VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vellore – 632014, Tamil Nadu, India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Winter Semester 2023-2024

Digital Assignment – 1

SLOT	**: C2+TC2**
Programme Name	**: B.Tech. CSE**
Course Name & code	**: Software Engineering & BCSE301L**
Class Number	**: VL2023240500732**
Faculty Name	**: Dr. Saurabh Agrawal**

Q.1. In the software process model what is the importance of the "Umbrella activity"? Is that releases of the forthcoming versions are possible without appropriate "Umbrella activity"?

In the realm of software engineering, umbrella activities encompass fundamental tasks that underpin and enhance the entire software development process. Serving as a comprehensive framework, these activities play a crucial role in orchestrating and aligning the various phases and tasks essential for constructing top-notch software solutions.

Intricacies of each umbrella activity, shedding light on their importance and elucidating how they actively contribute to the triumph of software development endeavours:

- Project Management: Project management encompasses the strategic planning, organization, and control of all facets of software development projects. This involves establishing project objectives, formulating schedules, allocating resources, and mitigating risks. Effective project management is pivotal in maintaining the trajectory of the software development process, adhering to timelines, and delivering a high-quality product within the specified budget.

- Requirements Engineering: Requirements engineering revolves around comprehending and delineating the needs and expectations of stakeholders for the software system. This process entails the gathering, analysis, documentation, and validation of requirements. It ensures that the software aligns with user expectations, addresses business needs, and adheres to relevant regulations and standards.

- Software Design and Architecture: Software design and architecture entail the creation of a comprehensive blueprint for the software system. This includes defining the overall structure, component interactions, and relationships within the system. The aim is to ensure that the software is scalable, maintainable, and modular, facilitating efficient development and future enhancements.

- Software Construction: Software construction involves the tangible coding and implementation of the software system. This phase comprises writing and testing code, integrating components, and generating executable software artefacts. The emphasis is on applying best coding practices, adhering to coding standards, and confirming that the software functions as intended.

- Software Testing and Quality Assurance: Software testing and quality assurance activities target the identification of defects, errors, and vulnerabilities within the software system. This encompasses the design of test cases, execution of tests, and validation of software functionality against predefined criteria. Systematic testing allows software engineers to uncover and address bugs, ensuring adherence to quality standards and delivering a reliable user experience.

- Software Maintenance: Software maintenance encompasses activities conducted post-deployment, managing updates, bug fixes, and enhancements throughout the software's lifecycle. This ensures the software remains functional, secure, and aligned with evolving user requirements and technological advancements.

- Configuration and Change Management: Configuration and change management activities involve overseeing software configurations and controlling changes introduced during development. This includes version control, release management, and handling change requests. These activities guarantee the software's stability, consistency, and traceability amid ongoing changes.

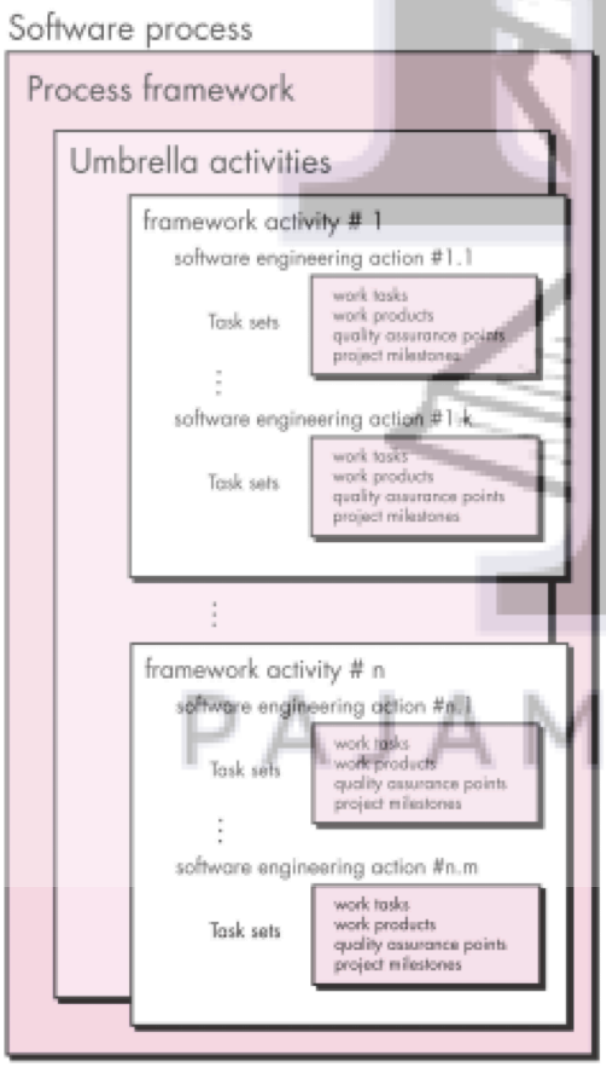
- Risk Management: Risk management centres on identifying potential risks and implementing strategies to mitigate them. This process involves risk analysis, prioritization based on potential impact, and the development of contingency plans. Proactive risk management helps minimize project disruptions, ensures data security, and results in a reliable software solution.

- Documentation: Documentation activities entail creating comprehensive and well-structured documentation for the software system. This includes user manuals, technical specifications, design documents, and system documentation. Proper documentation enhances understanding, promotes collaboration, and facilitates future maintenance and knowledge transfer.

Umbrella activities serve as the foundation of software engineering, offering a systematic framework for software development. Each of these activities is integral in guaranteeing favourable project outcomes and the delivery of top-notch software solutions. Through a comprehensive understanding and adept execution of these activities, software engineers can adeptly navigate the intricacies of the development process, mitigate risks, and successfully deliver valuable products that align with user requirements.

Fig: Umbrella Activities in Software Engineering

UMBRELLA ACTIVITY APPLIED IN THE SOFTWARE PROCESS:



Releases and Umbrella Activity:

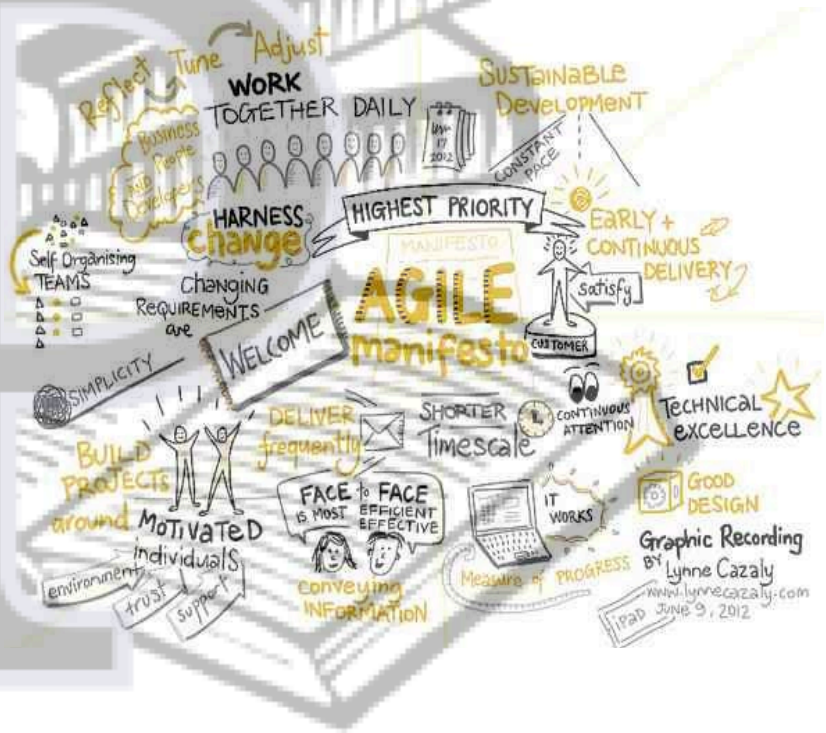
Effective software development relies on a well-structured release process, and the Umbrella Activity is integral to this framework. It encompasses planning, development, testing, and deployment, ensuring a systematic and interconnected approach. Attempting releases without incorporating Umbrella activities can lead to chaotic development, heightened risks, and compromised quality.

The Umbrella Activity serves as a comprehensive strategy, guiding the entire release process. It starts with meticulous planning, where objectives are defined, followed by a structured development phase. Rigorous testing is then conducted to identify and address potential issues before deployment. This approach minimizes errors and enhances overall efficiency.

Without the Umbrella Activity, development teams face increased risks and challenges. Miscommunication, inadequate testing, and rushed deployment become more likely, compromising the quality of the final product. The Umbrella Activity acts as a protective shield, ensuring a clear roadmap for each release and contributing to the successful and high-quality delivery of software products.

Q.2. How agile manifesto are suitable to overcome the weakness of the traditional software models?

The Agile Manifesto, a transformative cornerstone in the field of software development, emerged as a direct response to the perceived deficiencies inherent in traditional software development models, such as the widely employed Waterfall or sequential models. These conventional approaches, once considered standard, revealed inherent limitations that necessitated a paradigm shift. In the realm of Waterfall models, a rigid and sequential process mandated the completion of each development phase before progressing to the next, proving cumbersome when confronted with evolving project requirements. The Agile Manifesto, therefore, sought to inject a much-needed dose of adaptability and flexibility into the software development process, allowing teams to swiftly respond to changing requirements, even in the latter stages of development, thereby fostering a more dynamic and responsive development environment.



Here's an exploration of how the Agile Manifesto effectively addresses the weaknesses of traditional software models:

- Flexibility and Adaptability:

Unlike traditional models that adhere to a rigid, sequential process, Agile embraces changes in requirements, even late in the development process. This flexibility enables teams to promptly respond to customer feedback or evolving market conditions.

- Customer Collaboration over Contract Negotiation:

Agile prioritizes ongoing communication and collaboration with customers, moving away from the heavy reliance on detailed upfront specifications and contracts seen in traditional models. This ensures that the product aligns with evolving customer needs and expectations.

- Working Software over Comprehensive Documentation:

While traditional models often produce extensive documentation at the project's outset, Agile values working software over exhaustive documentation. It promotes a focus on delivering functional and valuable features, with documentation kept to a minimum, emphasizing executable code.

- Responding to Change over Following a Plan:

Traditional models emphasize strict adherence to a detailed plan, making it challenging to adapt to changes or unforeseen challenges. Agile, on the other hand, prioritizes the ability to respond to change, recognizing the need for adjustments as the project progresses.

- Individuals and Interactions over Processes and Tools:

While traditional models may lean towards predefined processes and specific tools, Agile emphasizes the importance of individuals and interactions within a team. It encourages collaboration, communication, and trust among team members over rigid adherence to processes and tools.

- Iterative and Incremental Development:

Traditional models often involve a prolonged development phase culminating in a single, significant release. Agile, however, advocates for iterative and incremental development, with frequent releases of small, functional increments. This approach enables quicker value delivery and continuous improvement throughout the project.

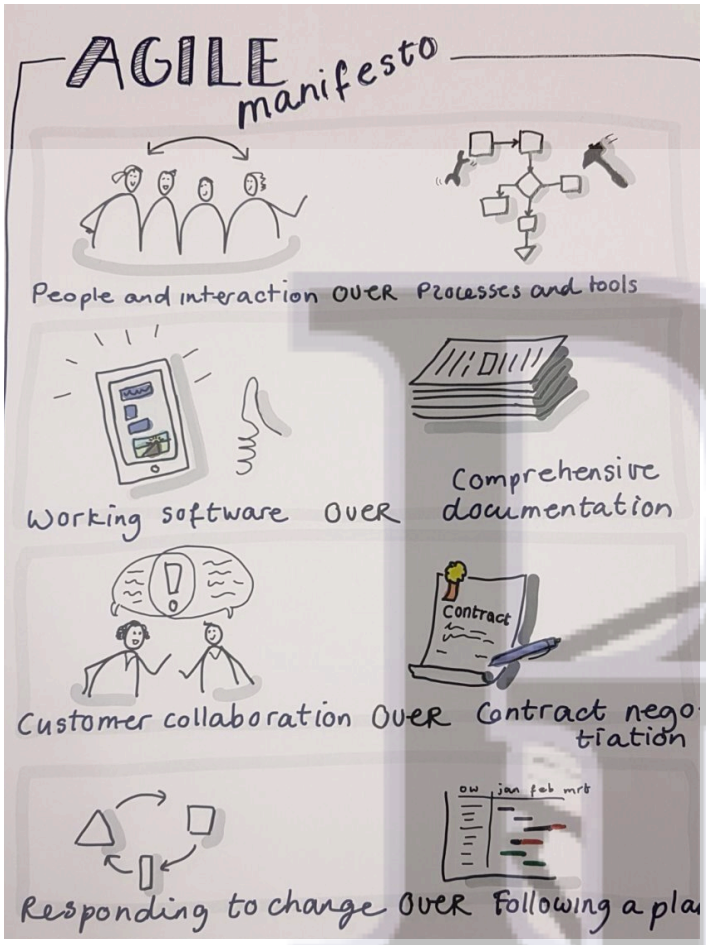
In essence, the Agile Manifesto addresses the shortcomings of traditional software models by championing a more flexible, customer-centric, and adaptive approach to software development. It fosters continuous collaboration, enables swift responses to change, and places a premium on delivering working software to effectively meet customer needs.

Difference between Traditional Software Development and Agile Software Development:

Traditional Software Development	**Agile Software Development**
It is used to develop simple software.	It is used to develop complicated software.
In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
It follows a linear organizational expectation structure.	It follows an iterative organizational structure.
It provides less security.	It provides high security.
Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.
It provides less functionality in the software.	It provides all the functionality needed by the users.
It supports a fixed development model.	It supports a changeable development model.
It is used by freshers.	It is used by professionals.
Development costs are less using this methodology.	Development costs are high using this methodology.

It majorly consists of five phases.	It consists of only three phases.
It is less used by software development firms.	It is normally used by software development firms.
The expectation is favoured in the traditional model.	Adaptability is favoured in the agile methodology.
Traditional software development approaches are formal in terms of communication with customers.	Agile software development methodologies are casual. In other words, customers who work with companies that utilize Agile software development approaches are more likely to interact with them than customers who work with companies that use traditional software development methodology.
For starters, typical software development approaches employ a predictive approach. There is full specification and prediction of the software development processes because the product is produced through rigorous and explicit planning. Changes are not permitted in this technique because the time and cost of project development are fixed.	Here, a flexible approach is used as the software development approaches are founded on the notion of continual design improvement and testing relies on team and client feedback.
Examples: ● Office productivity suites ● Data management software ● Media players ● Security programs	Examples: ● Sky ● Phillips ● JP Morgan Chase
Models based on Traditional Software Development: ● Spiral Model ● Waterfall Model ● V Model ● Incremental Model	Models based on Agile Software Development: ● Scrum ● Extreme Programming (XP) ● Crystal ● Dynamic Systems Development Method (DSDM) ● Feature Driven Development (FDD) ● Adaptive Software Development (ASD)

AGILE MANIFESTO:



12 AGILE PRINCIPLES: