ASSIGNMENT 01

NAME:- DEEPRAJ SANTOSHA GHADASHI

EMAIL:- deeprajghadshi1920@gamil.com

Q1.

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

Ans:

The prototyping model is a software development model that involves creating a working model of the software before developing the actual software product. The purpose of this model is to identify and fix any flaws or errors in the software design before the final product is developed.

The prototyping model is a popular choice for software development because it allows developers to gather feedback from users early in the development process, which helps to ensure that the final product meets the user's requirements. It also allows developers to test different design options and explore different solutions to problems without committing to a specific approach.

One of the key benefits of the prototyping model is that it can help to reduce the overall cost of a project. By identifying and fixing design flaws early in the development process, the cost of fixing those flaws later in the development process is reduced. This is because it is much cheaper to fix a problem in the early stages of development than it is to fix the same problem later in the development process or after the software has been deployed.

However, designing a prototype can also increase the overall cost of a project, as it requires additional time and resources to develop the prototype. Additionally, if the prototype is not designed effectively, it may not provide the desired benefits, which could lead to additional costs later in the development process.

In summary, while the prototyping model can help to reduce the overall cost of a software development project, it is important to ensure that the prototype is designed effectively to reap the benefits of the model.

Q2.

Compare iterative enhancement model and evolutionary process model?

Ans:

The comparison between the Iterative Enhancement Model and Evolutionary Process Model

The Iterative Enhancement Model is a linear and sequential development process that involves breaking down a software project into smaller and manageable iterations. Each iteration goes through the phases of requirement gathering, design, development, and testing. Feedback is gathered at the end of each iteration, and the software is enhanced based on that feedback.

The Evolutionary Process Model is an iterative and incremental development process that involves dividing the software development process into multiple evolutionary cycles. Each cycle involves the exploration and evolution of multiple features of the software. Customer involvement is high, and feedback is continuously gathered and used to evolve the software.

Overall, both models have their strengths and weaknesses and can be used in different software development scenarios. The choice of the model depends on the project requirements, customer needs, and other factors.

|  |  |  |
| --- | --- | --- |
| Criteria | Iterative Enhancement Model | Evolutionary Process Model |
| Scope | Fixed scope and goals | Flexible scope and goals |
| Development Phases | Divided into fixed and sequential phases, with each phase building on the previous phase | Overlapping and concurrent development phases, with each phase exploring and evolving multiple features |
| Requirements | Captured upfront | Continuously refined and evolved |
| Feedback | Gathered at the end of each iteration | Gathered continuously throughout the development process |
| Delivery Time | Longer delivery time | Faster delivery time |
| Risk Management | Risk is identified and addressed at each iteration | Risk is addressed in each evolutionary cycle |
| Approach | Enhance and refine the software through successive iterations | Develop the software in small, incremental steps |
| Key features | Incremental development through successive cycles or iterations | Evolutionary development through continuous iterations |
| Development stages | Emphasis on feedback, continuous improvement, and flexibility | Emphasis on constant change, adaptation, and flexibility |
| Customer involvement | Requirements gathering, design, implementation, testing, deployment | Requirements gathering, prototyping, iterative development, testing, deployment |
| Risks and challenges | High involvement throughout the development process | High involvement in the early stages, reduced involvement in later stages |

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:

The Spiral Model is a software development process model that combines elements of both the waterfall model and iterative development. The spiral model consists of a series of iterations called "spirals", with each spiral representing a phase of the software development process. As we move outward along the process flow path of the spiral model, the software that is being developed or maintained is progressively refined and improved.

# 

Specifically, the spiral model includes the following phases:

**Planning:** In this phase, the objectives of the project are defined, the constraints and risks are identified, and the software requirements are specified.

**Risk analysis**: In this phase, the risks associated with the project are analyzed , and strategies are developed to mitigate those risks.

**Engineering:** In this phase, the software is designed, developed, and tested.

**Evaluation**: In this phase, the software is evaluated to determine if it meets the requirements (Feedback) and if it is ready for deployment.

As we move outward along the process flow path of the spiral model, each iteration of the above phases leads to the development of more refined and improved software. In other words, the software is gradually improved and refined as each iteration progresses. This is because each iteration includes feedback from the previous iteration, which is used to improve the software in the next iteration. Therefore, the software that is being developed or maintained becomes more stable, reliable, and feature-rich as we move outward along the process flow path of the spiral model.

Q4.

Explain the Scrum Agile methodology?

Ans:

Scrum is a popular Agile methodology used in software development and project management. It is based on an iterative and incremental approach, where a project is divided into short sprints, usually of two to four weeks. Each sprint focuses on delivering a potentially shippable product increment.

Scrum is organized around three key roles: the Product Owner, the Scrum Master, and the Development Team.

The Product Owner is responsible for defining and prioritizing the product backlog, which is a list of features and requirements that need to be developed. The Product Owner is also responsible for ensuring that the team is delivering value to the customer.

The Scrum Master is responsible for facilitating the Scrum process and ensuring that the team is adhering to the Scrum framework. The Scrum Master acts as a coach to the team, helping them to overcome any obstacles that may arise.

The Development Team is responsible for delivering the product increment at the end of each sprint. The team is self-organizing and cross-functional, meaning that it contains all the skills necessary to complete the work.

During each sprint, the team holds a daily Scrum meeting, where they discuss progress and plan the work for the next 24 hours. At the end of each sprint, the team holds a sprint review meeting, where they present their work to the Product Owner and receive feedback. They also hold a sprint retrospective meeting to discuss what went well and what could be improved in the next sprint.

Scrum is designed to be flexible and adaptable, allowing teams to respond to changing requirements and priorities. It emphasizes communication, collaboration, and transparency, and is well-suited to projects with complex requirements and fast-changing environments.

Q5.

Explain the utility of Kanban CFD reports?

Ans.

Kanban CFD (Cumulative Flow Diagram) reports are a useful tool for tracking the flow of work in a Kanban system. A CFD report provides a visual representation of the progress of work items through the various stages of the workflow over time.

The report is created by plotting the number of work items in each stage of the workflow against time. This provides a clear view of how work items are flowing through the system and how quickly they are being completed.

Kanban it has no fixed planning, and no daily meeting are conducted. In kanban, changes can occur at any time i.e., frequent changes occur.

Kanban does not have the concept of a sprint, so it has no fixed timeline for delivering the product to the client.

It does not estimate the task.

In kanban methodology, we do not have any scrum master. It’s the responsibility of each individual to deliver a valuable product.

The utility of Kanban CFD reports is that they provide several key insights:

**Identify bottlenecks:** CFD reports can help identify bottlenecks in the workflow. If work items are piling up in a particular stage of the workflow, it may indicate that there is a bottleneck in that stage that needs to be addressed.

**Forecasting:** CFD reports can help with forecasting. By tracking the flow of work items over time, it is possible to estimate how long it will take to complete work items in the future.

**Monitor cycle time:** CFD reports provide data on how long it takes for work items to move through the workflow. This data can be used to monitor cycle time and identify areas where improvements can be made.

**Improve workflow:** CFD reports can help teams improve their workflow. By analyzing the data and identifying bottlenecks and inefficiencies, teams can make changes to the workflow to optimize it for efficiency.

Overall, Kanban CFD reports are a valuable tool for teams using a Kanban system. They provide insights into how work is flowing through the system, help identify areas for improvement, and enable teams to optimize their workflow for efficiency.