Unit-02

System development process model

What is SDLC?

→ SDLC stands for Software Development Life Cycle. It's a process used by software developers to design, develop, test, and deploy high-quality software. The stages typically include planning, analysis, design, implementation, testing, deployment, and maintenance. It provides a structured approach to software development, ensuring that the final product meets customer requirements and is delivered on time and within budget.

Major component of SDLC

1. **Planning**

→ Defining the project scope, goals, and requirements. Establishing a roadmap for the development process.

2. **Analysis**

→ Gathering and understanding user requirements. Analyzing the existing system and identifying areas for improvement.

3. **Design**

→ Creating a detailed blueprint for the system based on the requirements. This involves architectural, data, interface, and procedural design.

4. **Implementation**

→ Coding or building the system based on the design specifications. This is the phase where the actual development of the software takes place.

5. **Testing**

→ Systematically evaluating the software to ensure it meets the specified requirements. This includes unit testing, integration testing, and system testing.

6. **Deployment**

→ Introducing the software into the operational environment. This involves installing the system, training users, and transitioning from the old system to the new one.

7. **Maintenance**

→ Ongoing support, troubleshooting, and updates to ensure the system continues to meet user needs. This phase addresses issues that arise post-deployment and may involve making enhancements or fixing bugs.

Concept of software development process

→ Software project management is accomplished using processes such as initiating, planning, executing, controlling, and closing. Software project management is the art and science of planning and leading software projects. It is a subdiscipline of project management in which software projects are planned, implemented, monitored and controlled. Software project management comprises of several activities, which contains -:

- planning of project

- deciding scope of software product
- estimation of cost in various terms
- scheduling of tasks and events
- resource management

Concept of SDLC life cycle.

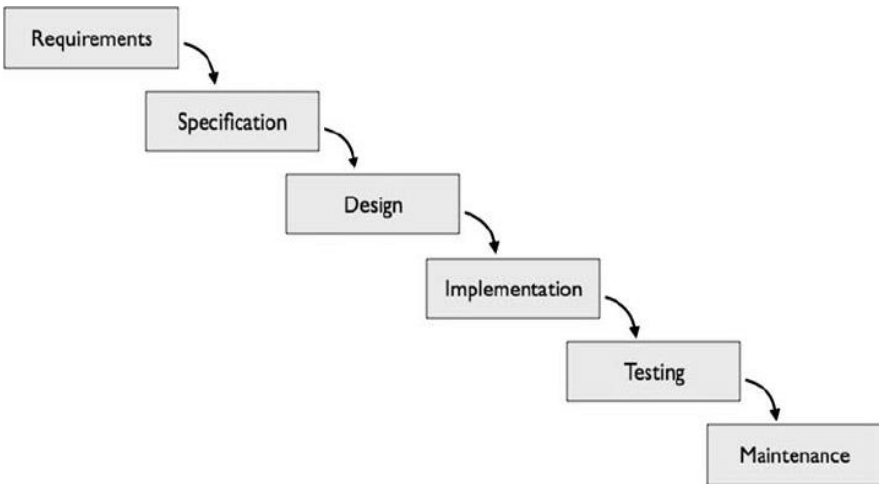
→ SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the predefined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase. SDLC stands for Software Development Life Cycle and is also referred to as the

Software development Model

→ There are different types of patterns and structure by which software/ system can develop. This pattern is known as software development Model. There are different types of models via software/ system can development. Based on software/ system size, nature, budget, timeframe software development company can implement different types of models. Their models are as follows.

Waterfall Model

→ The Waterfall Model is a traditional and linear approach to software development that is often used in system analysis and design. It was one of the earliest methodologies for software development and is characterized by a sequential, phased approach, where progress is seen as flowing steadily downwards (like a waterfall) through several distinct phases. Each phase must be completed before the next one begins, and there is little room for iteration or going back to a previous phase. The different phases in the Waterfall Model are:



1. **Requirements Gathering and Analysis:**

→ In this initial phase, the project team collaborates with stakeholders to gather and document all requirements for the system. It involves a comprehensive understanding of end-users' needs and the expectations of stakeholders.

2. **System Design:**

→ Based on the gathered requirements, this phase involves creating the system architecture and design. The system is broken down into smaller components, and the interaction between these components is defined. This phase lays the groundwork for the development process.

3. **Implementation:**

→ The actual coding or programming of the system occurs in this phase. The design specifications are translated into a working system. It is a crucial stage where the software is developed based on the established design.

4. **Testing:**

→ The system undergoes rigorous testing to identify and address any defects or issues. Testing is critical to ensuring that the system meets the specified requirements and functions correctly. It includes unit testing, integration testing, and system testing.

5. **Deployment:**

→ Once the system has been thoroughly tested and approved, it is deployed for use by end-users. This phase involves the installation of the system in the target environment, making it available for users to access and utilize.

6. **Maintenance:**

→ Ongoing maintenance and support activities are performed in this final phase. The system is monitored, and any issues that arise after deployment are addressed. Maintenance ensures the system's continued functionality and may involve updates, bug fixes, and enhancements based on user feedback or changing requirements.

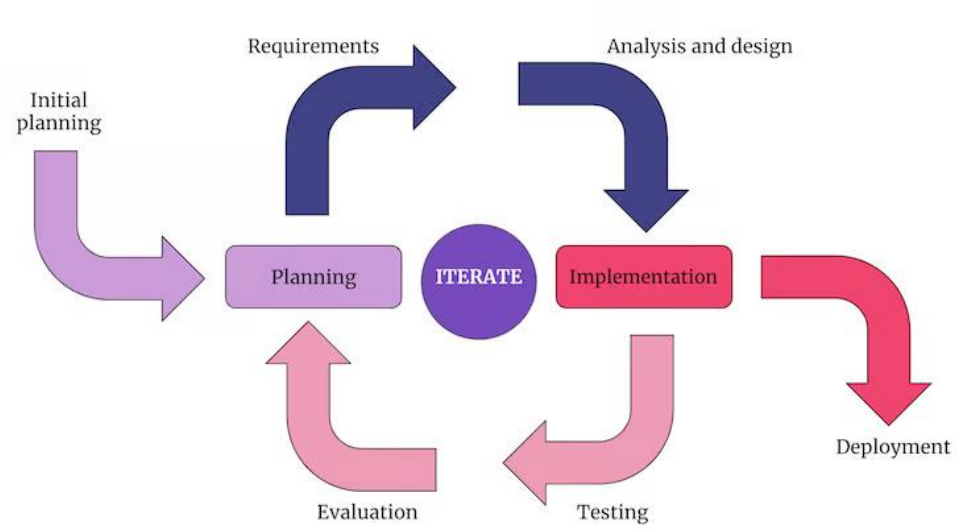
The Waterfall Model is straightforward and easy to understand, making it suitable for projects with well-defined and stable requirements. However, it has limitations, especially in dynamic and changing environments, as it doesn't accommodate changes easily once the project is underway.

Advantages:

- It is simple and suitable for small-sized projects.
- It is less expensive Disadvantages:
- It has no back track mechanism.
- It is not suitable for large size project. It lacks proper documentation.

Iterative and Incremental Development Model

Iterative and Incremental Development is a software development model that breaks down the development process into smaller, manageable parts, allowing for repetitive cycles of development and refinement. The different phases in the **Iterative and Incremental Model** are:



1. **Requirements**

→ Gather and document the initial set of requirements. These may be refined and expanded in subsequent iterations.

2. **Planning:**

→ Plan the development tasks for the current iteration based on the prioritized requirements. This involves determining the scope, schedule, and resources needed.

3. **Development:**

→ Implement the features and functionalities outlined in the current iteration plan. This phase results in a partial but functional version of the overall system.

4. **Testing:**

→ Conduct testing on the developed features to ensure they meet the specified requirements. Testing occurs throughout the development process and not just at the end.

5. **Feedback and Evaluation:**

→ Collect feedback from stakeholders, including end-users, based on the tested features. Evaluate the iteration's success and identify areas for improvement.

6. **Incremental Development:**

→ Incorporate the feedback into the next iteration. Build upon the existing codebase by adding new features or refining existing ones. Each iteration adds incremental improvements to the overall system.

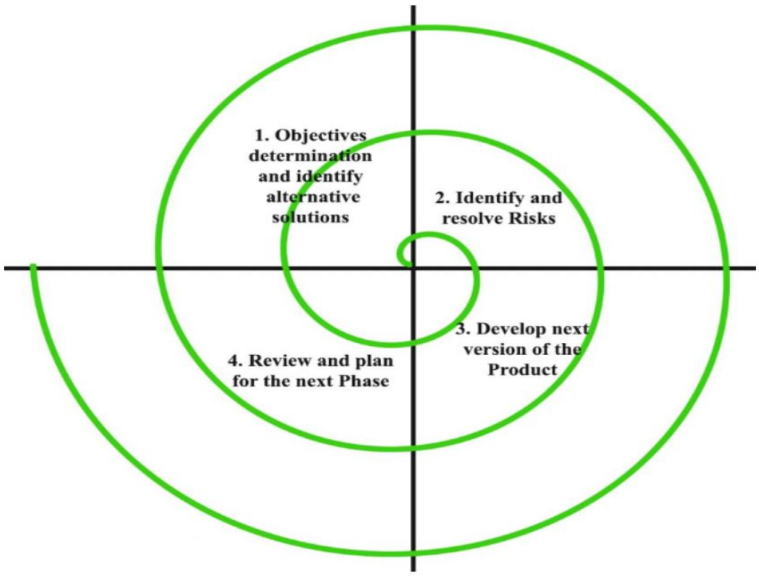
7. **Repetition:**

→ Repeat the process with each iteration, gradually evolving the software through successive cycles of development, testing, and refinement.

The Iterative and Incremental Development model allows for flexibility, adaptability to changing requirements, and the delivery of a functioning product at the end of each iteration. This approach is often associated with Agile methodologies.

Spiral model

→ The Spiral Development Strategy and Model is an iterative approach in software development that emphasizes risk management and accommodates changes in requirements. It is represented as a spiral, with each loop representing a phase in the development process. It involves repeating the cycle of planning, risk analysis, Engineering, Evaluation, Deployment, and Maintenance, making it adaptable to changing requirement in large and complex project. The different phases in the Spiral model are:



1. **Planning:**

→ In the planning phase, the project team defines the project's objectives, constraints, and potential alternative solutions. This involves understanding the scope of the project, identifying key deliverables, and establishing a comprehensive plan that outlines the tasks and activities to be carried out throughout the development process.

2. **Risk Analysis:**

→ Risk analysis is a critical phase where potential risks and uncertainties associated with the project are systematically identified, assessed, and documented. The goal is to understand and quantify the potential impact of these risks on the project's success. Strategies are then developed to mitigate and manage the identified risks effectively.

3. **Engineering (Development and Testing):**

→ The engineering phase involves the actual development of the software. It begins with building a small prototype or an initial version of the software based on the project plans and requirements. This phase encompasses coding, testing, and iterative refinement of the software to ensure that it meets the specified objectives.

4. **Evaluation:**

→ In the evaluation phase, the results of the engineering activities are reviewed. The progress made in developing the software is assessed, and the quality of the product is evaluated. Based on this assessment, the project team decides whether to proceed to the next iteration of the spiral. If the evaluation is satisfactory, the development process continues; otherwise, adjustments are made.

5. **Deployment:**

→ During the deployment phase, the tested and evaluated portion of the software is released to users. This involves delivering a working increment of the product for actual use. Deployment marks a significant milestone as users can start benefiting from the delivered functionality.

6. **Maintenance:**

→ The maintenance phase is ongoing and involves addressing issues, making enhancements, and providing support for the deployed system. This ensures that the software remains effective and relevant over time. Maintenance activities may include bug fixes, updates, and the incorporation of new features based on user feedback and evolving requirements.

The process then repeats in a series of cycles, allowing for refinement and adjustment based on feedback and changing requirements. The Spiral Model is particularly useful for complex projects with uncertain or changing requirements, providing a systematic approach to manage risks and ensure adaptability.

Agile Development Strategy and Model

→ The Agile Development Strategy and Model is an iterative and collaborative approach to software development that prioritizes flexibility, customer feedback, and continuous improvement. The different phases in the Agile Model are:

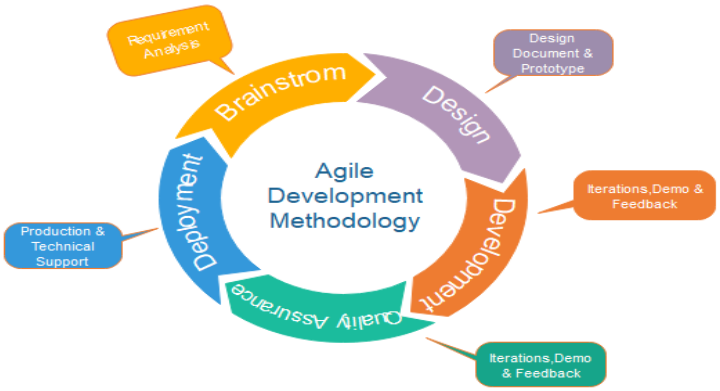


Fig. Agile Model

1. **User Stories and Backlog:**

→ Define features and functionalities through user stories, creating a prioritized product backlog based on customer needs. This phase focuses on understanding and capturing user requirements.

2. **Sprints:**

→ Break the development process into fixed-length iterations, known as sprints, usually lasting 2-4 weeks. Sprints provide a structured timeframe for focused development efforts.

3. **Sprint Planning:**

→ Plan upcoming sprint work by selecting user stories from the backlog based on priority and capacity. Sprint planning ensures a clear scope for the iteration.

4. **Development:**

→ Implement features outlined in selected user stories during the sprint. Development is a collaborative effort, with teams working to deliver functional increments of the product.

5. **Testing:**

→ Conduct testing throughout the sprint to ensure that developed features meet quality standards. Testing is an integral part of each sprint, promoting continuous quality assurance.

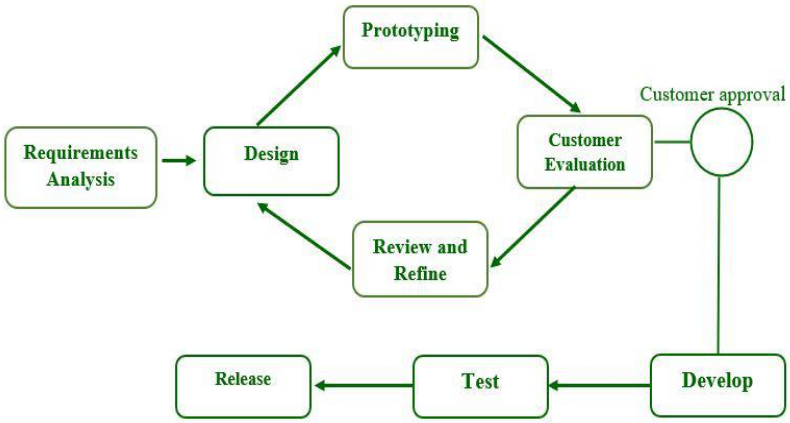
6. **Incremental Development:**

→ Repeat the entire process with each new sprint, incorporating feedback and building on the existing product incrementally. This iterative approach allows for continuous improvement and adaptability to changing requirements.

The Agile model emphasizes customer collaboration, adaptability to changing requirements, and the delivery of a working product at the end of each sprint. Popular Agile frameworks include Scrum and Kanban, providing structures and practices to implement Agile principles effectively.

Prototype model

The Prototype Model is an iterative software development approach where a prototype (an early approximation of a final system or product) is built, tested, and refined in successive cycles based on feedback from users. The different phases in the Prototype model are:



1. **Requirements Gathering:**

→ Initial requirements are collected from stakeholders. These requirements may be broad or detailed, depending on the understanding of the project at the outset.

2. **Quick Design:**

→ A basic, quick design of the system is created based on the gathered requirements. This design is used as a foundation for building the prototype.

3. **Prototype Development:**

→ A working prototype is developed using the quick design. This prototype is a functional representation of the final system but may not include all the features.

4. **User Evaluation:**

→ The prototype is presented to users for evaluation and feedback. Users interact with the prototype to identify strengths, weaknesses, and areas for improvement.

5. **Refinement:**

→ Based on user feedback, the prototype is refined and enhanced. This may involve adding features, improving existing functionalities, or addressing any issues identified during evaluation.

6. **Iterations:**

→ Steps 3-5 are repeated in multiple iterations. Each iteration results in an improved version of the prototype, incorporating feedback from users.

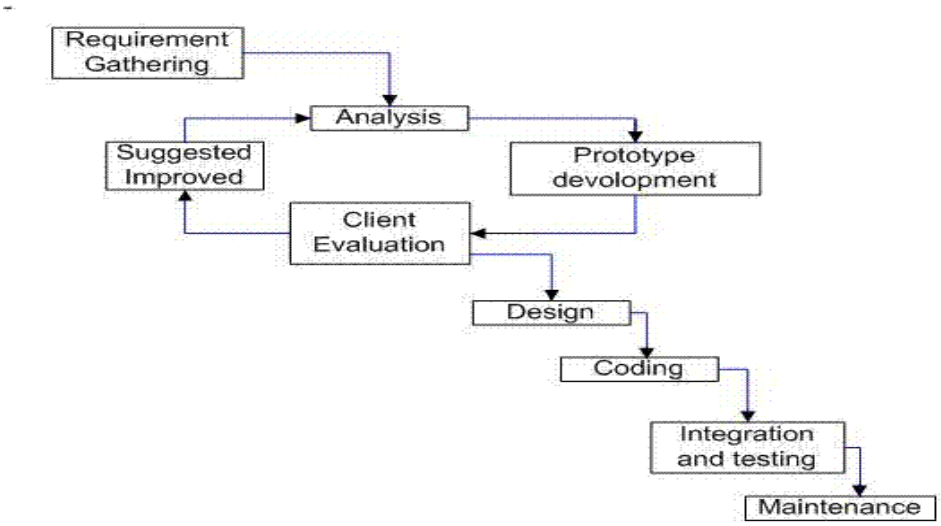
7. **Final Implementation:**

→ Once the prototype meets user expectations and requirements, the final system is implemented using the insights gained from the iterative prototyping process.

The Prototype Model is beneficial when requirements are not well-understood initially, allowing users to interact with a tangible representation of the system early in the development process. It facilitates better communication between developers and users, reduces the risk of misunderstandings, and supports a more accurate understanding of user needs. However, it may not be suitable for all projects, particularly those with well-defined requirements or strict budget and time constraints.

Evolutionary Model

→ Evolutionary Development is an approach to software development that emphasizes continuous refinement of the system through incremental iterations. It allows for the evolution of the software based on changing requirements and feedback. The different phases in the Evolutionary Model are:



Evolutionary Prototyping Model

1. **Requirements Identification:**

→ Gather and document initial high-level requirements for the system. This phase may involve discussions with stakeholders to understand their needs.

2. **Prototyping:**

→ Develop a basic prototype or a partial version of the system based on the identified requirements. The prototype serves as a tangible representation of the system's functionalities.

3. **Feedback and Iteration:**

→ Collect feedback from users and stakeholders by presenting the prototype. Use this feedback to refine and enhance the system. This phase involves a loop of prototyping, feedback collection, and iteration.

4. **Incremental Development:**

→ Develop the system incrementally, adding new features and functionalities in successive iterations. Each iteration builds upon the previous one, incorporating lessons learned and feedback.

5. **Continuous Refinement:**

→ Continuously refine and improve the software based on ongoing user feedback and changing requirements. This phase emphasizes adaptability and responsiveness to evolving needs.

6. **Testing:**

→ Conduct testing throughout the development process to ensure the evolving system meets quality standards. Testing is an integral part of each iteration to identify and address issues early.

7. **Deployment:**

→ Release the system for use once a satisfactory level of functionality has been achieved. Deployment includes incorporating all the refined features developed during the evolutionary process.

8. **Maintenance and Further Evolution:**

→ Provide ongoing maintenance and support for the deployed system. Simultaneously, allow for further evolution of the system based on user needs, market changes, or new requirements that emerge over time.

The Evolutionary Development model is particularly suitable for projects where requirements are expected to change or are not well-defined initially. It emphasizes adaptability, collaboration with users, and a continuous improvement approach to software development.