ASDM

Assignment no :1

SDLC

: Sourabh Sanjay Kurabetti

Q1 Discuss the prototyping model. What is the effect of designing a prototype on the overall cost of the project?

Ans- The prototyping model is a software development model in which a prototype or working model of the software is developed before the actual software is developed. The purpose of developing a prototype is to give stakeholders a visual representation of what the final software will look like and how it will function. This allows for feedback and changes to be made before the actual development process begins.

The prototyping model typically consists of four stages: requirements gathering, design, development of the prototype, and prototype evaluation. During the requirements gathering stage, the stakeholders are interviewed to understand their requirements and expectations. In the design stage, the prototype is designed based on the requirements gathered in the previous stage. In the development stage, the prototype is developed and tested, and in the evaluation stage, the stakeholders provide feedback on the prototype, which is then used to refine the design and development of the software.

One effect of designing a prototype on the overall cost of the project is that it can potentially increase the cost in the short term. Developing a prototype takes time and resources, which can increase the upfront costs of the project. However, this cost is often outweighed by the long-term benefits of having a well-designed and well-tested software product. By catching and addressing issues early on in the development process, the cost of fixing these issues later in the development process, or after the product has been released, is reduced. Additionally, by involving stakeholders in the prototype evaluation process, the likelihood of the final product meeting their requirements and expectations is increased, which can result in increased customer satisfaction and a better return on investment for the project.

Q2 Compare iterative enhancement model and evolutionary process model ?

Ans- The iterative enhancement model and the evolutionary process model are two different software development models. Here's how they compare:

Iterative Enhancement Model:

The iterative enhancement model is an iterative and incremental approach to software development. This model is based on the idea of developing a basic version of the software first and then adding more features and functionality in subsequent iterations. Each iteration consists of several phases, including requirements gathering, design, implementation, testing, and deployment. The model allows for changes to be made throughout the development process and enables the team to refine the software as they go.

Evolutionary Process Model:

The evolutionary process model is an iterative and incremental approach that focuses on the evolutionary nature of software development. This model involves multiple iterations of software development, each building on the previous one. The model is flexible and allows for changes to be made throughout the development process. The evolutionary process model emphasizes continuous customer feedback and the development of a working product in each iteration.

Here are some key differences between the two models:

Approach: The iterative enhancement model takes a step-by-step approach to software development, whereas the evolutionary process model takes a more fluid, adaptive approach.

Requirements: The iterative enhancement model assumes that the initial requirements are complete, while the evolutionary process model assumes that requirements will evolve over time.

Customer involvement: The evolutionary process model emphasizes continuous customer feedback, while the iterative enhancement model may involve less customer involvement.

Timeframe: The iterative enhancement model is typically used for shorter projects, while the evolutionary process model is better suited for longer projects.

Testing: The iterative enhancement model involves testing at the end of each iteration, while the evolutionary process model may involve testing throughout the development process.

Overall, both models are iterative and incremental in nature, but they have different approaches and priorities. The choice between the two depends on the specific needs and requirements of the project.

Q3: As we move outward along with process flow path of the spiral model, what can we say about software that is being developed or maintained?

Ans- As we move outward along the process flow path of the spiral model, we can say that the software being developed or maintained becomes more mature and stable. This is because the spiral model is an iterative and incremental software development process, which means that each iteration of the model builds upon the previous one and incorporates feedback from previous iterations.

In the initial stages of the spiral model, the software is in the early development phase and is typically incomplete and unstable. As the process moves outward, the software becomes more refined and stable due to the incorporation of feedback from previous iterations and testing.

At the later stages of the spiral model, the software is in the maintenance phase, which means that it has already been developed and deployed. During this phase, the software is being updated and maintained to ensure that it continues to function as intended and meets the changing needs of its users.

Overall, the spiral model emphasizes risk management, iteration, and continuous feedback, which helps to ensure that the software being developed or maintained is of high quality and meets the needs of its users.

Q4 Explain the Scrum Agile methodology?

Ans- Scrum is a popular agile framework for software development that is used to manage and deliver complex products. It is based on iterative and incremental practices that enable teams to deliver working software at regular intervals. Scrum is known for its ability to increase productivity, improve product quality, and foster teamwork and collaboration.

Scrum methodology is based on a set of roles, events, artifacts, and rules that work together to help teams deliver high-quality software in a flexible and efficient way. The key roles in Scrum are the Product Owner, the Scrum Master, and the Development Team.

The Product Owner is responsible for defining the product vision and prioritizing the product backlog. The Scrum Master is responsible for facilitating the Scrum process and ensuring that the team adheres to the Scrum framework. The Development Team is responsible for delivering the product increment and continuously improving the product.

Scrum events include the Sprint, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. The Sprint is a time-boxed iteration during which the team works on a set of product backlog items. The Sprint Planning is where the team decides what work will be completed during the Sprint. The Daily Scrum is a daily 15-minute meeting where the team members synchronize their work and plan for the day. The Sprint Review is a meeting where the team presents the product increment to stakeholders and gathers feedback. The

Sprint Retrospective is a meeting where the team reflects on the past Sprint and identifies areas for improvement.

Scrum artifacts include the Product Backlog, the Sprint Backlog, and the Increment. The Product Backlog is a prioritized list of product requirements. The Sprint Backlog is a list of items that the team plans to complete during the Sprint. The Increment is the sum of all the completed product backlog items.

Overall, Scrum is a powerful methodology for managing and delivering complex software projects in a flexible and efficient way.

Q5 Explain the utility of Kanban CFD reports?

Ans- Kanban CFD (Cumulative Flow Diagram) reports are useful tools for visualizing and understanding the flow of work in a Kanban system. A CFD chart tracks the number of items in various stages of a process over time, providing insights into the efficiency and effectiveness of the workflow.

Here are some of the ways Kanban CFD reports can be helpful:

Identifying bottlenecks: By analyzing the CFD chart, you can easily identify where work is getting stuck in the process. This can help you to identify bottlenecks and take action to improve the flow of work.

Predicting delivery times: The CFD chart can help you to predict how long it will take to complete a given amount of work. By understanding the average time it takes for work to move through each stage of the process, you can estimate when work will be completed and plan accordingly.

Monitoring progress: The CFD chart provides a clear and concise view of the progress of work over time. This makes it easy to see how much work is in progress, how much is completed, and how much is waiting to be started. This can be helpful in monitoring progress towards goals and identifying areas where improvements can be made.

Identifying trends: By analyzing the CFD chart over time, you can identify trends in the workflow. For example, you may notice that the number of items in a particular stage of the process is consistently increasing, indicating that changes are needed to improve the flow of work.

Overall, Kanban CFD reports provide a powerful tool for understanding and improving the flow of work in a Kanban system. They can help teams to identify bottlenecks, predict delivery times, monitor progress, and identify trends, all of which can lead to more efficient and effective workflows.