SE QUESTION BANK

1. **What is the significance of recognizing software requirements in the software engineering process?**

   Recognizing software requirements is a fundamental and crucial step in the software engineering process. It lays the foundation for successful software development by providing a clear understanding of what the software needs to accomplish and how it should behave. Here's why recognizing software requirements is significant:

   Clear Communication: Requirements act as a bridge between clients, stakeholders, and the development team. This helps in avoiding misunderstandings and ensures that everyone is on the same page.

   Risk Management: Recognizing and documenting requirements allows potential risks and challenges to be identified early in the development process. This enables the development team to mitigate risks and plan for potential issues before they become major roadblocks.

   Customer Satisfaction: Clear and well-defined requirements contribute to higher customer satisfaction. When the final product aligns with the initially agreed-upon requirements, it increases the chances of meeting customer expectations and needs.

   Validation and Verification: Requirements serve as a basis for validating and verifying the software. By comparing the developed software with the documented requirements, the development team can ensure that the software meets the desired functionality and quality standards.

2. **Describe the main characteristics of different process models used in software development.**

- **Waterfall Model:**
  Sequential Approach: The development process follows a linear and sequential sequence of phases, where each phase must be completed before moving to the next.
  Documentation: Emphasis is placed on extensive documentation at each phase.

- **Iterative Model:**
  Flexibility: Allows for adapting to changing requirements and addressing risks early in the development process.
  Incremental Development: Software is built incrementally, with each iteration adding new features or enhancing existing ones.

- **Agile Model:**
  Customer-Centric: Customer feedback drives development priorities and helps shape the software's direction.
  Flexibility: Changes to requirements are expected and even welcomed, with the goal

of delivering value early and often.

- **Spiral Model:**
  Risk Management: Focuses on addressing project risks through a series of iterative cycles.
  Flexibility: Well-suited for projects with high uncertainty, complex requirements, and significant risks.

- **V-Model (Validation and Verification Model):**
  Emphasis on Testing: Testing and verification are integral to each phase of development, ensuring quality at every step.
  Traceability: Requirements and corresponding test cases are linked, ensuring that each requirement is adequately tested.

3. **How does the Capability Maturity Model (CMM) contribute to improving software development processes?**

The Capability Maturity Model (CMM) is a framework that provides a structured approach to assessing and improving an organization's software development processes. The primary goal of CMM is to enhance the quality, efficiency, and effectiveness of software development by establishing a set of best practices and guidelines. CMM contributes to improving software development processes in the following ways:

Process Standardization: CMM defines a set of process areas and key practices that organizations can adopt.

Assessment and Benchmarking: CMM provides a structured way to assess an organization's current software development processes.

Incremental Improvement: This staged approach allows organizations to focus on specific areas at a time and gradually enhance overall process maturity.

Focus on Quality and Predictability: This results in improved product quality, reduced defects, and greater predictability in meeting project schedules and budgets.

Risk Management: CMM emphasizes identifying and mitigating risks in software development processes

Best Practices Adoption: CMM provides a repository of best practices that organizations can adopt.

4. **Explain the differences between prescriptive process models and evolutionary process models.**

Prescriptive Process Models:	Evolutionary Process Models:
Sequential Approach: Prescriptive process models, such as the Waterfall model and the V-Model, follow a sequential and well-defined path of development phases.	Iterative and Incremental Approach: Evolutionary process models, such as Agile and Iterative models, emphasize iterative development and incremental delivery of software.
Phases: These models typically involve phases like requirements analysis, design, implementation, testing, deployment, and maintenance.	Cycles/Iterations: These models involve repeating cycles of development, where each cycle includes phases like requirements, design, implementation, testing, and deployment.
Rigidity: These models tend to be more rigid and less adaptable to changes in requirements.	Flexibility and Adaptability: Evolutionary models are more flexible and adaptable to changing requirements.
Suitability: These models are suited for projects with clear and stable requirements, where changes are expected to be minimal.	Suitability: Evolutionary models are suited for projects where requirements are subject to change, customer collaboration is crucial, and delivering functional software quickly is a priority.

5. **Provide examples of situations where using a specific process model would be more suitable.**

Waterfall Model:
Situation: When the project has well-defined and stable requirements that are unlikely to change significantly.
Example: Developing a simple website with clear and fixed specifications, where the client's requirements are fully understood and won't change during the development process.

Agile Model:
Situation: When the project's requirements are subject to change and customer collaboration is crucial.
Example: Developing a mobile app for an e-commerce platform where features and user interfaces may need to be adjusted based on user feedback and market trends.

V-Model:
Situation: When a strong emphasis on testing and verification is required for critical and safety-critical systems.
Example: Developing software for medical devices or aerospace systems where thorough testing and validation are essential to ensure safety and compliance.

Incremental Model:
Situation: When the project requires regular delivery of specific features or modules

for early value creation.
Example: Developing a content management system where basic functionality (user authentication, content creation) is delivered in the first increment, followed by additional modules (e.g., advanced search, user analytics) in subsequent increments.

6. Compare and contrast the Waterfall model and Agile methodologies in terms of project planning and progress tracking.

Parameters	Waterfall model	Agile
Project planning	Detailed planning upfront	Iterative planning, adapting as the project evolves.
Progress Tracking	Linear and milestone-driven	Incremental and sprint-based, with frequent updates.
Flexibility	Less adaptable to changes.	Highly adaptable to changes and feedback.
Risk Management	Risks addressed early; less adaptable to emerging risks.	Ongoing risk management, quick response to risks.
Communication	Limited stakeholder involvement after initial planning.	Continuous collaboration and involvement of stakeholders.

7. Apply process metrics to evaluate the efficiency and effectiveness of Waterfall , Agile (both Scrum & Kanban) methodologies, considering factors such as development speed, adaptability to change and customer satisfaction.

Process metrics are essential for evaluating the efficiency and effectiveness of software development methodologies. Let's apply process metrics to assess the Waterfall model, Agile methodologies (Scrum and Kanban), and consider development speed, adaptability to change, and customer satisfaction:
Development Speed:
Waterfall:
Process Metric: Time to Complete Project.
Evaluation: Measure the time taken from project initiation to final delivery. Compare against estimated timelines.
Agile (Scrum):
Process Metric: Sprint Velocity.
Evaluation: Measure the number of story points completed in each sprint. Calculate average sprint velocity over time.
Agile (Kanban):
Process Metric: Cycle Time.
Evaluation: Measure the time taken for a task or user story to move from the "To Do" column to the "Done" column on the Kanban board.

Adaptability to Change:

Waterfall:
Process Metric: Change Requests.
Evaluation: Track the number of change requests submitted during development and how well they were accommodated.

Agile (Scrum):
Process Metric: Sprint Changes.
Evaluation: Measure the frequency of changes introduced mid-sprint. Observe how well the team adapts to new requirements.

Agile (Kanban):
Process Metric: Lead Time Distribution.
Evaluation: Analyze the distribution of lead times for tasks or user stories. Identify how quickly changes are integrated.

Customer Satisfaction:
Waterfall:
Process Metric: Customer Feedback.
Evaluation: Collect feedback from the customer after project completion. Assess whether the delivered product meets their expectations.

Agile (Scrum):
Process Metric: Sprint Review Ratings.
Evaluation: After each sprint review, gather feedback and ratings from stakeholders. Assess their satisfaction with the delivered work.

Agile (Kanban):
Process Metric: Customer Feedback Loop Time.
Evaluation: Measure the time it takes for customer feedback to be implemented after being received.

8. Justify the relevancy of the fallowing comparison for software development models.

Features	Water fall Model	Incremental Model	Prototyping Model	Spiral Model
Requirement Specification	Beginning	Beginning	Frequently Changed	Beginning
Understanding Requirements	Well Understood	Not Well Understood	Not Well Understood	Well Understood
Cost	Low	Low	High	Expensive
Availability of reusable component	No	Yes	Yes	Yes
Complexity of System	Simple	Simple	Complex	Complex
Risk Analysis	Only at beginning	No risk analysis	No risk analysis	Yes

User involvement in all phases of SDLC	Only at beginning	Intermediate	High	High
Guarantee of Success	Less	High	Good	High
Overlapping Phases	Absent	Absent	Present	Present
Implementation Time	Long	Less	Less	Depends on Project
Flexibility	Rigid	Less flexible	Highly flexible	Flexible
Changes Incorporated	Difficult	Easy	Easy	Easy
Expertise Required	High	High	Medium	High
Cost Control	Yes	No	No	Yes
Resource Control	Yes	Yes	No	Yes

Rubrics :

Indicator	**Average**	**Good**	**Excellent**	**Marks**
Organization (2)	Readable with some mistakes and structured (1)	Readable with some mistakes and structured (1)	Very well written and structured (2)	
Level of content(4)	Minimal topics are covered with limited information (2)	Limited major topics with minor details are presented(3)	All major topics with minor details are covered (4)	
Depth and breadth of discussion(4)	Minimal points with missing information (1)	Relatively more points with information (2)	All points with in depth information(4)	
Total Marks(10)				