Aaron Rodrigues
9635
TE Comps B

1. Significance of Recognizing Software Requirements:
Recognizing software requirements is a crucial initial step in the software engineering process. Requirements define what the software should achieve and the features it must have. Properly identifying and documenting requirements helps ensure that the final product meets user needs and expectations. Clear requirements reduce misunderstandings between developers and clients, guide design and development efforts, provide a basis for testing and validation, and facilitate project management and communication.

2. Main Characteristics of Different Process Models:
There are several software development process models, each with its own

characteristics:
- Waterfall Model: Sequential, linear approach where each phase (requirements, design, implementation, testing, deployment) is completed before moving to the next.
- Agile Methodologies (e.g., Scrum and Kanban): Iterative and incremental approach, focusing on flexibility, collaboration, and customer feedback. Work is divided into small cycles (sprints) in Scrum, while Kanban emphasises continuous flow and visual management of tasks.

3. Contribution of Capability Maturity Model (CMM):
The Capability Maturity Model (CMM) is a framework for improving the software development process. It provides a structured approach to assessing and improving an organisation's software engineering capabilities. By defining levels of maturity (initial, repeatable, defined, managed, optimising), CMM helps organisations identify weaknesses and

characteristics:
- Waterfall Model: Sequential, linear approach where each phase (requirements, design, implementation, testing, deployment) is completed before moving to the next.
- Agile Methodologies (e.g., Scrum and Kanban): Iterative and incremental approach, focusing on flexibility, collaboration, and customer feedback. Work is divided into small cycles (sprints) in Scrum, while Kanban emphasises continuous flow and visual management of tasks.

3. Contribution of Capability Maturity Model (CMM):
The Capability Maturity Model (CMM) is a framework for improving the software development process. It provides a structured approach to assessing and improving an organisation's software engineering capabilities. By defining levels of maturity (initial, repeatable, defined, managed, optimising), CMM helps organisations identify weaknesses and

strengths in their processes. It guides them in gradually improving their processes, thereby enhancing efficiency, quality, and predictability.

4. Differences between Prescriptive and Evolutionary Process Models:
- Prescriptive Models (e.g., Waterfall): Follow a predetermined sequence of phases. Changes are discouraged after the initial planning phase.
- Evolutionary Models (e.g., Agile): Allow for flexibility and change throughout the project's lifecycle. Development happens in iterations, and the product evolves based on frequent feedback.

5. Suitability of Process Models in Different Situations:
- Waterfall: Suitable when requirements are well-defined and unlikely to change significantly.
- Agile (Scrum): Suitable when requirements are subject to change, and frequent customer collaboration is possible.

- Kanban: Suitable for continuous improvement scenarios with a focus on managing and optimising workflow.

6. Comparison of Waterfall and Agile (Scrum) in Project Planning and Progress Tracking:
- Waterfall: Detailed planning upfront, limited flexibility, progress tracked through completion of sequential phases.
- Agile (Scrum): Adaptive planning, flexibility to change requirements, progress tracked through sprint cycles and regular reviews.

7. Applying Process Metrics to Evaluate Waterfall, Agile (Scrum & Kanban):
- Development Speed: Agile generally has faster development due to iterative cycles, while Waterfall may be slower due to its sequential nature.
- Adaptability to Change: Agile outperforms Waterfall in handling changes due to its

iterative
and incremental approach.
- Customer Satisfaction: Agile methodologies prioritise customer feedback, enhancing customer satisfaction. Waterfall might have less customer involvement until the end.
8.

1. Development Approach:
- Waterfall: Linear phases, no overlap.
- Incremental: Iterative additions in phases.
- Prototyping: Initial version for feedback.
- Spiral: Iterative cycles with risk focus.

2. Flexibility and Change:
- Waterfall: Limited adaptability.
- Incremental: Adaptation in increments.
- Prototyping: Responsive to feedback.
- Spiral: Built-in flexibility.

3. Risk Management:
- Waterfall: Early risk assessment.

- Incremental: Increment-specific risks.
- Prototyping: Risks refined via feedback.
- Spiral: Continuous risk assessment.

4. Customer Involvement:
- Waterfall: Minimal until end.
- Incremental: Feedback in each increment.
- Prototyping: Strong early involvement.
- Spiral: Continuous customer interaction.

5. Suitability for Large Projects:
- Waterfall: Challenging due to late changes.
- Incremental: Manageable for large projects.
- Prototyping: Complexity might pose challenges.
- Spiral: Suited for complexity.

6. Documentation:
- Waterfall: Extensive documentation.
- Incremental: Incremental documentation.
- Prototyping: Less formal but crucial.

- Spiral: Emphasis on risk and progress documentation.

7. Progress Tracking:
- Waterfall: Phase-based tracking.
- Incremental: Increment completion.
- Prototyping: Prototype evolution.
- Spiral: Iteration and risk tracking.