BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3103

SOFTWARE ENGINEERING

Unit 11

People and Software Engineering

Table of Contents

1. INTRODUCTION

Software engineering is not just about writing code, it involves collaboration, communication, and teamwork to create innovative and reliable software solutions. In this dynamic and ever-evolving field, people are at the heart of the software development process. They bring creativity, problem-solving skills, and expertise to turn ideas into reality.

Software engineering is not only about technical skills but also about effective communication and understanding of user needs. It requires the ability to work in diverse teams, as software development projects often involve professionals from various disciplines. In the modern software engineering landscape, people are not just developers but also problem-solvers, user advocates, and agents of change. They understand the impact of software on society, and ethical considerations are integral to their decision-making process.

Traditional Software Engineering is the conventional method of software development, which adheres to a linear and sequential process, commonly referred to as the Waterfall model. This approach involves breaking the development process into separate phases, such as requirements gathering, design, implementation, testing, and deployment. Each phase is completed before proceeding to the next, and making changes to requirements or design becomes difficult as the process progresses. The primary focuses of Traditional Software Engineering are thorough initial planning, detailed documentation, and strict adherence to predetermined timelines and this has been covered in previous topics.

Modern Engineering, commonly known as Agile Software Development, marks a departure from conventional methods. It encompasses diverse Agile methodologies like Scrum, Kanban, and Extreme Programming (XP). The fundamental principle of modern engineering is to be adaptable, flexible, and customer-focused. Instead of adhering to a rigid and linear approach, modern engineering adopts iterative and incremental development, enabling continuous customer feedback and collaboration.

1.1 Learning Objectives

At the end of the topic, students should be able to understand,

- *Define the core principles of Agile Software Development and explain its emphasis on customer collaboration and incremental development.*
- *Define the core principles of Agile Software Development and explain its emphasis on customer collaboration and incremental development.*
- *Apply Agile principles to create a project plan and backlog for a software development project, considering iterative delivery and customer feedback.*

2. AGILE METHODOLOGIES

Agile Software Development is a methodology that emphasizes flexibility, collaboration, and customer satisfaction in the software development process. Rooted in the Agile Manifesto, it prioritizes values such as individuals and interactions, working software, customer collaboration, and responsiveness to change.

This approach follows an iterative and incremental development process, where software is developed in small increments, allowing for quick and frequent delivery of working products. Agile teams work closely with customers throughout the development cycle to understand and address their specific requirements, ensuring the delivered product meets their needs and expectations. Collaboration and continuous feedback play a vital role in Agile Software Development, enabling teams to adapt to changing requirements and deliver high-quality software in a timely manner.

The Agile Software Development process typically consists of the following steps as shown in Figure 1:

Figure 1: Process of Agile Software Development

a. *Plan:* During the planning phase, the team collaborates with stakeholders to define the project's scope, set objectives, and establish priorities. User stories are created to capture customer requirements and form the basis for the development process.

b. *Design:* In the design phase, the team creates a high-level architectural design and detailed technical specifications. The design ensures that the software is scalable, maintainable, and aligns with the project's goals.
c. *Develop:* The development phase involves writing the code based on the defined requirements and design. Agile development is often done in short iterations or sprints, where the team focuses on delivering a small working increment of the software.
d. *Test:* Testing is a continuous and integral part of Agile Software Development. Throughout the development process, the team conducts various testing activities, including unit testing, integration testing, and user acceptance testing, to ensure the quality and functionality of the software.
e. *Deploy:* During the deployment phase, the working software is released to the production environment or made available to end-users. Agile teams aim to deliver functional increments in a regular and frequent manner to gain rapid feedback from customers.
f. *Review:* After each iteration or release, the team holds a review meeting with stakeholders to gather feedback and assess the progress of the project. The review helps in identifying areas for improvement and determining adjustments needed for future iterations.

2.1 Agile Manifesto

The Agile Manifesto is a set of guiding values and principles for Agile Software Development. It was created in February 2001 by a group of seventeen software developers who came together in the USA. The manifesto outlines a customer-centric and iterative approach to software development, promoting flexibility, collaboration, and responsiveness to change.

The four core values of the Agile Manifesto are:

i. ***Individuals and interactions over processes and tools:*** Here people and their interactions are very important in the software development process. Agile encourages effective communication, collaboration, and teamwork among individuals to make sure the project is a success.

ii. ***Working software over comprehensive documentation:*** Agile believes in delivering a working product that customers and stakeholders can use over having a ton of documentation. Of course, documentation is still important, but having functional software that's demonstrable and validated is the main goal.

iii. ***Customer collaboration over contract negotiation:*** Agile promotes continuous collaboration with customers and end-users throughout the development process. Instead of being tied down by rigid contracts, Agile teams keep the communication lines open and actively involve customers in feedback and requirement discussions. This way, they can create valuable and customer-centric solutions.

iv. ***Responding to change over following a plan:*** Agile acknowledges that change is inevitable in software development. Instead of being afraid of change, Agile embraces it and sees it as a reality. Agile teams are adaptable and quick to respond to changing requirements and priorities, which helps them deliver valuable software even in fast-paced and dynamic environments.

The Twelve principles of the Agile Manifesto are:

i. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

ii. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

iii. Deliver working software frequently, with a preference for shorter timescales.

iv. Business people and developers must work together daily throughout the project.

v. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

vi. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

vii. Working software is the primary measure of progress.

viii. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

ix. Continuous attention to technical excellence and good design enhances agility.

x. Simplicity—the art of maximizing the amount of work not done—is essential.

xi. The best architectures, requirements, and designs emerge from self-organizing teams.

xii. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

3. AGILE FRAMEWORKS

Agile Frameworks are structured methodologies that provide guidelines and practices for implementing Agile principles in software development. These frameworks help development teams organize their work, facilitate collaboration, and deliver high-quality products in an iterative and incremental manner.

Some popular Agile frameworks include ***Scrum, Kanban, and Extreme Programming (XP).***

3.1 Scrum

Scrum is an Agile framework for software development that provides a structured and iterative approach to managing projects. It was originally introduced as a part of the Agile Manifesto in 2001 and has since gained widespread adoption in various industries. Scrum promotes collaboration, transparency, and continuous improvement within development teams.

At its core, Scrum divides the project into fixed-length iterations called "sprints," typically lasting two to four weeks. Each sprint results in a potentially shippable product increment. Scrum teams work in a self-organizing and cross-functional manner, ensuring that the right people with the necessary skills are involved in delivering each increment.

3.1.1 Importance Of Scrum In Agile Methodology:

Scrum plays a crucial role in the Agile methodology due to several key reasons:

a. ***Flexibility and Adaptability:*** Scrum embraces change and allows teams to respond quickly to evolving requirements. This adaptability is vital in today's fast-paced and dynamic business environments.
b. ***Customer-Centric Approach:*** Scrum prioritizes delivering value to customers at every iteration. Frequent deliveries of working software enable stakeholders to provide early feedback and influence the project's direction.
c. ***Transparency and Visibility:*** Scrum promotes transparency by making all work visible on the Scrum board. This visibility ensures that team progress, impediments, and accomplishments are accessible to all stakeholders.

d. ***Continuous Improvement:*** Regular sprint reviews and retrospectives encourage continuous improvement. The team reflects on what went well and what can be enhanced, fostering a culture of learning and growth.
e. ***Effective Time Management:*** The fixed-length sprints in Scrum provide clear time boundaries for work, helping teams manage time effectively and prioritize tasks based on the most critical requirements.
f. ***Early Risk Mitigation:*** By delivering functioning increments at the end of each sprint, potential risks are identified and addressed early in the development process.
g. ***Improved Collaboration:*** The Scrum framework promotes collaboration among team members, stakeholders, and customers, fostering open communication and knowledge sharing.
h. ***Enhanced Team Morale:*** Scrum empowers team members to take ownership of their work and make decisions collectively. This autonomy and sense of responsibility boost team morale and productivity.
i. ***Predictable Delivery:*** With Scrum's incremental approach, teams can plan and predict delivery schedules more accurately, increasing stakeholders' confidence in the project's success.
j. ***Reduced Waste:*** Scrum helps eliminate wasteful activities by focusing on delivering valuable software in a timely manner and avoiding unnecessary work.

3.1.2 Scrum Roles

There are three key roles that play essential roles in the software development process. Each role has distinct responsibilities and interactions within the Scrum framework:

a. Scrum Master
b. Product Owner
c. Development Team

a. Scrum Master:

The Scrum Master is a facilitator and servant-leader for the Scrum team. Their primary responsibility is to ensure that the Scrum framework is understood and followed by all

team members. The Scrum Master removes any impediments or roadblocks that hinder the team's progress and fosters an environment of continuous improvement.

Responsibilities of the Scrum Master:

- Guiding the Scrum team in adopting Scrum practices and principles.
- Coaching and mentoring the team to become more self-organizing and cross-functional.
- Facilitating Scrum events, such as Sprint Planning, Daily Standup, Sprint Review, and Sprint Retrospective.
- Shielding the team from external disruptions to maintain focus during the sprint.
- Promoting collaboration between the Development Team and the Product Owner.
- Resolving conflicts within the team and encouraging healthy communication.
- Tracking and visualizing progress using Scrum artefacts like the Sprint Burndown Chart.
- Assisting the Product Owner in managing the Product Backlog and refining user stories.

b. Product Owner:

The Product Owner is responsible for maximizing the value delivered by the Scrum team. They represent the stakeholders and customers and are the single authority for deciding what features and functionality will be included in the product.

Responsibilities of the Product Owner:

- Creating and maintaining the Product Backlog, which is a prioritized list of features and user stories.
- Collaborating with stakeholders to gather requirements and feedback, ensuring the product meets their needs.
- Defining clear and actionable user stories with acceptance criteria for the Development Team.
- Prioritizing the items in the Product Backlog based on business value and customer needs.

- Providing a clear vision and direction to the Scrum team for each sprint and release.
- Being available to the Development Team during Sprint Planning to clarify requirements and answer questions.
- Participating in Sprint Review meetings to accept or reject completed work based on predefined acceptance criteria.
- Adapting the Product Backlog based on market changes, customer feedback, and new insights.

c. Development Team

The Development Team is a self-organizing and cross-functional group responsible for delivering the potentially shippable product increment at the end of each sprint. They collaborate to design, develop, test, and deliver high-quality software.

Responsibilities of the Development Team:

- Estimating the effort required to complete the selected Product Backlog items during Sprint Planning.
- Organizing and managing their own work to achieve the sprint goal.
- Collaborating with the Product Owner to understand requirements and acceptance criteria.
- Developing the functionality outlined in the user stories and meeting the Definition of Done.
- Ensuring continuous integration and frequent testing to maintain a potentially releasable product increment at the end of each sprint.
- Participating in Daily Standup meetings to provide updates on progress, challenges, and planned work for the day.
- Collaborating with other team members to achieve the sprint goal and deliver value to customers.
- Participating in Sprint Retrospectives to identify improvements in the team's processes and practices.

3.1.3 Scrum Ceremonies

Scrum ceremonies are essential meetings that occur during the Scrum framework to ensure effective collaboration, communication, and continuous improvement within the development team and with stakeholders. *Each ceremony serves **specific objectives** and produces **valuable outcomes**.*

The different Scrum ceremonies are:

a. Sprint Planning
b. Daily Standup
c. Sprint Review
d. Sprint Retrospective

a. Sprint Planning:

Objective: The Sprint Planning meeting kicks off each sprint and aims to define what work the Development Team will undertake during the sprint.

Outcomes:

- The Product Owner presents the highest-priority items from the Product Backlog to the Development Team.
- The Development Team collaborates with the Product Owner to understand the requirements of each item.
- The Development Team selects the appropriate amount of work from the Product Backlog to be completed during the sprint, based on their capacity and velocity.
- The Development Team breaks down the selected items into smaller tasks and creates a Sprint Backlog with their commitment to the upcoming sprint.

b. Daily Standup (Daily Scrum):

Objective: The Daily Standup is a short, time-boxed meeting that takes place every day during the sprint. Its purpose is to facilitate communication and coordination within the Development Team.

Outcomes:

- Each team member provides a brief update on what they worked on since the last Daily Standup, what they plan to work on next, and any obstacles or impediments they are facing.
- The Development Team identifies any potential roadblocks or issues and collaborates to resolve them.
- The Daily Standup helps maintain focus, align the team's efforts, and ensure everyone is aware of progress toward the sprint goal.

c. Sprint Review:

Objective: The Sprint Review occurs at the end of each sprint and aims to inspect the increment of work completed during the sprint and gather feedback from stakeholders.

Outcomes:

- The Development Team demonstrates the completed functionality to the stakeholders, showcasing the potentially shippable product increment.
- The Product Owner discusses the items that were completed and the ones that were not, providing an overview of the sprint's achievements.
- Stakeholders and customers can provide feedback on the delivered functionality, ensuring that the product meets their expectations.
- The Product Backlog may be updated based on the feedback and insights gathered during the Sprint Review.

d. Sprint Retrospective:

Objective: The Sprint Retrospective takes place after the Sprint Review and focuses on continuous improvement of the Scrum process and team dynamics.

Outcomes:

- The Scrum Team reflects on the sprint, discussing what went well and what could be improved.

- The team identifies actionable items for process improvement and agrees on changes to be implemented in the next sprint.
- The Sprint Retrospective helps the team adapt and optimize its work methods, leading to increased productivity and quality.

3.1.4 Scrum Artefacts

The essential Scrum artefacts are key elements that play a crucial role in effective project management within the Scrum framework. They provide transparency, clarity, and accountability throughout the software development process.

Scrum Artefacts are:

- Product Backlog
- Sprint Backlog
- Increment

a. Product Backlog:

The Product Backlog is a prioritized and dynamic list of all the features, enhancements, bug fixes, and other work items required to build and improve the product. It represents the single source of truth for the Development Team and the Product Owner regarding the project's scope and requirements.

b. Sprint Backlog:

The Sprint Backlog is a subset of the Product Backlog and contains the work items selected by the Development Team for a specific sprint. It details the tasks required to fulfil the Sprint Goal.

c. Increment:

The Increment is the sum of all the completed and potentially shippable product backlog items at the end of a sprint. It must meet the team's Definition of Done.

3.2 Kanban

Kanban is an Agile project management methodology that originated from the manufacturing industry, particularly from Toyota's production system. The word "Kanban" in Japanese means "visual signal" or "card," and it represents a visual approach to managing work and workflow.

In the software development context, Kanban provides a flexible and adaptive way to manage projects by visualizing work, limiting work in progress, and optimizing flow.

3.2.1 Principles And Key Concepts Of Kanban:

Kanban is built on several core principles, like:

i. *Visualizing Work:* Kanban emphasizes visualizing the entire workflow on a Kanban board, where work items are represented as cards and moved through different stages of the process. This provides a clear and real-time understanding of the status of work.
ii. *Limiting Work in Progress (WIP):* Kanban imposes WIP limits on each stage of the workflow, preventing the team from taking on more work than they can effectively handle. This ensures that the team stays focused and completes tasks before starting new ones.
iii. *Managing Flow:* Kanban emphasizes optimizing the flow of work by identifying bottlenecks and constraints in the process. This enables teams to improve efficiency and reduce lead times.
iv. *Continuous Improvement:* Kanban promotes a culture of continuous improvement, where teams regularly assess their processes and make incremental changes to enhance productivity and quality.

Key Concepts:

i. *Kanban Board:* The Kanban board is a visual representation of the workflow and is central to the Kanban methodology. It is typically a physical or digital board divided into columns that represent different stages of the process, such as "To Do," "In Progress," and "Done." Work items, represented by cards, are placed in the appropriate

columns as they move through the workflow. The Kanban board provides real-time visibility into the progress of work, making it easy for the team to track and manage tasks. Kanban Board Components are shown in Figure 1.

Work-in-Progress Limits (WIP Limits) restrict the maximum amount of tasks in the different stages of the workflow. Limiting WIP allows you to finish work items faster by helping your team focus only on current tasks.

ii. *Kanban Cards:* Kanban cards are visual representations of individual work items on the Kanban board. Each card contains information about the task, such as a brief description, of the assignee, due dates, and other relevant details. The cards are moved across the board from left to right as the work items progress through the workflow. The physical movement of cards on the Kanban board symbolizes the flow of work through the project stages.

iii. *Columns and Swim lanes:* Columns in the Kanban board represent the different stages of the workflow. Work items flow from one column to the next as they advance in the process. Swim lanes are additional categorization elements in the Kanban board that can be used to group related tasks or work items. Swim lanes provide additional clarity and organization, especially in complex projects with multiple workstreams or teams.

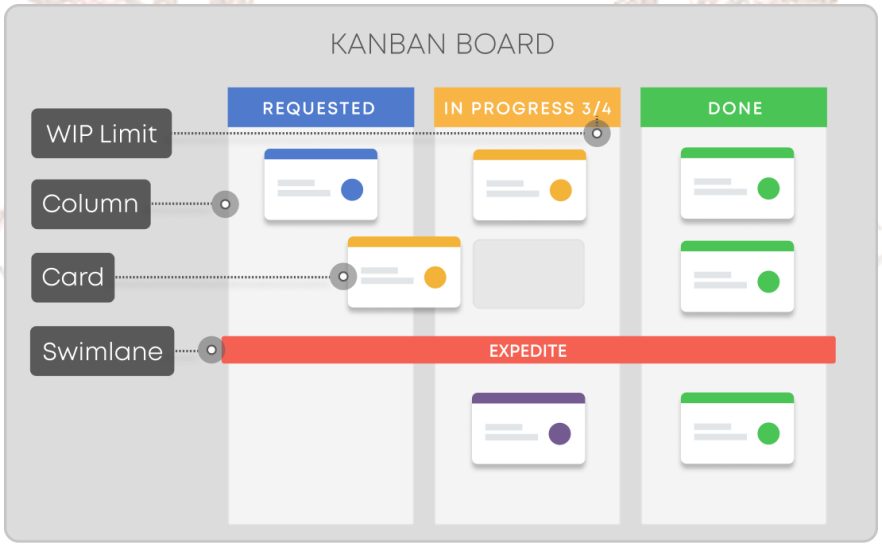

Figure 1: Kanban Board Components

3.2.2 Benefits Of Kanban

Some of the benefits of Kanban are:

- *Enhanced visibility:* Kanban provides a visual representation of the workflow, allowing team members and stakeholders to have a real-time view of work items and their progress. This visibility enhances transparency and makes it easier to track tasks.
- *Increased productivity:* By limiting work in progress and optimizing workflow, Kanban helps teams become more productive. It reduces multitasking and ensures that team members focus on completing tasks before taking on new ones.
- *Greater flexibility:* Kanban is adaptable and can accommodate changes in priorities and requirements. Teams can easily adjust their workflow to handle new tasks or unexpected events.
- *Improved team focus:* With WIP limits and clear priorities, Kanban enables teams to maintain focus on essential tasks. This prevents distractions and ensures that the team works on what matters most.
- *Decreased levels of waste:* Kanban helps identify and eliminate inefficiencies and waste in the workflow. This leads to better resource utilization and reduces unnecessary work.
- *Better collaboration:* Kanban promotes collaboration among team members by encouraging regular standup meetings and fostering a culture of shared responsibility. This collaboration enhances communication and problem-solving.
- *Improved predictability:* Through data and metrics, Kanban helps teams become more predictable in their delivery timelines. This predictability benefits stakeholders and allows for better planning.
- *Alignment with company values:* Kanban principles align with Lean and Agile values, making it a natural fit for organizations embracing these methodologies. It reinforces a culture of continuous improvement and customer focus.

3.2.3 Comparison between Scrum and Kanban

Aspect	Scrum	Kanban
Methodology Type	Iterative and incremental development	Continuous flow
Iterations/Sprints	Fixed-length iterations called sprints	No fixed-length iterations
Planning	Sprint planning for each iteration	Continuous planning and execution
Work in Progress (WIP)	WIP limits typically not used	Strict WIP limits to control flow
Roles	Scrum Master, Product Owner, Development Team	No prescribed roles
Meetings	Daily Scrum, Sprint Review, Sprint Planning, Sprint Retrospective	Daily standup meetings
Adaptability	Changes only considered at sprint boundaries	Changes can be made at any time
Visibility	Clear visibility into sprint progress	Continuous visibility of work on the board
Risk Management	Risk addressed at sprint boundaries	Risks can be addressed continuously
Predictability	Predictable within sprints	Predictable based on flow efficiency

4. SOFTWARE DEVELOPMENT AND TEAM DYNAMICS

Effective team dynamics play a crucial role in driving successful software development projects. A cohesive and collaborative team with strong communication, motivation, and adaptability can significantly enhance the software development process, leading to the delivery of high-quality software that meets user expectations and business needs.

4.1 Software development team roles and Responsibilities

Team roles and responsibilities are defined to ensure that the right skills and expertise are available to complete the project successfully. Each team member plays a specific role with distinct responsibilities that contribute to the overall software development process.

The common software development team roles and their responsibilities:

i. ***Software Developer/Engineer:***
   - Designing, coding, and testing software applications based on project requirements and specifications.
   - Collaborating with other team members to ensure seamless integration of software components.
   - Troubleshooting and debugging software issues to identify and fix defects.
   - Maintaining code quality and following coding standards.
   - articipating in code reviews to ensure best practices and identify potential improvements.

ii. ***Quality Assurance (QA) Engineer/Tester:***
   - Developing test plans and test cases to validate software functionality.
   - Executing manual and automated tests to identify defects and ensure software meets quality standards.
   - Collaborating with developers and stakeholders to reproduce and document software issues.
   - Verifying bug fixes and validating the overall product quality.
   - Monitoring and reporting on software quality metrics.

iii. Product Owner (PO):

- Defining and prioritizing product features and requirements based on customer needs and business value.
- Maintaining and refining the product backlog, ensuring it is up-to-date and reflects project priorities.
- Collaborating with the development team to clarify requirements and acceptance criteria.
- Reviewing and accepting completed work during the Sprint Review.
- Ensuring the product meets customer expectations and aligns with the overall product vision.

iv. Scrum Master:

- Facilitating the Scrum process and ensuring that Scrum practices are followed.
- Removing impediments and roadblocks that hinder the team's progress.
- Coaching and guiding the team in adopting Agile and Scrum principles.
- Facilitating Scrum ceremonies, such as Sprint Planning, Daily Standup, Sprint Review, and Sprint Retrospective.
- Promoting a collaborative and productive team environment.

v. Business Analyst (BA):

- Gathering and analyzing requirements from stakeholders.
- Documenting user stories, use cases, and functional specifications.
- Translating business requirements into technical specifications for the development team.
- Collaborating with the Product Owner and development team to ensure alignment between business needs and software solutions.

vi. UX/UI Designer:

- Creating user interface (UI) designs and wireframes based on user requirements and best design practices.
- Collaborating with stakeholders and developers to ensure design feasibility and usability.

- Conducting user research and usability testing to gather feedback and improve the user experience.
- Delivering design assets and guidelines to the development team for implementation.

vii. DevOps Engineer:

- Managing and maintaining the software development infrastructure and tools.
- Automating build, deployment, and testing processes to ensure efficient and continuous delivery.
- Monitoring and managing application performance and availability.
- Collaborating with development and operations teams to ensure smooth operations and rapid response to issues.

viii. Project Manager:

- Planning and coordinating project activities and timelines.
- Managing project resources, budgets, and risks.
- Tracking project progress and ensuring adherence to project milestones and deadlines.
- Communicating with stakeholders and providing project status updates.
- Identifying and resolving project issues to keep the development on track.

5. ETHICS AND PROFESSIONALISM IN SOFTWARE ENGINEERING

Ethics and professionalism play a vital role in the field of software engineering. Software engineers have a responsibility to uphold ethical standards and maintain professionalism in their work to ensure the safety, integrity, and trustworthiness of the software they develop.

Some ***key aspects of ethics and professionalism*** in software engineering are:

i. ***Client Confidentiality:*** Software engineers must respect client confidentiality and handle sensitive information with the utmost care. They should not disclose any proprietary or confidential information without proper authorization.
ii. ***User Privacy:*** Engineers should prioritize the privacy and security of user data. Collecting, storing, and using user data should comply with relevant laws and regulations, and users should be informed about the data being collected and its purpose.
iii. ***Quality and Integrity of Software:*** Engineers have a professional obligation to deliver high-quality software that performs as expected and is free from defects. They should not engage in any unethical practices that compromise the integrity of the software.
iv. ***Avoiding Plagiarism and Intellectual Property Infringement:*** Engineers should respect intellectual property rights and avoid plagiarism. They should not use copyrighted code, content, or software without proper authorization.
v. ***Informed Consent:*** When using third-party libraries, APIs, or services, engineers should obtain proper consent and adhere to the terms and conditions set by the providers.
vi. ***Compliance with Laws and Regulations:*** Engineers should adhere to all applicable laws and regulations related to software development, including data protection, copyright, and security regulations.
vii. ***Transparency and Honesty:*** Engineers should be transparent about the limitations and risks associated with their software. They should not misrepresent the capabilities of the software to clients or users.
viii. ***Avoiding Conflicts of Interest:*** Engineers should avoid conflicts of interest that may compromise the impartiality of their work or decision-making process.

ix. ***Continuous Professional Development:*** Professionals in software engineering should stay updated with the latest industry developments and best practices to deliver the best possible results.

x. ***Teamwork and Collaboration:*** Professionals should foster a positive and collaborative work environment. They should treat their colleagues with respect and value diverse perspectives.

xi. ***Responsible Software Deployment:*** Engineers should consider the potential impact of their software on society and ensure that it is used responsibly and ethically.

xii. ***Reporting Unethical Behaviour:*** If software engineers encounter unethical behavior or practices within their organization or industry, they should report it to appropriate authorities or follow the established whistleblowing processes.

6. TRADITIONAL SOFTWARE ENGINEERING VS MODERN ENGINEERING

Aspect	Traditional Software Engineering	Modern Engineering
Development Process	Follows a linear and sequential approach, often referred to as the Waterfall model. Each phase (requirements, design, implementation, testing, deployment) is completed before moving on to the next one.	Embraces Agile methodologies such as Scrum, Kanban, etc., which are iterative and incremental. The development process is flexible, with a focus on adaptability, customer collaboration, and continuous delivery.
Project Management	Involves rigid project management with detailed planning, Gantt charts, and fixed schedules. Changes are difficult to accommodate once the project plan is set.	Adopts a more flexible project management approach, allowing for changes in requirements and priorities throughout the project. Agile frameworks emphasize regular feedback and frequent releases.
Requirement Gathering	Requires detailed and extensive upfront requirements gathering before proceeding with development.	Emphasizes ongoing customer collaboration and feedback, allowing requirements to evolve during the development process.
Team Structure	Typically involves large, hierarchical teams with specialized roles.	Encourages small, cross-functional teams that collaborate closely and collectively take ownership of tasks.
Delivery Model	Often delivers the complete software product at the end of the development cycle.	Emphasizes continuous delivery, releasing working software frequently and incrementally.
Testing and Quality Assurance:	Testing is usually conducted at the end of the development process.	Emphasizes continuous testing and test-driven development (TDD) to ensure higher software

		quality throughout the development lifecycle.
Customer Focus	Customer feedback is limited to early requirements gathering and post-implementation reviews.	Embraces a customer-centric approach with continuous customer collaboration throughout the development process.
Documentation	Emphasizes comprehensive and detailed documentation at each phase.	Prioritizes lightweight documentation and focuses on working software as the primary documentation.
Risk Management	Risk management is performed at the beginning of the project, with a focus on risk avoidance.	Emphasizes ongoing risk assessment and mitigation throughout the development process.

7. SUMMARY

Agile is a software development methodology that emphasizes flexibility, customer collaboration, and continuous improvement. It emerged as a response to the limitations of traditional approaches like the Waterfall model. In Agile, software is developed iteratively and incrementally, allowing for frequent releases of working software.

The core principles of Agile are outlined in the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change. Agile methodologies, such as Scrum, Kanban, and Extreme Programming (XP), align with these principles.

In Agile, cross-functional teams work closely with customers and stakeholders throughout the development process. Customer feedback is incorporated regularly, ensuring that the software meets their needs effectively. Agile teams adapt to changing requirements and priorities, delivering value in shorter timescales.

Agile encourages transparency through visual tools like Kanban boards, which help manage work and limit work in progress. Daily standup meetings keep the team aligned, and regular retrospectives promote continuous improvement.

The benefits of Agile include enhanced visibility, increased productivity, better collaboration, improved predictability, and a customer-focused approach. Agile fosters a culture of innovation, adaptability, and continuous learning, making it a popular and effective methodology for software development in today's dynamic and fast-paced environment.

Traditional Software Engineering follows a structured and linear approach with extensive upfront planning, while Modern Engineering embraces flexibility, adaptability, and customer collaboration through iterative development. The choice between these approaches depends on the project's nature, requirements, and the organization's culture and goals. Modern engineering methodologies have become increasingly popular due to their focus on customer satisfaction, continuous delivery, and the ability to adapt to evolving project conditions.

8. SELF-ASSESSMENT QUESTIONS

1. Which of the following is NOT an Agile methodology?
   a) Scrum
   b) Kanban
   c) Waterfall
   d) Extreme Programming (XP)
2. In Scrum, who is responsible for prioritizing the Product Backlog?
   a) Scrum Master
   b) Development Team
   c) Product Owner
   d) Stakeholders
3. Which Agile methodology emphasizes limiting Work in Progress (WIP) to optimize flow?
   a) Scrum
   b) Kanban
   c) Extreme Programming (XP)
   d) Lean Development
4. What does the Kanban board represent?
   a) A tool for tracking project budgets
   b) A visual representation of the team's skills and expertise
   c) A project management software used in Agile development
   d) A visual representation of the workflow and tasks in progress
5. Sprint in Scrum is _____
   a) A meeting with stakeholders to gather requirements
   b) A fixed timebox for software development and testing
   c) A document containing all user stories
   d) A report showing the progress of the development team
6. In Kanban, what does it mean when a task or item reaches its Work in Progress (WIP) limit?
   a) It is removed from the board.

b) It is assigned to a different team member.

c) It is put on hold until a later time.

d) No new tasks can be started until it is completed or moved.

7. What is a primary characteristic of Traditional Software Engineering?
   a) Continuous customer collaboration
   b) Iterative development process
   c) Sequential and linear approach
   d) Emphasis on adaptability
8. Which software engineering methodology emphasizes continuous customer feedback and collaboration throughout the development process?
   a) Waterfall Model
   b) Extreme Programming (XP)
   c) Kanban
   d) Scrum
9. What is the key advantage of Modern Software Engineering over Traditional Software Engineering?
   a) Fixed scope and requirements
   b) Continuous integration and delivery
   c) Rigidity and predictability
   d) Limited customer collaboration
10. Which of the following methodologies embraces an iterative and incremental approach to software development?
    a) Waterfall Model
    b) Traditional Software Engineering
    c) Kanban
    d) Modern Software Engineering

9. SELF-ASSESSMENT ANSWERS

1. c) Waterfall
2. c) Product Owner
3. b) Kanban
4. d) A visual representation of the workflow and tasks in progress
5. b) A fixed timebox for software development and testing
6. d) No new tasks can be started until it is completed or moved.
7. c) Sequential and linear approach
8. d) Scrum
9. b) Continuous integration and delivery
10. d) Modern Software Engineering

10. TERMINAL QUESTIONS

1. Briefly explain the Process of Agile Software Development.
2. Explicate the core values and principles of the Agile Manifesto.
3. Explain the importance of Scrum in Agile Methodology.
4. Elucidate Scrum Roles and Ceremonies.
5. Briefly explain the principles and key concepts of Kanban.
6. Write a short note on Software Development and Team Dynamics.
7. Differentiate traditional Software Engineering and Modern Engineering.
8. Differentiate Scrum with Kanban in Modern Engineering.

11. TERMINAL ANSWERS

1. Refer to Section 11.2.
2. Refer to Section 11.2.1.
3. Refer to Section 11.3.1.1.
4. Refer to Section 11.3.1.3.
5. Refer to Section 11.3.2.1.
6. Refer to Section 11.4
7. Refer to Section 11.6
8. Refer to Section 11.3.2.3