FR. CONCEICAO RODRIGUES COLLEGE OF ENGG.

Fr. Agnel Ashram, Bandstand, Bandra (W) Mumbai 400 050.

SEMESTER / BRANCH: V/COMPUTER Engineering

SUBJECT: Software Engineering **(CSC502)/ First Assignment**

Date: 19-08-23 Due Date : 25-08-23

CSC502.1: Recognize software requirements and various process models. (Understanding)
CSC502.2: Develop project Plan, schedule and track the progress of the given project (Applying)

Questions :

1. What is the significance of recognizing software requirements in the software engineering process?
2. Describe the main characteristics of different process models used in software development.
3. How does the Capability Maturity Model (CMM) contribute to improving software development processes?
4. Explain the differences between prescriptive process models and evolutionary process models.
5. Provide examples of situations where using a specific process model would be more suitable.
6. Compare and contrast the Waterfall model and Agile methodologies in terms of project planning and progress tracking.
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.
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)				

Q1) What is the significance of recognizing software requirements in the software engineering process?

The significance of recognizing software requirements lies in the following key points:

1. Understanding User Needs: Software requirements capture the needs, expectations, and goals of users and stakeholders. Recognizing and documenting these requirements ensure that the software system addresses the actual needs of its intended users.

software project. This prevents scope creep, which occurs when the project's objectives and features continuously expand beyond what was initially planned.

4. Communication: Well-defined requirements facilitate effective communication among team members, stakeholders, developers, and users. Everyone involved can have a common understanding of what the software should achieve.

5. Risk management: Identifying requirements early in the process helps identify potential risks and challenges. Addressing these issues

Divides the project into distinct phases (requirements, design, implementation, testing, deployment).

- Each phase must be completed before the next one begins.
- Emphasizes documentation and planning.
- Suitable for projects with well-defined and stable requirements.

2. Iterative model:

- Involves repeating cycles of development.
- Each cycle produces a working version of the software.
- Each iteration builds upon the previous one, incorporating feedback.

/>

4. Agile model (e.g., Scrum, Kanban):

- Emphasizes flexibility, collaboration, and customer feedback.
- Uses short development iterations called "sprints."
- Focuses on delivering incremental value to customers.
- Allows for changes and adjustments throughout development.

5. Spiral model:

- Combines iterative and incremental development with risk assessment.
- Each cycle involves four phases: planning, risk

/>- Emphasizes rigorous testing to ensure quality.

7. Rapid Application Development (RAD):

- Focuses on rapid prototyping and user involvement.
- Uses a series of prototyping cycles to refine the software.
- Well-suited for projects requiring fast delivery and frequent changes.

Q3) How does the Capability maturity model (Cmm) contribute to improving software development processes?

The Cmm achieves these goals through several

/>

2. Continuous Improvement:

The Cmm emphasizes the importance of continuous improvement. Organizations are encouraged to evaluate their current processes, identify areas for enhancement, and systematically work towards higher maturity levels.

3. Performance Measurement:

The Cmm encourages the establishment of metrics and measurements to assess the effectiveness of processes. This enables organizations to track progress, identify bottlenecks, and make data-driven decisions for

Predictability:

As organizations move up the maturity levels of the CMM, they develop more reliable and predictable processes. This leads to improved project planning, estimation, and delivery, reducing the likelihood of cost and schedule overruns.

Q4) Explain the differences between prescriptive process models and evolutionary process models.

Prescriptive Process models:

Also known as plan-driven or predictive models, prescriptive process models emphasize

follow a sequential and predefined order of phases, with each phase building upon the completion of the previous one.

2. Detailed Planning: Planning is done extensively upfront, including requirements gathering, design, development, testing, and deployment. The entire project is planned before execution begins.

3. Stability: Prescriptive models work best when project requirements are stable and well-defined at the beginning of the project.

4. Minimal Changes: Changes to requirements are typically discouraged or require formal change management due to the linear nature of the process.

/>risk mitigation strategies are established.

7. Suitability: Prescriptive models are suitable for projects with well-defined requirements, low levels of uncertainty, and where the project can be accurately estimated upfront.

Evolutionary Process models:

Evolutionary process models, also known as adaptive models, emphasize flexibility and adaptability to changing requirements and circumstances. These models involve iterative and incremental development, allowing the software to evolve gradually. Examples of evolutionary models include Agile methodologies (e.g., Scrum, Kanban) and the Spiral

/>functionality.

2. Continuous Adaptation: Evolutionary models embrace changing requirements and encourage flexibility in adapting to new insights and evolving user needs.

3. Early Feedback: Users are involved early in the process, providing feedback on working prototypes or increments of the software.

4. Less Upfront Planning: While some planning is involved, evolutionary models focus more on adapting to feedback and adjusting the development process as the project progresses.

5. Emphasis on Collaboration: Collaborative teamwork, frequent communication, and

suitable for projects with changing requirements, high levels of uncertainty, and a need for quick iterations and user feedback.

Q5) Provide examples of situations where using a specific process model would be more suitable.

Waterfall model:

The Waterfall model is suitable when the project requirements are well-defined and stable, and there is little likelihood of significant changes throughout the project. It's also appropriate for projects with a strict budget and timeline. For example:

is a need for

flexibility, frequent collaboration, and the ability

to adapt to changing requirements.

It's great for projects with evolving or unclear

requirements and for teams that value

continuous improvement. For example:

- Developing a complex software system with

multiple features that may

change during development.

- Designing a website with a dynamic user

interface that requires regular

updates based on user feedback.

- Spiral model:

The Spiral model is suitable when the project

large-scale enterprise software system that needs continuous user feedback and improvements.

Incremental model:

The Incremental model is suitable when the project can be divided into smaller manageable modules or components. It's great for projects where certain modules can be developed independently and then integrated. For example:

- Building an online e-commerce platform where individual modules like user authentication, product catalog, and payment gateway can be developed

lifecycle. It's especially useful for projects with stringent quality and testing requirements. For example:

- Developing safety-critical systems like automotive control systems or medical equipment.
- Building software for industries with strict regulatory compliance, such as financial services or healthcare.

Rapid Application Development (RAD) model:

The RAD model is suitable when there's a need to quickly develop a prototype or working version of the software for demonstration or evaluation. It's beneficial for

/>deadline.

Q6) Compare and contrast the Waterfall model and Agile methodologies in terms of project planning and progress tracking.

1. Waterfall model:

- Project Planning:

1. Sequential Phases: The Waterfall model follows a linear and sequential approach. Each phase (requirements, design, implementation, testing, deployment, maintenance) is completed one after the other.

2. Detailed Planning: Extensive planning is done at the beginning of the project to

/>2. Late Feedback: Feedback from stakeholders is usually gathered towards the end of the project, which may result in potential changes being difficult and costly to accommodate.

- Advantages:

1. Clear project scope and requirements.

2. Well-defined milestones and deliverables.

3. Predictable timeline and budget (assuming no major scope changes).

- Disadvantages:

1. Limited flexibility to accommodate changing requirements.

2. Late-stage discovery of defects or issues can

(sprints or iterations).

2. Adaptive Planning: Initial planning is done with a high-level vision, and detailed planning is carried out for the upcoming iteration. Plans can be adjusted based on ongoing feedback and changing priorities.

- Progress Tracking:

1. Iterative Progress: Progress is tracked at the end of each iteration, with working increments of the product delivered at the end of every cycle.

2. Frequent Feedback: Agile methodologies prioritize continuous feedback from stakeholders and end-users, allowing for early detection of issues and quicker

improvement.

- Disadvantages:

1. Can be challenging to estimate the overall project timeline and scope accurately.

2. may require more frequent communication and collaboration with stakeholders.

3. Complexities in managing changing priorities and balancing workloads.

Q7) 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.

Average time per phase, time taken to move from one phase to another.

- Scrum:

- Scrum emphasizes time-boxed iterations (sprints) that lead to regular, potentially shippable increments.

- metrics: Velocity (amount of work completed per sprint), lead time (time taken from request to delivery), cycle time (time taken for a single item to move from start to finish).

- Kanban:

- Kanban focuses on optimizing flow by reducing work-in-progress (WIP) and ensuring

to change due to its rigid sequential nature.

- metrics: Change requests after the initial planning phase, percentage of completed requirements that changed.

- Scrum:

- Scrum is adaptable through sprint planning, backlog grooming, and sprint reviews that allow continuous feedback and adjustments.

- metrics: Changes requested mid-sprint, number of scope changes during a sprint.

- Kanban:

- Kanban's focus on flexibility allows teams to adapt to changing priorities and requirements

by the long duration before the final product is delivered.

- metrics: User acceptance testing results, customer feedback at the end of the project.

- Scrum:

- Regular feedback and incremental delivery in Scrum help ensure customer satisfaction throughout the project.

- metrics: Customer feedback at sprint reviews, Net Promoter Score (NPS).

- Kanban:

- Kanban's focus on delivering value continually contributes to maintaining customer

because it highlights the distinct characteristics and trade-offs associated with each model. This comparison is valuable for making informed decisions about which development model to use for a particular project based on the project's requirements, constraints, and goals.

1. Requirement Specification:

- Relevant because it addresses how each model handles requirement specification at the beginning or throughout the project, which impacts early planning and design decisions.

project costs, especially when comparing cost-efficiency among different models.

4. Availability of Reusable Component:

- Relevant because it addresses the utilization of existing components, libraries, or modules, which impacts development speed, quality, and maintenance.

5. Complexity of System:

- Relevant because it reflects how well a model suits projects of varying complexities, helping choose an appropriate approach based on project scope.

affects the product's alignment with user needs and preferences.

8. Guarantee of Success:

- Relevant because it addresses the level of confidence in achieving project success, guiding decision-making when risk tolerance is a factor.

9. Overlapping Phases:

- Relevant because it indicates whether certain phases overlap, which can impact project efficiency and time-to-market.

10. Implementation Time:

integrated into the project,

influencing the project's ability to respond to

evolving requirements.

13. Expertise Required:

- Relevant because it identifies the expertise

level needed to successfully implement each

model, assisting in resource allocation and skill

assessment.

14. Cost Control and Resource Control:

- Relevant because these factors inform how well

the chosen model manages project

costs and resources.