Design
3. Coding
4. Testing
5. Maintenance

Software Process Framework

A generic framework that applies to all software projects.

Framework Activities:

1. Communication – Understanding requirements
2. Planning – Scheduling, estimation
3. Modeling – Design and analysis
4. Construction – Coding and testing
5. Deployment – Delivery and feedback

Umbrella Activities
(Applied throughout)

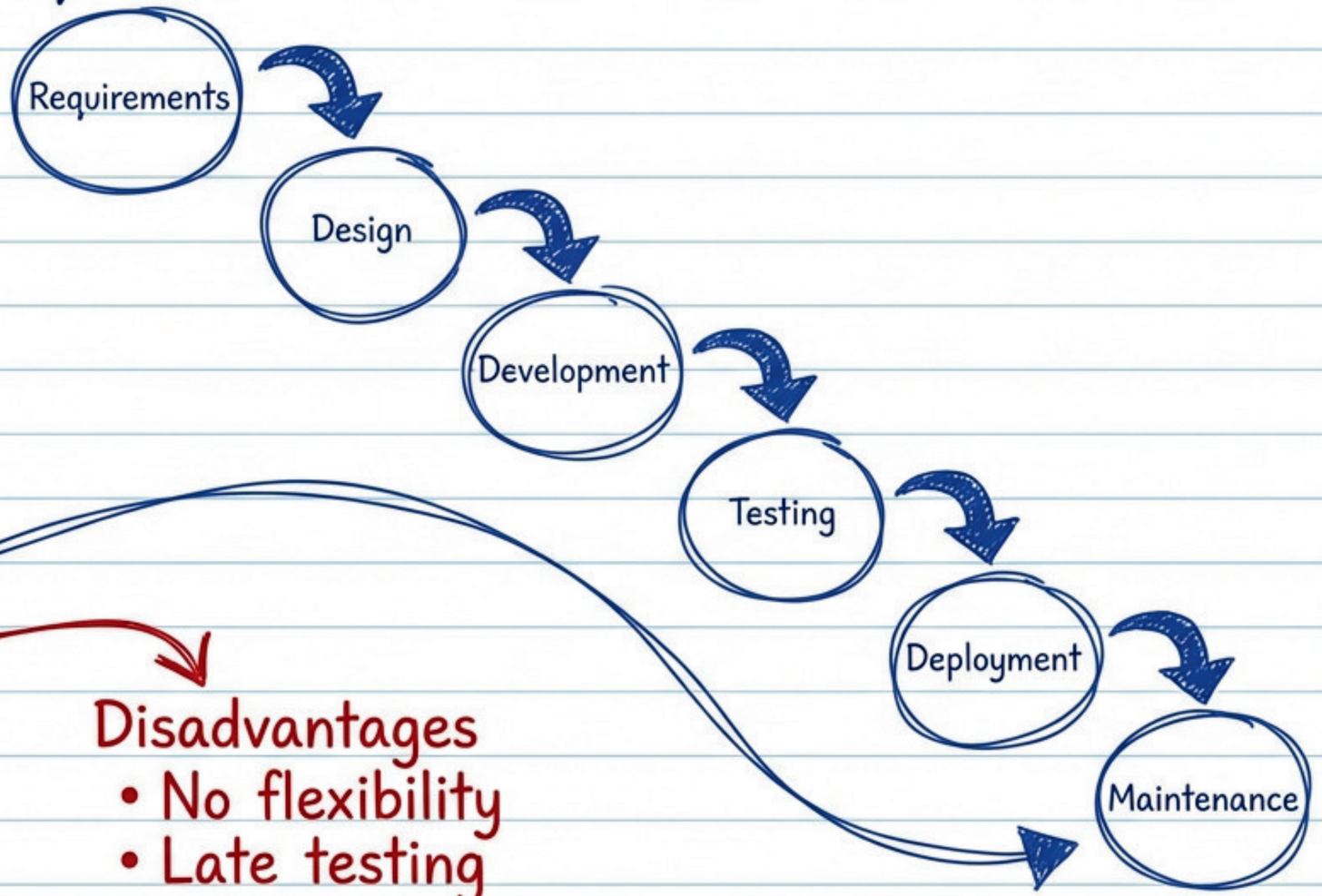
Software project tracking & control,
Risk management, SQA,
Technical reviews,
Documentation,
Configuration managem,
Reusability manag·gt,
Measurement.

3. Software Process Models: Waterfall Model

Definition: A linear and sequential model.

Phases:

1. Requirement Analysis
2. System Design
3. Implementation
4. Testing
5. Deployment
6. Maintenance



Pros & Cons

Advantages

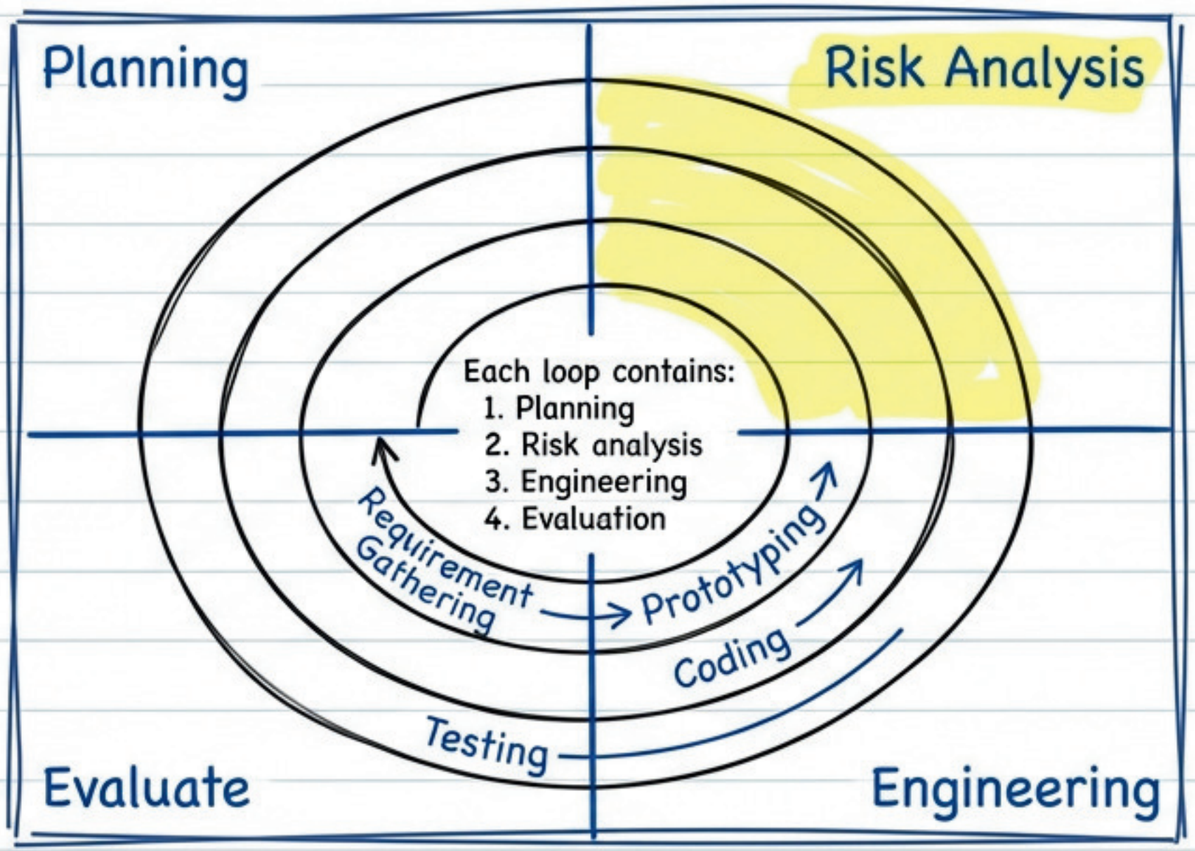
- Simple and easy
- Good for small projects

Disadvantages

- No flexibility
- Late testing
- Difficult to change requirements

The Spiral Model

Combines iterative development + risk analysis.



Pros & Cons

Advantages

- Risk handling
- Suitable for large projects

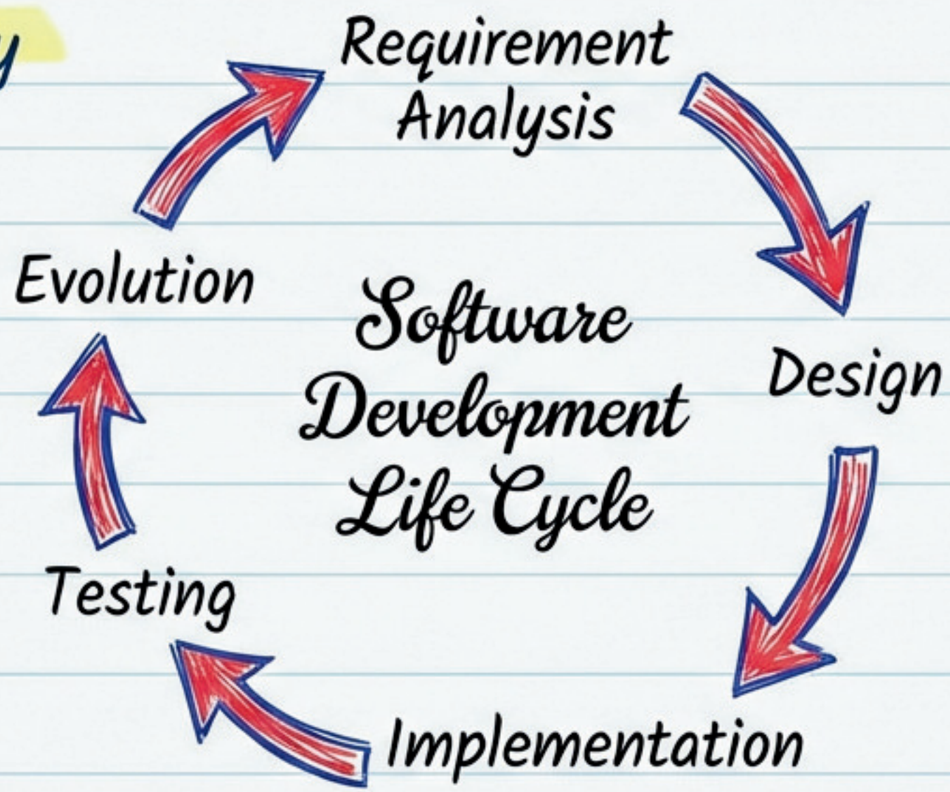
Disadvantages

- Complex
- Expensive

4. Software Development Life Cycle (SDLC)

SDLC defines the steps involved in software creation.

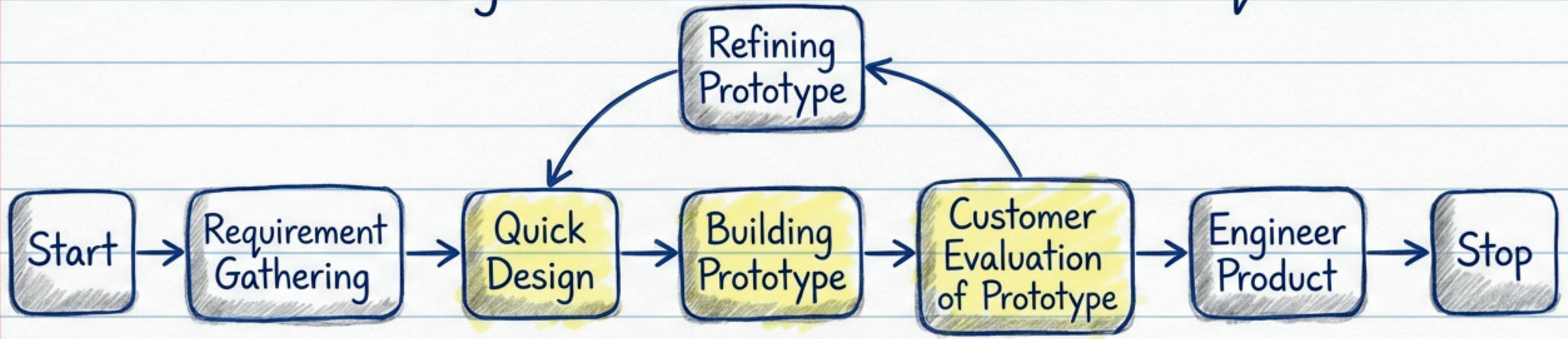
1. Feasibility Study
2. Requirement Analysis
3. Design
4. Coding



4. Coding
5. Testing
6. Deployment
7. Maintenance

Prototype Model

Definition: A working model is built to understand requirements.



Types:
Throwaway Prototype
Evolutionary Prototype

Advantages:
Better requirement clarity
User involvement

Disadvantages:
- Poor documentation
- Not scalable

Agile & Advanced Models

Agile Model

Focuses on iterative development and customer collaboration.

Principles:

- Working software over documentation
- Customer interaction
- Responding to change.

Popular Agile Methods:

- Scrum
- XP (Extreme Programming).

Extreme Programming (XP) Emphasizes:

- Pair programming
- Continuous testing
- Frequent releases
- Simple design

Team Software Process (TSP):

- Focuses on team-level planning
- Improves quality and productivity.

5. Requirement Analysis

Software Requirement: A condition or capability that the system must satisfy.

Functional Requirements
(What the system should do)

Non-Functional Requirements
(Performance, security, usability, reliability)

Requirement Engineering Process

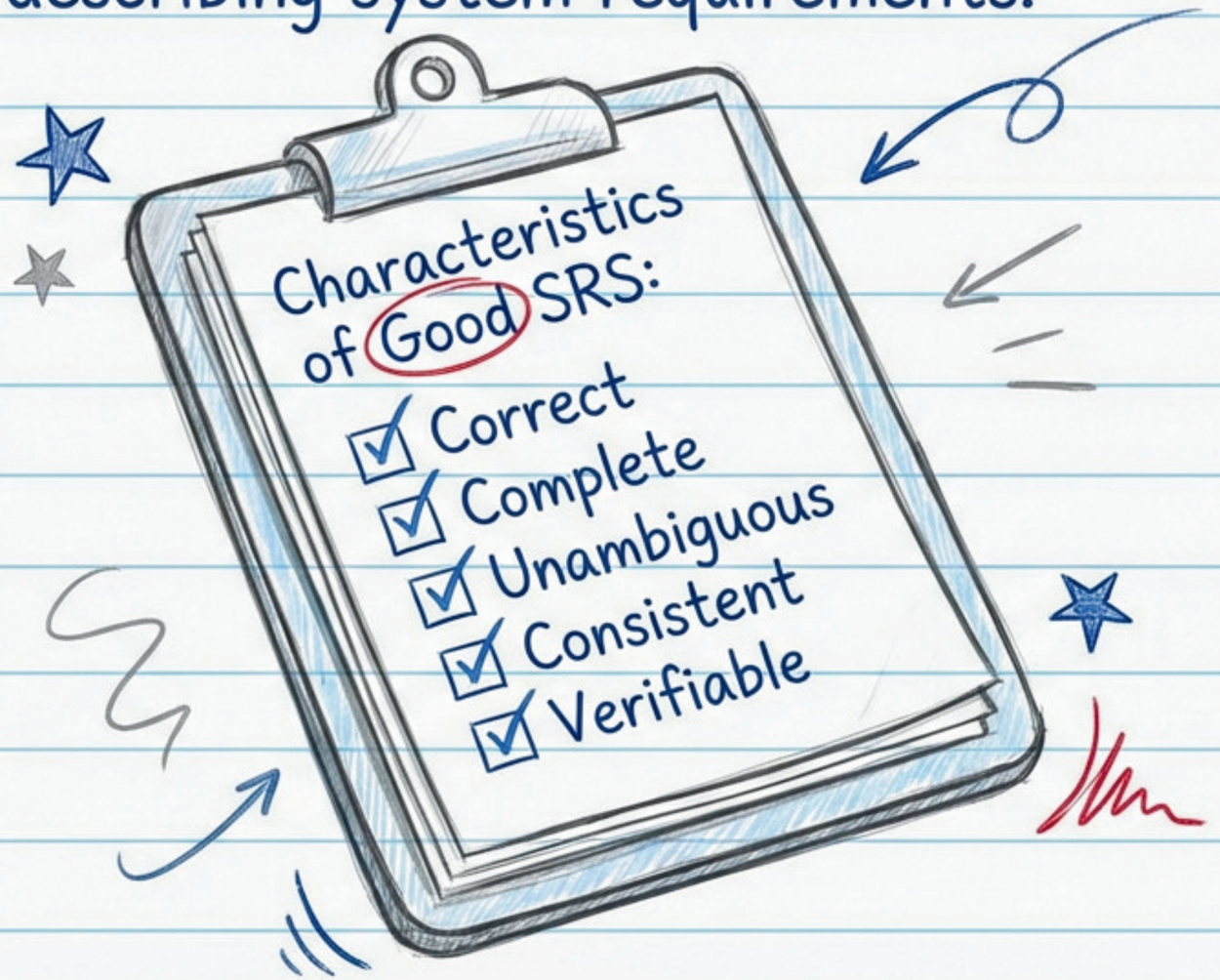
1. Requirement elicitation
2. Requirement analysis
3. Requirement specification
4. Requirement validation

Requirement Analysis Models

- ✓ **Flow-Oriented Modeling:** Uses Data Flow Diagrams (DFD). Shows how data moves.
- ✓ **Data Modeling:** Uses Entity Relationship Diagrams (ERD).
- ✓ **Behavioral Modeling:** Shows system behavior over time.

The SRS Document

Software Requirement Specification (SRS): A formal document describing system requirements.

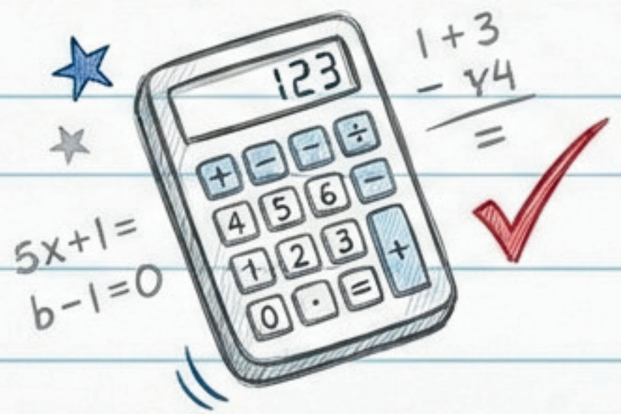


6. Software Project Management

Project Management: Planning, organizing, and controlling resources to achieve project goals.

Project Estimation

- Estimating: Cost, Effort, Time.
- Estimation Techniques: LOC (Lines of Code), Function Point Analysis.



Scheduling

- Defines: Task sequence, Deadlines, Resource allocation.
- Tools: Gantt Chart, PERT Chart.

Risk Management

Identifying and managing potential problems.

Types of Risks:

- Technical risk ✓
- Project risk ✓
- Business risk ✓

Risk Identification & Refinement

- List all risks
- Analyze probability and impact
- Prepare mitigation plan

7. Software Design

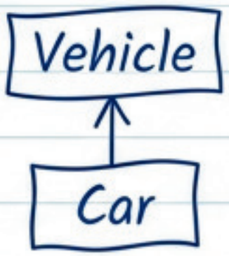
Transforming requirements into architecture and detailed design.

Identifying Objects & Classes

- Objects → real-world entities
- Classes → blueprint of objects

Relationships in Design

1. Generalization – Inheritance



2. Specialization – Specific class from general

3. Aggregation – HAS-A relationship

Design Diagrams

Use Case Diagram: Shows interaction between user (actor) and system.
Sequence Diagram: Shows message flow over time.
Class Diagram: Shows class structure, attributes, methods.

Principles, Tools & Testing

1. Design Tools

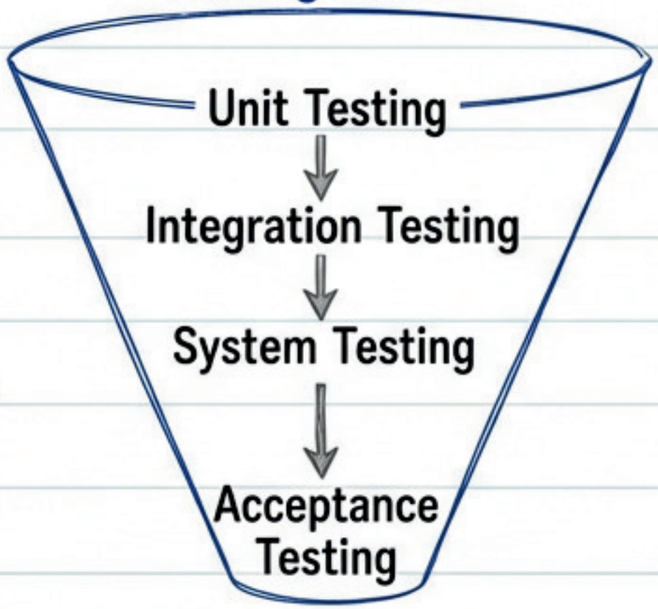
1. Data Flow Diagram (DFD)
2. Data Dictionary
3. Decision Table
4. Decision Tree

2. Modularization & Coupling

- **Modularization:** Dividing system into modules (Easy maintenance, Better testing). ✓
- **Coupling:** Degree of dependency.
  → Low coupling (good), High coupling (bad).

3. Software Testing

Testing Levels



Methods

- **Black Box** (Input/output based)
- **White Box** (Internal logic/code)
- **Basis Path Testing** (Cyclomatic Complexity)

Software Engineering provides a structured, disciplined approach to building reliable, efficient, and maintainable software systems. It is essential for managing complexity, reducing cost, and improving quality.

notes by ~ Kamal Kishor