Software Engineering

Short Notes based on ***<u>Exit Exam</u>***

Zemzem labs

1. Introduction to Software Engineering

Key Definitions

- **Software Engineering:** Systematic, disciplined process of developing, operating, and maintaining software.
- **Software Process:** Structured set of activities:
    1. Requirements
    2. Design
    3. Implementation
    4. Testing
    5. Deployment
    6. Maintenance

Software Characteristics

- Intangible
- Complex
- Evolves continuously
- Does not wear out but **deteriorates** due to changes (software aging)

2. Software Process Models

1. Waterfall Model

- Linear, sequential
- Easy to manage
- No going back once a phase is complete
- Suitable for stable requirements

2. V-Model

- Validation ↔ Verification
- Testing mapped to each development phase

3. Incremental Model

- Build small increments
- Early partial delivery
- More flexible than waterfall

4. Spiral Model

- Risk-driven
- Combines waterfall + prototyping
- Focus on identifying and reducing risks

5. Agile / Scrum

- Iterative & incremental
- Adaptive planning, quick delivery
- Daily stand-ups, sprints, product backlog
- Works best with changing requirements

3. Software Requirements Engineering

Types of Requirements

- **Functional Requirements (FR):** What system should do
- **Non-Functional Requirements (NFR):** Quality attributes (performance, security, reliability)

Requirement Engineering Steps

1. Elicitation
2. Analysis
3. Specification
4. Validation
5. Management (change control)

SRS (Software Requirement Specification)

Good SRS qualities:

- Correct, Complete
- Consistent
- Unambiguous
- Verifiable
- Traceable

4. Software Design

Levels of Design

- **High-Level Design (HLD):** Architecture, modules
- **Low-Level Design (LLD):** Algorithms, data structures

Architectural Styles

- Layered architecture
- Client-server
- MVC
- Pipe-and-filter
- Microservices (modern)

Design Principles

- Abstraction
- Modularity
- Encapsulation
- Separation of Concerns
- Low Coupling & High Cohesion

UML Diagrams

- **Structural:** Class, Component, Deployment
- **Behavioral:** Use case, Sequence, Activity, State diagram

5. Software Testing

Testing Levels

- **Unit Testing** → individual functions
- **Integration Testing** → combined modules
- **System Testing** → full system
- **Acceptance Testing** → user validation

Testing Types

- Black-box testing → based on inputs/outputs
- White-box testing → internal logic
- Grey-box → mix of both

Common Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Control flow testing
- Path testing
- Regression Testing
- Smoke & Sanity tests

6. Software Quality Assurance (SQA)

Activities

- Reviews
- Walkthroughs
- Inspections
- Audits
- Process monitoring

Standards

- ISO 9001
- CMMI levels (1–5)
    - Level 1: Initial
    - Level 2: Repeatable
    - Level 3: Defined
    - Level 4: Managed
    - Level 5: Optimizing

7. Project Management

Key Responsibilities

- Planning
- Scheduling
- Tracking
- Risk management
- Configuration management

Effort Estimation

- **LOC-based estimation**
- **Function Point Analysis (FPA)**
- **COCOMO (Basic/Intermediate/Detailed)**

Scheduling Tools

- Gantt chart
- PERT / CPM
- Work Breakdown Structure (WBS)

Risks

- Technical risk
- Schedule risk
- Cost risk
- Operational risk

8. Software Configuration Management (SCM)

Purpose

- Control changes to software work products
- Ensure consistency & traceability

Key Elements

- Version control (Git)
- Build management
- Release management

Change Management

- Change request
- Impact analysis
- Approval
- Implementation

9. Maintenance

Types

- **Corrective** – fix defects
- **Adaptive** – support new environment
- **Perfective** – improve performance/features
- **Preventive** – reduce future problems

Software Evolution (Lehman's Laws)

Software continually evolves or becomes obsolete.

10. Risk Management in SE

Process

1. Risk identification
2. Risk analysis
3. Risk prioritization
4. Risk mitigation
5. Risk monitoring

Risk Categories

- Requirement risks
- Technology risks
- People risks
- Security risks
- Project risks

11. Software Metrics

Used to measure software quality, complexity, size, productivity.

Product Metrics

- LOC
- Cyclomatic complexity
- Defect density

Process Metrics

- Productivity
- Schedule variance

- Cost performance index

Metrics Properties

- Simple, Objective, Useful, Valid, Reliable

⭐ Ultra-Condensed Summary (Exam Revision)

- **SE process:** Requirements → Design → Coding → Testing → Deployment → Maintenance
- **Models:** Waterfall, V-Model, Incremental, Spiral, Agile
- **SRS:** FR + NFR, must be complete, consistent, unambiguous
- **Design:** High cohesion, low coupling, modularity
- **Testing:** Unit → Integration → System → Acceptance
- **Estimation:** Function Point, COCOMO
- **Maintenance:** Corrective, Adaptive, Perfective, Preventive
- **Quality:** Reviews, Inspections, CMMI
- **SCM:** Version control + change control