Comprehensive Summary of Software Engineering Concepts

1. Software Requirements Engineering

- **Definition: Software requirements describe system functionalities and constraints.**
- **Types:**
  - **Functional Requirements: Define what the system should do (e.g., user login, order placement).**
  - **Non-Functional Requirements: Define system constraints (e.g., performance, security, usability).**
- **Classification:**
  - **Product Requirements: Behavior-related constraints like execution speed.**
  - **Organizational Requirements: Policy-driven, such as compliance with standards.**
  - **External Requirements: Factors like legal and interoperability constraints.**

2. Requirements Engineering Process

- **Elicitation Techniques:**
  - **Interviews: Structured (closed) or exploratory (open).**
  - **Scenarios & Use Cases: Real-world examples of system behavior.**
  - **Prototypes: Working models to refine requirements.**
  - **Facilitated Meetings: Group discussions to refine software needs.**
  - **Observation (Ethnography): Studying real-world usage patterns.**
  - **User Stories: Agile technique defining needs from a user perspective.**
- **Validation Factors:**
  - **Correctness, Consistency, Completeness, Realism, Verifiability.**
- **Analysis Techniques:**
  - **Classification, Conceptual Modeling, Requirements Negotiation.**
- **Software Requirement Specification (SRS):**
  - **A formal document detailing functional and non-functional requirements.**

3. Agile Software Development

- **Motivation: Developed for adaptability in fast-changing environments.**
- **Agile Manifesto Values:**
  - **Individuals & interactions over processes & tools.**
  - **Working software over comprehensive documentation.**
  - **Customer collaboration over contract negotiation.**
  - **Responding to change over following a plan.**
- **Extreme Programming (XP):**
  - **User Stories: Incremental feature descriptions.**
  - **Pair Programming: Two developers work on the same code for quality assurance.**

    - **Test-Driven Development (TDD): Writing tests before code to ensure reliability.**
- **Scrum Framework:**
    - **Roles: Product Owner, Scrum Master, Development Team.**
    - **Key Components: Sprints, Sprint Backlog, Daily Stand-ups, Reviews.**
    - **Artifacts: Product Backlog, Sprint Backlog, Increments.**
    - **Scaling Agile: Managing large teams with frameworks like SAFe or LeSS.**

4. Software Development Models

- **Waterfall Model: Plan-driven, rigid structure.**
    - **Pros: Useful for stable projects.**
    - **Cons: Difficult to accommodate changes.**
- **Incremental Model: Iterative development, early feedback.**
- **Prototype Model: Early version to refine requirements.**
- **Spiral Model: Combines iterative development with risk management.**

5. Software Design & Implementation

- **Architecture Design: Defines software structure.**
- **Interface Design: Establishes interactions between system components.**
- **Component & Database Design: Organizing system elements for efficiency.**
- **Software Validation:**
    - **Unit Testing: Testing individual components.**
    - **Integration Testing: Ensuring smooth collaboration between modules.**
    - **System Testing: Checking overall functionality.**
    - **Acceptance Testing: Customer approval phase.**

6. Software Evolution & Maintenance

- **Reasons for Evolution: Business needs, security updates, scalability demands.**
- **Challenges: Maintaining code quality, avoiding technical debt.**
- **Strategies: Continuous integration, regular testing, agile adaptations.**

7. Use Cases & Diagrams

- **Purpose: Defines user-system interactions.**
- **Components: Actors, Use Cases, Relationships.**
- **Relationships:**
    - **Include: Reusing functionality.**
    - **Extend: Adding optional behavior.**
    - **Generalization: Parent-child relationships between elements.**

8. Key Takeaways

- **Software engineering combines structured methodologies with adaptability.**
- **Agile promotes flexibility, continuous feedback, and customer collaboration.**
- **Requirements engineering ensures software meets user and business needs.**

- **Development models vary based on project scope, complexity, and change tolerance.**
- **Testing and validation prevent costly errors and ensure quality software.**
- **Software must evolve to stay relevant in changing environments.**
-

Flashcards for Agile Software Development

Motivation for Agile

Q: Why was Agile developed?A: To address limitations of plan-driven software development in fast-changing environments by enabling incremental delivery and adaptability.

Q: What are the key advantages of Agile over traditional methods?A: Faster delivery, better adaptability, continuous feedback, customer collaboration, and reduced risk.

Q: Why do plan-driven approaches struggle in fast-changing environments?A: They require upfront requirements, making it difficult to adapt to changes once development starts.

Agile Manifesto & Principles

Q: What are the four key values of the Agile Manifesto?A: 1. Individuals & interactions over processes & tools2. Working software over comprehensive documentation3. Customer collaboration over contract negotiation4. Responding to change over following a plan

Q: What is the primary focus of Agile development?A: Delivering working software quickly and continuously adapting to changing requirements.

Extreme Programming (XP)

Q: What is the goal of Extreme Programming (XP)?A: To improve software quality and responsiveness through frequent releases, continuous testing, and customer feedback.

Q: What are User Stories in XP?A: Short, simple descriptions of features from the user's perspective to define requirements incrementally.

Q: What is pair programming?A: Two developers working together on the same code to improve quality and knowledge sharing.

Q: Why is refactoring important in XP?A: It keeps the code clean, maintainable, and adaptable without changing functionality.

Q: What is Test-Driven Development (TDD)?A: Writing tests before writing code to ensure correctness and guide development.

Scrum Framework

Q: What are the three key roles in Scrum?A: Product Owner, Scrum Master, Development Team.

Q: What is the purpose of a Sprint?A: A fixed-length iteration (2-4 weeks) where a working product increment is developed.

Q: What are the key Scrum events?A: Sprint Planning, Daily Stand-ups, Sprint Review, and Sprint Retrospective.

Q: What is the Product Backlog?A: A prioritized list of all desired features and tasks in a project.

Q: What is the Sprint Backlog?A: A subset of the Product Backlog selected for development in the current sprint.

Q: What is an Increment in Scrum?A: A working, potentially shippable piece of software developed during a sprint.

Agile Project Management

Q: What is backlog prioritization?A: The process of ordering backlog items based on business value and urgency.

Q: What happens in a Sprint Review?A: The team presents the completed work to stakeholders for feedback.

Q: What is a Sprint Retrospective?A: A meeting where the team reflects on the past sprint and identifies areas for improvement.

Scaling Agile

Q: What are the challenges of scaling Agile to large teams?A: Communication, system complexity, team coordination, and stakeholder involvement.

Q: How can Agile be scaled effectively?A: Using frameworks like SAFe or LeSS, fostering cross-team communication, and maintaining Agile principles.

Q: What is Continuous Integration (CI)?A: Frequently merging and testing code changes to detect issues early.

Q: Why is Agile beneficial for large-scale projects?A: It enhances adaptability, improves collaboration, and ensures continuous delivery.

Agile vs. Traditional Software Development

Q: How does Agile differ from Waterfall?A: Agile is iterative, adaptive, and focused on continuous feedback, while Waterfall follows a linear, sequential approach.

Q: When is Waterfall preferable over Agile?A: In highly regulated, safety-critical, or well-defined projects where changes are minimal.

Key Benefits of Agile

Q: How does Agile improve product quality?A: Through continuous testing, refactoring, and collaboration with stakeholders.

Q: How does Agile enhance customer satisfaction?A: By involving customers throughout development and delivering usable increments frequently.

Q: How does Agile manage risk better than traditional methods?A: By allowing early identification of issues and adapting the project scope continuously.

Additional Flashcards for Deeper Understanding

Q: What are the key differences between XP and Scrum?A: XP focuses on engineering practices (TDD, pair programming), while Scrum focuses on project management (Sprints, roles, backlogs).

Q: How does Agile ensure better team collaboration?A: Through daily stand-ups, shared backlogs, and collective ownership of the work.

Q: What is the role of a Scrum Master in detail?A: Facilitator, coach, and servant leader who removes obstacles, enforces Scrum rules, and supports the team.

Q: What are the most common Agile testing strategies?A: Unit testing, integration testing, automated testing, and acceptance testing.

Q: How does Agile handle technical debt?A: Through continuous refactoring, prioritizing maintenance tasks, and automated testing.

Q: How is risk management handled in Agile?A: By delivering small increments, continuous testing, and involving stakeholders in decision-making.

Q: What are some best practices for writing effective user stories?A: Keep them simple, focused, testable, and include acceptance criteria.

Q: What is velocity in Agile, and why is it important?A: A measure of work completed per sprint, helping teams plan future sprints realistically.

Q: How do teams measure progress in Agile projects?A: Using burndown charts, velocity tracking, and completed increments.

Q: What are some common Agile anti-patterns?A: Lack of customer involvement, skipping retrospectives, overloading sprints, and poor backlog refinement.

Q: Why is Agile widely adopted in software development?A: It enables faster delivery, adaptability, and higher stakeholder satisfaction.

Q: What is the key takeaway from Agile methodologies?A: Agile is about delivering value incrementally, responding to change, and fostering collaboration.

Flashcards for Software Engineering Concepts

Understanding Software Development Challenges

Q: What are common challenges in software development?A: Miscommunication, unclear expectations, bugs, crashes, maintainability, scalability, and efficiency issues.

Q: What are some critical failures in software history?A: Therac-25 radiation overdoses, Schiaparelli Mars lander crash, CrowdStrike outage affecting global systems.

Q: Why do software failures still happen despite advancements?A: Increasing system complexity, fast-changing demands, and evolving business needs.

Q: How can software engineers prevent major failures?A: Implement rigorous testing, ensure clear requirement gathering, conduct risk assessments, and adopt best engineering practices.

Attributes of Good Software

Q: What are the key attributes of good software?A: Scalability, modularity, maintainability, security, and efficiency.

Q: What does it mean for software to be modular?A: Loosely coupled but tightly cohesive components that work well together.

Software Engineering Process

Q: What is software engineering?A: A discipline concerned with all aspects of software production, from specification to maintenance.

Q: What are the four major phases of the software process?A: Specification, development, validation, and evolution.

Q: What is software validation?A: Ensuring the software meets customer requirements through rigorous testing.

Q: What is software evolution?A: The continuous modification of software to adapt to changing requirements and environments.

Types of Software Development

Q: What are the different types of software?A: Stand-alone applications, transaction-based applications, embedded systems, batch processing, entertainment systems, modeling & simulation systems.

Q: What are generic software products?A: Stand-alone software sold to multiple users (e.g., Photoshop, CAD software).

Q: What are customized software products?A: Software developed for a specific customer with unique requirements (e.g., Tesla’s embedded systems).

Software Process Models

Q: What is the difference between a plan-driven and an Agile process?A: Plan-driven follows a rigid, predefined plan, while Agile allows incremental development and adaptability.

Q: What is the Waterfall model?A: A sequential software development approach with distinct phases like specification, design, implementation, testing, and maintenance.

Q: What are the disadvantages of the Waterfall model?A: Difficult to accommodate changes, rigid phase transitions, and long time before users see the product.

Q: What is the Incremental Model?A: A model where development occurs in small increments, allowing for early user feedback and adaptation.

Q: What are the benefits of the Incremental Model?A: Faster delivery, reduced cost of changes, easier customer feedback, and early usability testing.

Q: What is the Prototype Model?A: An approach where an early working version of the system is created for customer feedback and design exploration.

Q: What is the Spiral Model?A: A risk-driven model that combines iterative development with risk assessment.

Q: What are the key phases of the Spiral Model?A: Objective setting, risk assessment, development & validation, and planning the next phase.

Software Specification & Requirements Engineering

Q: What is requirements engineering?A: The process of understanding and defining what services a system must provide and its constraints.

Q: What are the three main activities of requirements engineering?A: Elicitation & analysis, specification, and validation.

Q: What are the two types of software requirements?A: User requirements (high-level needs) and system requirements (detailed specifications).

Software Design & Implementation

Q: What is software design?A: The process of defining the architecture, components, and interactions of a system.

Q: What are the four key design activities?A: Architectural design, interface design, component design, and database design.

Software Validation & Testing

Q: What is software verification & validation (V&V)?A: Ensuring the software meets its specification and customer expectations.

Q: What are the different levels of software testing?A: Unit testing, integration testing, system testing, and acceptance testing.

Q: What is automated testing?A: The use of tools to execute tests automatically, ensuring consistency and efficiency.

Software Evolution & Maintenance

Q: Why is software evolution necessary?A: Changing business needs, emerging security threats, and new technology adoption.

Q: What is the difference between software development and maintenance?A: Development is creating the initial software, while maintenance involves continuous updates and fixes.

Key Takeaways

Q: Why is software engineering important?A: It ensures that software is built efficiently, reliably, and can evolve to meet future demands.

Q: How does Agile improve software development?A: It allows iterative progress, flexibility, and continuous user feedback.

Q: What is the role of documentation in software engineering?A: It provides clarity on design, functionality, and maintenance, ensuring long-term usability.

Q: Why is security a major concern in modern software?A: Increasing digital footprints and cyber threats require strong security measures and trust in software.

Q: What is the impact of poor software engineering practices?A: Higher failure rates, financial losses, security breaches, and user dissatisfaction.

Q: What is the significance of scalability in software?A: Ensures the software can handle growing user demands and data loads efficiently.

Q: Why is requirement gathering critical?A: Poor requirements lead to costly changes, misaligned products, and project failures.

Q: How do software engineers manage risks in development?A: Through risk assessment, iterative testing, and continuous validation.

Q: What are the most critical principles in software engineering?A: Clarity in requirements, strong design principles, rigorous testing, adaptability, and security awareness.

Flashcards for Software Requirements Engineering

Software Requirements

Q: What is a software requirement?A: A system's functionality description and operational constraints.

Q: What is requirements engineering (RE)?A: The process of gathering, analyzing, documenting, and validating software requirements.

Q: What are functional requirements?A: Define system behavior, inputs, and expected outputs.

Q: What are non-functional requirements?A: System constraints like performance, security, and usability.

Q: Why are non-functional requirements critical?A: They impact usability, efficiency, and system reliability.

Types of Non-Functional Requirements

Q: What are product requirements?A: Define system behavior, e.g., speed, reliability.

Q: What are organizational requirements?A: Driven by policies, standards, or business needs.

Q: What are external requirements?A: Constraints from external factors, e.g., legal regulations.

Examples of Requirements

Q: Give an example of a functional requirement in a food delivery app.A: "Users can place orders and pay via a digital wallet."

Q: Give an example of a non-functional requirement in a food delivery app.A: "90% of menu load times must be under 2 seconds."

Requirements Elicitation Techniques

Q: What is requirements elicitation?A: The process of gathering system requirements from stakeholders.

Q: What are common elicitation techniques?A: Interviews, scenarios, use cases, prototypes, and observations.

Q: What are the two types of interviews in elicitation?A: Closed (structured) and open (exploratory).

Q: What is a scenario in requirements elicitation?A: A real-world example illustrating system use and potential errors.

Q: What is a use case?A: A description of user interactions with a system.

Q: What is a prototype in requirements elicitation?A: A working model to clarify and validate requirements.

Q: What is ethnography in software requirements gathering?A: Observing users in their real environment to understand workflows.

Q: What is a user story in Agile development?A: A brief feature description in the format: "As a [user], I want [feature] so that [benefit]."

Requirements Validation

Q: Why is requirements validation important?A: Detects errors early, reducing costly fixes later.

Q: What are key factors in requirements validation?A: Validity, consistency, completeness, realism, and verifiability.

Q: What is an example of an inconsistency in requirements?A: "Real-time stock reports" but "stock updates occur daily."

Requirements Analysis

Q: What happens during requirements analysis?A: Classification, prioritization, modeling, and conflict resolution.

Q: What is conceptual modeling?A: Creating abstract representations of system requirements.

Q: What is requirements negotiation?A: Resolving stakeholder conflicts and aligning requirements.

Software Requirements Specification (SRS)

Q: What is an SRS document?A: A structured document detailing functional and non-functional requirements.

Q: Why is an SRS document important?A: Serves as a reference for developers, testers, and stakeholders.

Requirements Specification Techniques

Q: What are structured specifications?A: Organized textual requirements for clarity and consistency.

Q: What is tabular specification?A: Using tables to display requirements systematically.

Use Cases in Requirements Engineering

Q: What is a use case diagram?A: A visual representation of system-user interactions.

Q: What are the components of a use case diagram?A: Actors, use cases, associations, and relationships.

Q: What are the three types of actors in a use case?A: Primary (users), supporting (external systems), offstage (regulatory bodies).

Q: What are the benefits of use cases?A: Improve clarity, stakeholder agreement, and system design.

Use Case Examples

Q: How does Uber Pay use APIs in its payment integration?A: Enables users to pay via external services seamlessly.

Q: What is an include relationship in a use case diagram?A: One use case reuses another's functionality.

Q: What is an extend relationship in a use case diagram?A: One use case adds optional behavior to another.

Final Takeaways

Q: Why is requirements engineering critical in software development?A: Ensures system alignment with user needs and prevents costly rework.

Q: How do well-defined requirements improve project success?A: Provide clarity, reduce misunderstandings, and streamline development.

Q: What is the impact of poor requirements gathering?A: Leads to project delays, cost overruns, and user dissatisfaction.

Q: How does Agile handle requirements differently than traditional models?A: Uses user stories, continuous feedback, and incremental development.

Q: What is the ultimate goal of software requirements engineering?A: To build a system that meets user needs, business objectives, and technical feasibility.