Crystal Fernandes
9539

Significance of Recognizing Software Requirements:

Foundation: Requirements provide the foundation for software development, outlining functionalities and constraints.
Communication: Clear requirements ensure mutual understanding among stakeholders, avoiding misunderstandings.
Scope Control: Requirements define project scope, preventing scope creep during development.
Risk management: Identifying requirements helps manage potential risks and challenges.
Resource Allocation: Clear requirements aid in resource allocation for the project.
Validation and Satisfaction: Requirements guide testing and ensure user needs are met, leading to satisfaction.
Traceability: Requirements enable tracking changes and rationale, vital for maintenance.
Characteristics of Different Process models:

Waterfall model: Linear phases, suitable for stable requirements.
Incremental model: Iterative, early deliverables, adaptable to changes.
Prototyping model: Early prototypes, feedback-driven, useful for exploring requirements.
Spiral model: Risk-driven, iterative with phases and evaluations,

ideal for complex projects.

Capability maturity model (Cmm) and Process Improvement:

Structured Approach: Cmm provides a structured framework to assess and improve software development processes.
Levels of maturity: Defines five levels of maturity, from initial (chaotic) to optimized (continuous improvement).
Guidance: Offers guidelines to achieve higher maturity levels through defined processes and best practices.
Quality Focus: Emphasizes quality, consistency, and efficiency in software development.
measurable Improvement: Provides measurable indicators to gauge process improvement.
Best Practices: Encourages adoption of proven best practices, leading to more predictable and successful outcomes.

Prescriptive Process models vs. Evolutionary Process models:
Prescriptive models: Also known as plan-driven models, they emphasize a structured, planned approach from the start.
Examples: Waterfall, V-model
Phases are well-defined and sequenced.
Suitable for projects with clear and stable requirements.
Changes are difficult to accommodate after initial planning.
Evolutionary models: Emphasize iterative development and evolving

solutions over time.
Examples: Agile, Scrum.
Development occurs in iterations, allowing flexibility and adaptability.
Requirements can evolve, making them suitable for dynamic projects.
Feedback-driven, enabling continuous improvement.
Regular deliveries promote stakeholder involvement.

Process model Suitability Examples:
Waterfall: Suitable for projects with well-defined, stable requirements, and where changes are costly or undesirable. For instance, developing a simple brochure website.
Agile: Ideal for projects with evolving requirements, frequent feedback, and a need for rapid adaptability. For example, developing a mobile app with changing user needs.
Waterfall vs. Agile: Project Planning and Progress Tracking:

Waterfall:
Project Planning: Detailed planning upfront, with all phases outlined before starting.
Progress Tracking: Linear progression through phases, each with its own deliverables and milestones.
Feedback: Limited customer involvement until the end, making it challenging to catch deviations early.
Adaptability: Less adaptable to changing requirements after initial planning.
Agile:
Project Planning: High-level planning at the beginning, detailed planning before each iteration (sprint).
Progress Tracking: Progress tracked through regular iterations, each delivering functional increments.
Feedback: Frequent stakeholder involvement, allowing quick adjustments and course corrections.

Adaptability: Highly adaptable to changing requirements and priorities throughout the project.

Process metrics Comparison: Waterfall, Agile (Scrum + Kanban)

Development Speed:

Waterfall: Slower development due to sequential phases and upfront planning.
Agile (Scrum): Faster development due to time-boxed iterations (sprints) with continuous feedback and adjustments.
Agile (Kanban): Steady development pace, focused on optimizing flow and reducing bottlenecks.
Adaptability to Change:
Waterfall: Less adaptable; changes are costly and require revisiting multiple phases.
Agile (Scrum): moderately adaptable; changes can be accommodated in upcoming sprints.
Agile (Kanban): Highly adaptable; changes can be made quickly as the workflow is continuous.
Customer Satisfaction:
Waterfall: Customer satisfaction might be lower if requirements evolve or if feedback isn't gathered until the end.

Agile (Scrum): High customer satisfaction due to regular feedback and incremental delivery of valuable features.
Agile (Kanban): Continuous focus on meeting customer needs, leading to higher satisfaction.
Efficiency:

Waterfall: might be less efficient due to potential rework in later stages if initial requirements aren't accurate.
Agile (Scrum): Efficient through frequent collaboration, shorter development cycles, and adaptability.
Agile (Kanban): Efficient by optimizing workflow, minimizing waste, and addressing bottlenecks.
Effectiveness:

Waterfall: Effective for stable projects with well-defined requirements and minimal changes.
Agile (Scrum): Effective for projects with evolving requirements, where regular iterations lead to better alignment.
Agile (Kanban): Effective for continuous and incremental improvement, responding to changing demands.
Risk management:

Waterfall: Risks are addressed in early planning stages, but changes in later phases can lead to increased risk.
Agile (Scrum): Risks are continuously managed through iterations,

allowing for early identification and resolution.
Agile (Kanban): Focus on managing risks as they arise, leading to timely resolution.
Overall, Agile methodologies (Scrum and Kanban) generally outperform Waterfall in terms of development speed, adaptability, customer satisfaction, efficiency, and risk management. Scrum provides a structured framework for iterative development, while Kanban focuses on optimizing workflow. The choice between Scrum and Kanban depends on the project's specific needs and context. Waterfall may be suitable for projects with stable requirements and minimal changes, but it lacks the agility and responsiveness of Agile methodologies.

allowing for early identification and resolution.
Agile (Kanban): Focus on managing risks as they arise, leading to timely resolution.
Overall, Agile methodologies (Scrum and Kanban) generally outperform Waterfall in terms of development speed, adaptability customer satisfaction, efficiency, and risk management. Scrum provides a structured framework for iterative development, while Kanban focuses on optimizing workflow. The choice between Scrum and Kanban depends on the project's specific needs and context. Waterfall may be suitable for projects with stable requirements and minimal changes, but it lacks the agility and responsiveness of Agile methodologies.

8) Relevancy:
Waterfall model:
Relevance: The Waterfall model is well-suited when requirements are well understood and stable from the beginning, and changes are minimal.
Cost: The cost is generally low due to the linear nature of the model, but changes can be expensive later in the process.
Complexity: Suitable for simpler systems with straightforward requirements.

Risk Analysis: Risk analysis is done primarily at the beginning, and the model assumes a low level of uncertainty.
User Involvement: User involvement is highest at the beginning during requirement gathering.
Guarantee of Success: The guarantee of success is less compared to other models due to limited adaptation to changes.
Incremental model:
Relevance: Suitable when there's a clear overall vision but specifics evolve over time. Allows for building a working core and adding features incrementally.
Cost: Initial cost might be low, but costs can increase as more increments are developed.
Complexity: Applicable for systems with moderate complexity and evolving requirements.
Risk Analysis: Risk analysis is ongoing, and each increment adds to risk reduction.
User Involvement: User involvement is maintained throughout the development process, especially during each increment.
Guarantee of Success: Higher guarantee of success compared to Waterfall, as regular feedback minimizes deviations.
Prototyping model:
Relevance: Suitable when requirements are not well understood or frequently changing. Allows users to visualize the system early.
Cost: Initial cost might be higher due to iterative nature, but later changes are easier to incorporate.

Complexity: Can handle moderately complex systems with evolving requirements.
Risk Analysis: Focuses on getting user feedback early, addressing risks associated with unclear requirements.
User Involvement: High user involvement throughout due to iterative feedback loops.
Guarantee of Success: Good guarantee of success due to regular user validation and feedback.
Spiral model:
Relevance: Suited for projects with high uncertainty, complex requirements, and evolving risks.
Cost: Can be expensive due to multiple cycles of development and risk analysis.
Complexity: Applicable for complex and evolving systems.
Risk Analysis: Involves frequent risk analysis and mitigation in each spiral phase.
User Involvement: User involvement throughout, adapting to changes and providing feedback.
Guarantee of Success: High guarantee of success due to continuous risk analysis and iterative development.