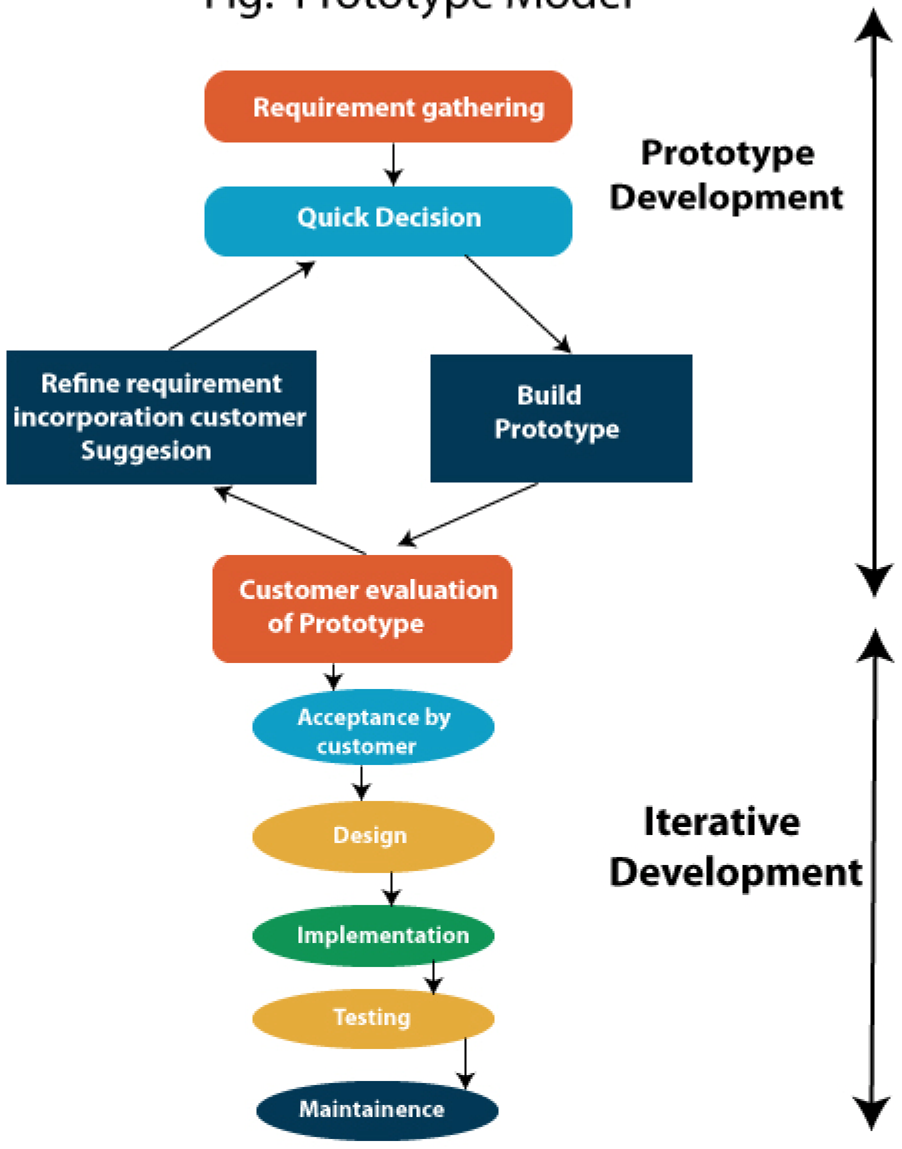
ASSIGNMENT ON SDLC MODEL

1.Discuss the prototyping model.

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

Fig: Prototype Model



Advantages of Prototype Model

- The customer gets to see partial products early in the lifecycle, hence ensuring customer satisfaction.
- The developed prototype can be reused for bigger projects in the future.
- There is scope to accommodate new requirements.
- Errors and missing functionalities can be identified much early in the lifecycle because the users are actively involved.
- User feedback is accommodated quickly.
- The model is very straightforward and does not require skilled experts to implement.

Disadvantages of Prototype Model

- Prototyping is a slow and time taking process.
- Documentation is poor as the requirements change frequently.
- When the customer evaluates the prototype, there may be much too many variances in software needs.
- There is a risk of inadequate requirement analysis owing to too much dependency on the prototype.

Types of Prototype Models in Software Engineering

There are four types of Prototype Models available:

1. Rapid Throwaway Prototyping
2. Evolutionary Prototyping
3. Incremental Prototyping
4. Extreme Prototyping

What is the effect of designing a prototype on the overall cost of the project?

Prototyping may have some initial costs of developing, but it reduces the overall budget by helping your product to be free of the errors or glitches that could have occurred if the idea was made from scratch without any prior user testing. Furthermore, prototyping also helps to understand the intrinsic flaws, shortcomings and drawbacks that can be improved during the product development process. If the prototyping process is ignored completely, it might result in the restructuring and redesigning of the entire product after spending all your resources on its development. So, the effect of designing a prototype on the overall cost of a software project is to actually reduce the additional costs of restructuring and reframing it after its full-fledged development- which might cost a fortune.

2. Compare iterative enhancement model and evolutionary process model.

Iterative Enhancement Model: This model has the similar phases as the waterfall model, but with fewer restrictions. In general the phases occur in the same order as in the waterfall model but these may be conducted in several cycles. A utilizable product is released at the end of the each cycle with each release providing additional functionality.

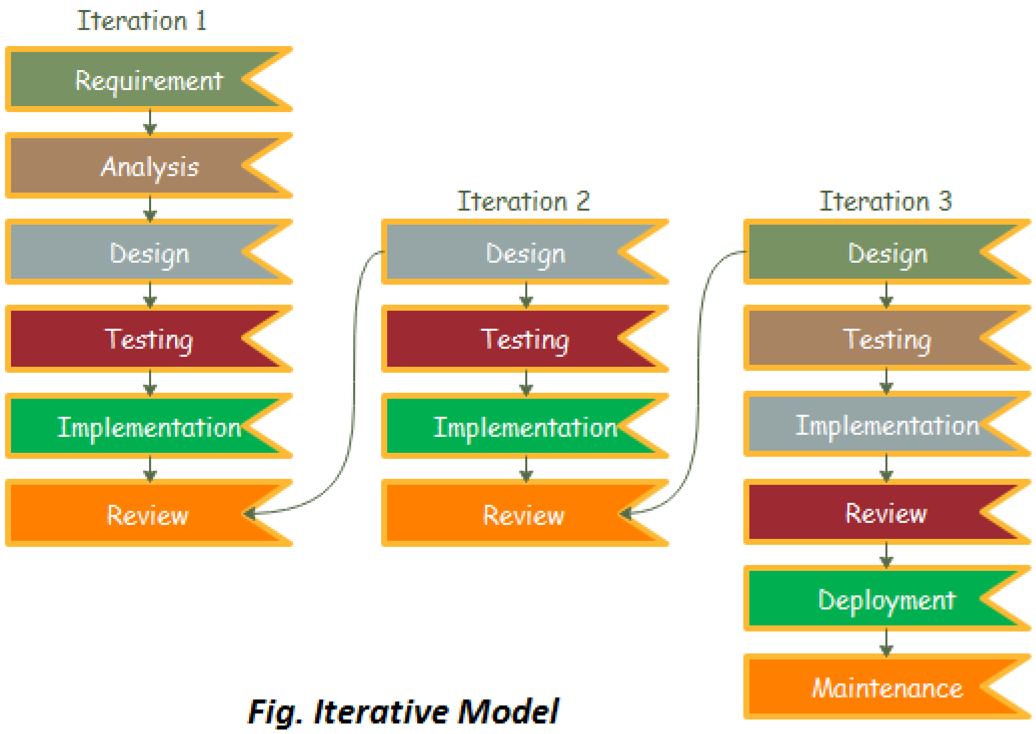


Fig. Iterative Model

Evolutionary Development Model: Evolutionary development model bear a resemblance to iterative enhancement model. The similar phases as defined for the waterfall model occur here in a cyclical fashion. This model is different from iterative enhancement model in the sense that this doesn't require a useable product at the end of each cycle. In evolutionary development requirements are implemented by category rather than by priority.

3. 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.

As we moves outward on the spiral, the product moves toward a more complete state and the level of abstraction at which work is performed is reduced (i.e., implementation specific work accelerates as we move further from the origin).

4.Explain the Scrum Agile methodology.

Scrum is a framework of rules, roles, events, and artifacts used to implement Agile projects. It is an iterative approach, consisting of sprints that typically only last one to four weeks, with the objective of continuously improving a product. This approach is commonly used in software development and ensures that your Scrum teams deliver a version of the product regularly.

Scrum was designed using a software model that follows a set of roles, responsibilities, and meetings. It can be used for any complex project but works best when the result is a concrete product rather than a service. Jeff Sutherland and Ken Schwaber are credited with creating Scrum as a framework for project management.

Scrum in Agile requires particular roles and responsibilities. The Scrum process includes the following steps:

- **Product owner**: The product owner represents the customer's best interest. This person has the ultimate authority over the final product.
- **Scrum master**: This person is a facilitator, responsible for arranging the daily meetings, improving team interactions, and maximizing productivity. The project manager often takes on the role of Scrum master, but they can delegate it to anyone on the team who is a Scrum expert and strong facilitator.
- **Backlog**: The backlog is a prioritized list of tasks and requirements included in the final product. It's the responsibility of the product owner to create the backlog.
- **Sprint**: A sprint is a set timeframe for completing each set of tasks from the backlog. Every sprint should be the same length. Two weeks is typical, but a sprint can be anywhere between one to four weeks long, depending on the team and project needs.
- **Daily meetings**: A Scrum project team is expected to meet every day to discuss progress. These meetings are typically referred to as a daily Scrum or daily stand-up.
- **Review:** This is a meeting where development teams show the work that was completed in an individual sprint and focus on how they can deliver a better product.
- **Retrospective**: In the retrospective meeting, the team reviews their overall system and processes and how they can be improved for the next sprint.

Agile Scrum is the same thing as Scrum. Agile is the overarching methodology, while Scrum is the project management framework that follows the principles of Agile. You could have an Agile team that doesn't use Scrum, but Scrum will always use the Agile methodology. It is sometimes referred to as Scrum methodology, but Scrum is an Agile framework that helps teams collaborate and deliver a final product.

As mentioned, Agile Scrum is common in software development, where teams require an iterative and incremental approach that focuses on continuous improvement. It's important that teams engage in sprint planning, where they plan the work to be completed in future sprints. Other key Agile Scrum activities include daily stand-ups, sprint reviews, and sprint retrospectives.

Agile Scrum teams are cross-functional, with different team members offering different areas of expertise. The Scrum master's role is to lead the team members, ensure they follow the principles of the Scrum framework, and eliminate potential roadblocks that may impede progress.

Within each sprint, the development team completes tasks outlined in the product backlog with the overall goal of creating an improved version of the product, often with additional features and fewer bugs. This process is overseen by the product owner, who ensures that team members are working on the right tasks according to priority.

In summary, Agile Scrum is a widely used framework that values adaptability, team collaboration, and a drive to pursue better outcomes. While it is often associated with software development, it is also suitable for other industries that need to retain a sense of flexibility in their workflows, such as marketing operations and event planning.

5.Explain the utility of Kanban CFD reports.

A cumulative flow diagram (CFD) is an advanced analytic tool in the Kanban method. It provides teams with a visualization of workflow efforts and overall project progress. The cumulative flow diagram allows teams to monitor how stable their workflow is, anticipate bottlenecks so they can adjust their workflow accordingly, and help make processes more predictable.

By graphing how tasks accumulate over time and their overall distribution across the process stages, cumulative flow diagrams visualize massive amounts of data from which you can gather quantitative and qualitative insights into past and present problems with your workflow stability—and where you should focus on making your processes more efficient.

Cumulative flow diagrams collect every task that has gone through your workflow to visualize three critical metrics:

- **Cycle time:** This is the total time it takes your team to complete each task from the beginning to the end. One of the benefits of CFDs is that you can see where you can optimize your workflow to reduce cycle times.
- **Work in progress:** This is the number of tasks your team is actively handling at a certain time. Cumulative flow diagrams will visualize inefficiencies in your project timeline when your team has too much or too little work in progress at any given point.
- **Throughput:** This is the number of tasks your team can complete over a given period. As this is the ultimate measure of your team's productivity, cumulative flow diagrams should

show where you can align your efforts and resources so that throughput increases over time.