2...

Software Process and Life Cycle Models

Learning Objectives...

- To learn the concept of SDLC.
- To study about Prescriptive Process Models in detail.

2.1 INTRODUCTION

- A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.
- In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

2.1.1 Need of SDLC

- The development team must determine a suitable life cycle model for a particular plan and then observe to it.
- Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, it would point to chaos and project failure. This problem can be defined by using an example.

- Suppose a software development issue is divided into various parts and the parts are assigned to the team members. From then on, suppose the team representative is allowed the freedom to develop the roles assigned to them in whatever way they like. It is possible that one representative might start writing the code for his part, another might choose to prepare the test documents first, and some other engineer might begin with the design phase of the roles assigned to him. This would be one of the perfect methods for project failure.
- A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

2.2 ACTIVITIES OF SDLC

2.2.1 SDLC life-Cycle Phases

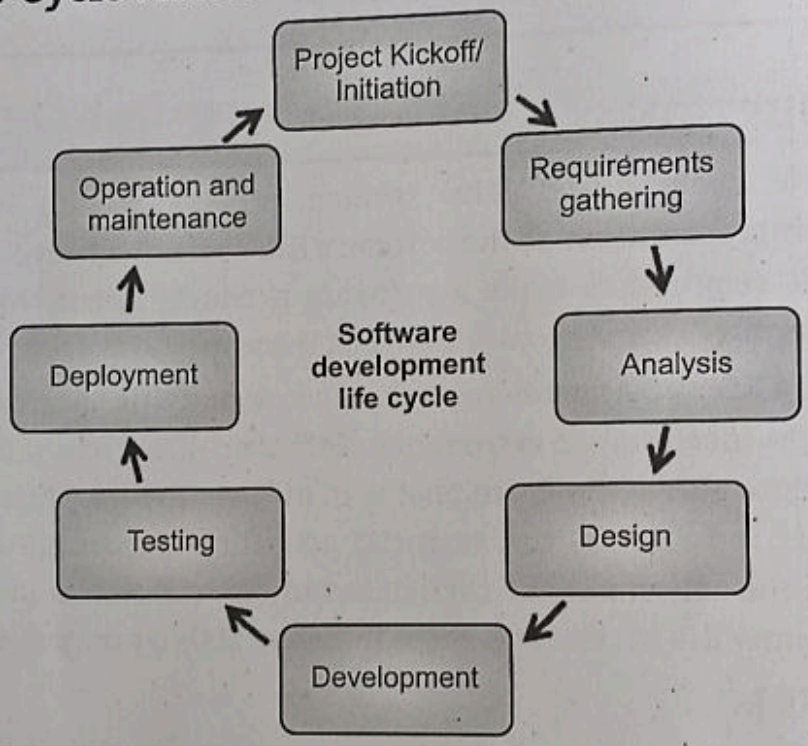

Fig. 2.1: Software Development Life Cycle

1. **Requirement Gathering (Preliminary Investigation):**
   - Business prerequisites are acknowledged in this phase.
   - Meetings with supervisors, partners and customers are held to decide the prerequisites like:
     - Who is going to use the software application?
     - How is the software application going to be used?
     - What information is the software going to process?

- These are general inquiries that get replied during the prerequisites acknowledgement stage.
- A Requirement Specification document is then created to serve the purpose of guideline for the next phase of the cycle.
- It usually consists of the following:
  - Functional Requirement Specification
  - Business Requirement Specification
  - Client/Customer Requirement Specification
  - User Requirement Specification
  - Business Design Document
  - Business Document

2. Requirement Analysis:

- The stage of requirements analysis is the most important in SDLC. It is usually performed by the senior members of the software development team along with marketing and industry experts. This is the crucial part of the project where the software development team leadership must understand the essence of the software to be developed, the specifics of the business case and the potential positioning of the software being developed against the products of competitors, (assuming they exist in the market).
- The outcome of this stage, usually, is the SRS document, which stands for Software Requirements Specification.
- At this stage, the identification of the development risks as well as the quality assurance methodology baseline is decided upon.
- The outcome of the analysis stage is to define the technical solutions which will lead to the success of the project with minimum risk.

3. Design:

- In our simplified model, we assume that after Stage 1, the requirements analysis, the leader of the project writes down the decisions that were taken and develops the so-called Software Requirement Specification (SRS) document. The SRS is the reference guide for the software architects to deliver the best possible result for the given software.
- The outcome of the work of the software architect(s) is the System Design Specification (SDS). This document sometimes is also called the Design Document Specification (DDS). The best scenario is that the SDS is reviewed by the important stakeholders of the project from different outlooks, such as: risk assessment, product robustness, design modularity, budget and time constraints. The best design model is then discussed and selected for the product.

- The design of the software clearly outlines the architectural modules of the product as well as data flow and communication diagrams within the product itself, in addition, it specifies any third-party integration that is relevant to the product.

4. **Coding:**
   - The complexity of this stage heavily depends on the outcome of the previous two stages.
   - The better the SRS and SDS, the easier it is for the software engineers to develop the required software modules. It is no secret that if the first prototypes/versions of the software are required within a very short timescale, then this can adversely affect the quality of the final software. Quality also depends heavily on the analysis capabilities of each individual taking part in the coding process, as does having a properly prepared SRS and SDS.
   - The more experience and brain-power the analysts, architects and developers have, the less time and documentation is needed in the design phase.
5. **Testing and Integration:**
   - Although this stage is called testing, in real life, the situation is that faults found in the software in the testing phase, lead back to the development phase and then back to the testing phase in circles, until the software finally reaches the necessary quality. The testing phase normally consists of two internal phases:
     - Automated unit/functional tests.
     - Acceptance tests performed by a human.
6. **Deployment:**
   - In this phase, the fully developed and tested software is deployed to the production environment and made accessible to end users. This involves setting up infrastructure, installing the application, and ensuring everything runs smoothly. Users are given access, and support materials or training may be provided. Post-deployment monitoring is carried out to detect any issues and gather feedback.
7. **Maintenance:**
   - Ignoring the technical side of this phase, this is when the software is released to the alpha/beta or stable state and feedback starts to come in. Analysis of the feedback can then lead to second, third or fourth iterations of the SDLC which means, the process is continued again with phases 1, 2, 3, 4 and 5, gradually eliminating any issues that have been found in the released software.
   - The whole software development process can also be planned over several iterations, especially, when it comes to Agile, where the main focus is to release a working product as soon as possible and implement different features later on.
   - So, it doesn't mean that going over several iterations always means bug-fixing or tuning. As we just mentioned — it can also be a pre-planned process.

2.2.2 Advantages of SDLC

1. **Provides Clear Project Guidelines and Goals:** SDLC offers a structured approach to software development by clearly defining each phase of the process. This helps teams understand their roles, responsibilities, and deliverables, ensuring everyone is aligned with the project's objectives from start to finish.
2. **Enhances Project Planning and Control:** By breaking down the development process into well-defined stages, SDLC allows for better planning of timelines, budgets, and milestones. It also facilitates tracking progress and managing changes efficiently throughout the project lifecycle.
3. **Improves Resource Management:** SDLC helps in identifying and allocating the right resources–such as personnel, tools, and technologies at the right time. This ensures optimal use of available resources, minimizes wastage, and avoids bottlenecks.
4. **Ensures Quality and Customer Satisfaction:** With built-in phases for validation and verification (like testing and reviews), SDLC emphasizes quality at every step. Meeting user requirements and ensuring the final product performs as expected leads to higher customer satisfaction.
5. **Reduces Project Risks and Cost Overruns:** The structured nature of SDLC allows for early detection of issues, better risk assessment, and timely corrective actions. This minimizes the chances of project failure, delays, and unexpected costs, making development more predictable and cost-effective.

2.3 TYPES OF SDLC PROCESS MODEL

- Software Processes is a coherent set of activities for specifying, designing, implementing and testing software systems. A software process model is an abstract representation of a process that presents a description of a process from some particular perspective. There are many different software processes but all involve:
  - **Specification:** Defining what the system should do.
  - **Design and implementation:** Defining the organization of the system and implementing the system.
  - **Validation:** Checking that it does what the customer wants.
  - **Evolution:** Changing the system in response to changing customer needs.
- SDLC models are generally categorized into two types:

(i) **Descriptive Model:** These describe what happens during software development without prescribing specific processes. They're useful for documentation, understanding current practices, and improvement.

(ii) **Prescriptive Model:** These models define how software should be developed. They offer structured approaches with rules, roles, and deliverables to follow. Most formal SDLC models fall into this category.

2.4 PRESCRIPTIVE PROCESS MODELS

- Prescriptive models are structured, process-driven approaches to software development. They guide developers with specific phases and roles.

2.4.1 Waterfall Model

- The Waterfall Model is one of the earliest and most traditional software development models. It follows a linear and sequential approach, where each phase must be completed before moving on to the next. It is called "waterfall" because the process flows steadily downwards, like a waterfall, through the phases of the SDLC.

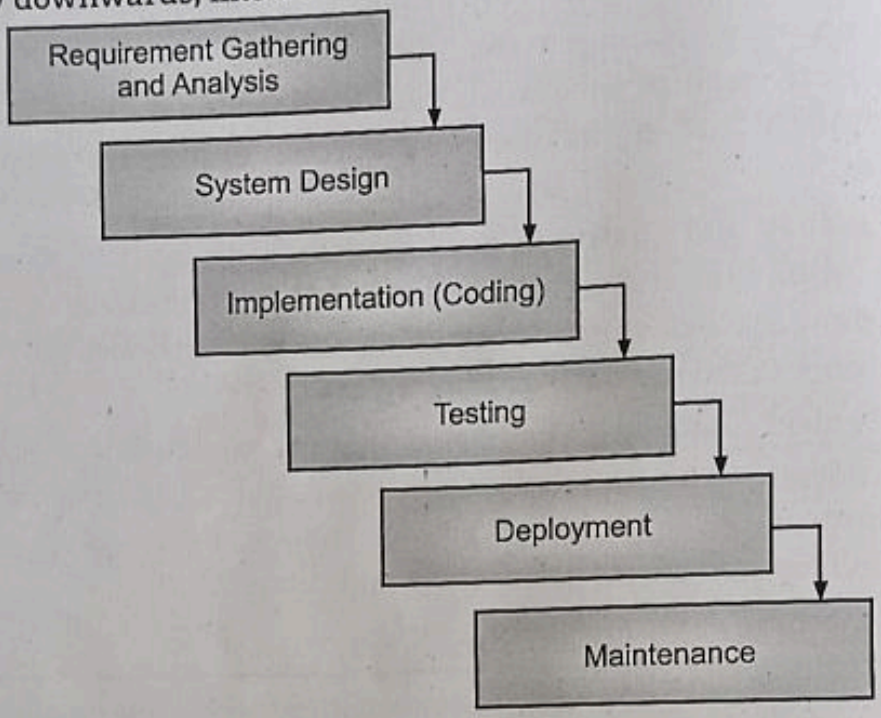

Fig. 2.2: The Water Model

Phases of the Waterfall Model:

1. **Requirement Gathering and Analysis:** All software requirements are collected and documented. These are expected to be complete and clearly understood from the start.
2. **System Design:** Based on the requirements, system architecture and design specifications are created to guide development.
3. **Implementation (Coding):** Developers write code according to the design documents, usually in a structured and module-wise manner.
4. **Testing:** The completed software is tested for defects, functionality, and compliance with the requirements.
5. **Deployment:** Once testing is complete and successful, the software is deployed into the production environment.
6. **Maintenance:** After deployment, the system may need bug fixes, updates, or enhancements, which are handled in this phase.

Advantages of the Waterfall Model:

1. Simple to understand and use.
2. Well-suited for small projects with clear, fixed requirements.
3. Each phase has specific deliverables and a review process.
4. Easier to manage due to its rigid structure.

Disadvantages of the Waterfall Model:

1. Not flexible to changes in requirements.
2. Difficult to go back to previous phases.
3. Late discovery of issues or misunderstandings.
4. Not ideal for complex or long-term projects with evolving needs.

2.4.2 Incremental Process Model

- The Incremental Process Model is a software development approach where the system is built and delivered in small, manageable sections or increments. Each increment adds a portion of the system's functionality and is developed through all the typical SDLC phases from requirements to deployment.

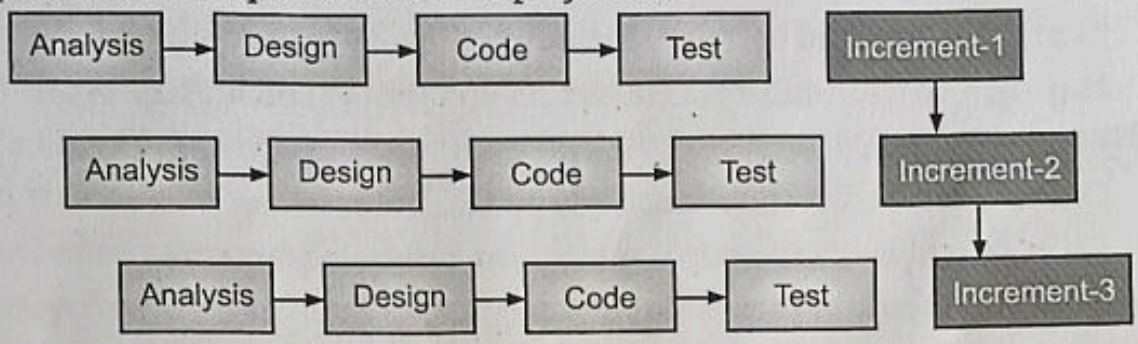

Fig. 2.3: Incremental Model

Key Characteristics:

1. Software is developed and delivered in parts (increments).
2. Each increment is a fully functional subset of the final system.
3. Earlier increments are used and tested while later ones are being developed.
4. Feedback from users can be incorporated in later stages

Phases of the Incremental Process Model:

- The Incremental Process Model breaks down software development into multiple increments, where each increment goes through a complete development cycle. Over time, each new increment adds functionality until the full system is complete.

1. **Requirement Analysis for the Increment:**
   - Identify and analyze requirements for the current increment.
   - The overall system requirements are usually divided into smaller parts.
   - Only the requirements for the current increment are fully detailed.

2. **System Design:**
   - Design the system architecture for the increment.
   - Consider integration with previous increments.
   - Focus on both new functionality and how it will interact with existing components.
3. **Implementation (Coding):**
   - Code the new features or modules defined in the current increment.
   - Use modular or component-based programming to ease integration.
4. **Testing:**
   - Conduct unit testing, integration testing, and system testing.
   - Ensure the new increment works and does not affect previously developed increments.
5. **Integration:**
   - Integrate the new increment with earlier increments.
   - Test the overall system to confirm seamless functionality.
6. **Deployment:**
   - Deploy the updated software including the new increment to users.
   - May involve training, documentation updates, or rollout strategies.
7. **Maintenance and Feedback:**
   - Gather user feedback on the newly added features.
   - Fix bugs and prepare improvements for future increments.

Advantages of Incremental Process Model:

1. Delivers working software early and frequently.
2. Easier to test and debug smaller modules.
3. Allows for customer feedback during development.
4. Lower initial delivery cost and faster releases.
5. More flexible to changes than the waterfall model.

Disadvantages:

1. Requires good planning and design up front.
2. Integration of increments can be complex.
3. Not ideal if requirements are not well understood.
4. May lead to inconsistent design if not properly managed.

2.4.3 Evolutionary Process Models

- Evolutionary process models focus on developing software through repeated cycles (iterations) and incremental refinement. They are well-suited for projects where requirements are not fully understood upfront or likely to change over time. These

models emphasize iteration and continuous user feedback. Two popular evolutionary models are Prototyping and the Spiral Model.

(a) Prototyping:

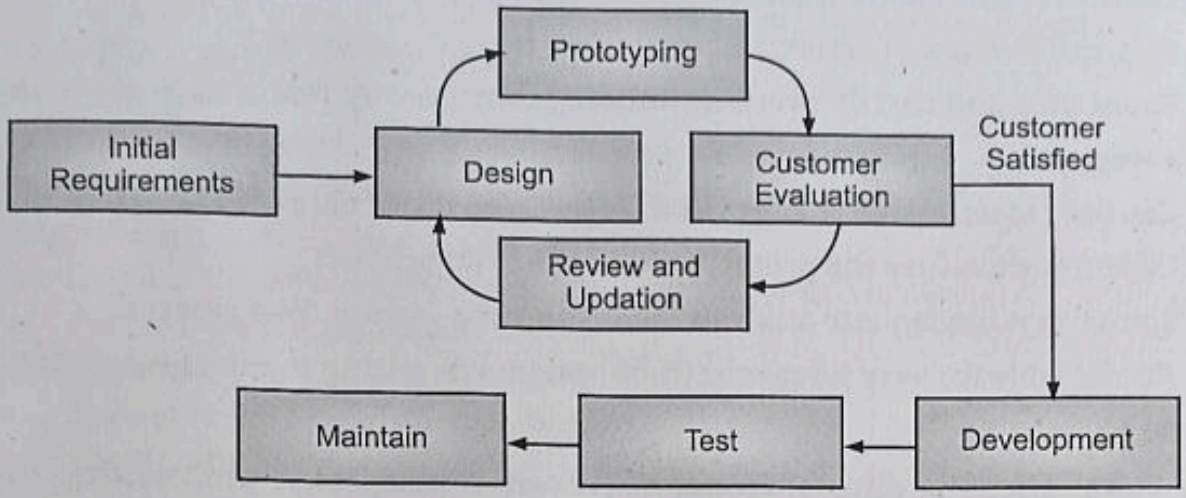

Fig. 2.4: Prototyping

Overview:

- Involves building a quick, simplified version of the software called a prototype.
- The prototype is developed early to help users understand requirements better.
- Users interact with the prototype and provide feedback.
- Based on feedback, the prototype is refined through multiple iterations until the final system is developed.

Key Characteristics:

- Helps clarify and refine user requirements.
- Reduces misunderstanding between users and developers.
- Speeds up the feedback loop.
- Can save time and cost by avoiding building the wrong product.

How it Works?

- **Requirement Gathering:** Basic requirements are collected from users, focusing on key features.
- **Quick Design:** A simple design is created to build the prototype.
- **Build Prototype:** A working model of the system is developed rapidly, often with limited functionality.
- **User Evaluation:** The prototype is presented to users for feedback. They interact with it and suggest changes or improvements.
- **Refinement:** Based on user feedback, the prototype is revised and improved through multiple iterations.
- **Final System Development:** Once the prototype meets user expectations, the final system is developed using a more formal process.

Advantages:

1. Helps users and developers better understand unclear or complex requirements.
2. Early user involvement increases satisfaction and reduces misunderstandings.
3. Allows detection of errors and omissions early in the project.
4. Saves time and cost by avoiding building unnecessary features.

Disadvantages:

1. Can lead to excessive changes and scope creep if not managed properly.
2. Users may confuse the prototype with the final product.
3. Sometimes inadequate analysis leads to a poor-quality final system.
4. Not suitable for very large or critical systems requiring rigorous validation.

(b) Spiral Model:

- The Spiral Model is an evolutionary software development process that combines iterative development with systematic risk management. It emphasizes repeated refinement through cycles (called spirals), allowing for continuous improvement and risk assessment throughout the project.

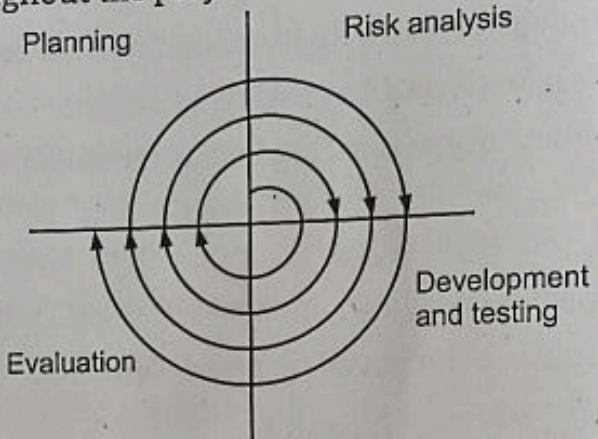

Fig. 2.5: Spiral Model

Overview:

- Combines elements of both design and prototyping in stages.
- Focuses heavily on risk assessment and mitigation.
- Development is divided into spirals (cycles), each consisting of planning, risk analysis, engineering, and evaluation.
- Each spiral results in a prototype or partial product, progressively refined until the final system is complete.

Key Characteristics:

1. Highly iterative and flexible.
2. Emphasizes early identification and resolution of risks.
3. Suitable for large, complex, and high-risk projects.
4. Allows for incremental releases with user feedback.

How it Works:

- Each spiral cycle consists of four main activities:
  1. **Planning:**
     - Define objectives, alternatives, and constraints for the current phase.
     - Gather requirements and set goals.
  2. **Risk Analysis**
     - Identify potential risks (technical, financial, schedule).
     - Analyze and develop strategies to mitigate these risks.
  3. **Engineering (Development and Testing):**
     - Develop the software product or prototype for this cycle.
     - Perform testing and validation.
  4. **Evaluation**
     - Review the results with stakeholders.
     - Decide whether to proceed to the next spiral or make changes.

Advantages:

1. Effective handling of risks reduces chances of project failure.
2. Flexibility to incorporate changes at any point.
3. Continuous user feedback improves product quality.
4. Provides early prototypes, improving requirement understanding.

Disadvantages:

1. Can be expensive and complex to manage.
2. Requires expertise in risk assessment.
3. Not ideal for small projects with well-defined requirements.

Limitations:

1. Can be expensive and complex to manage.
2. Requires expertise in risk analysis.
3. Not ideal for small projects with well-understood requirements.

(c) V-V Model:

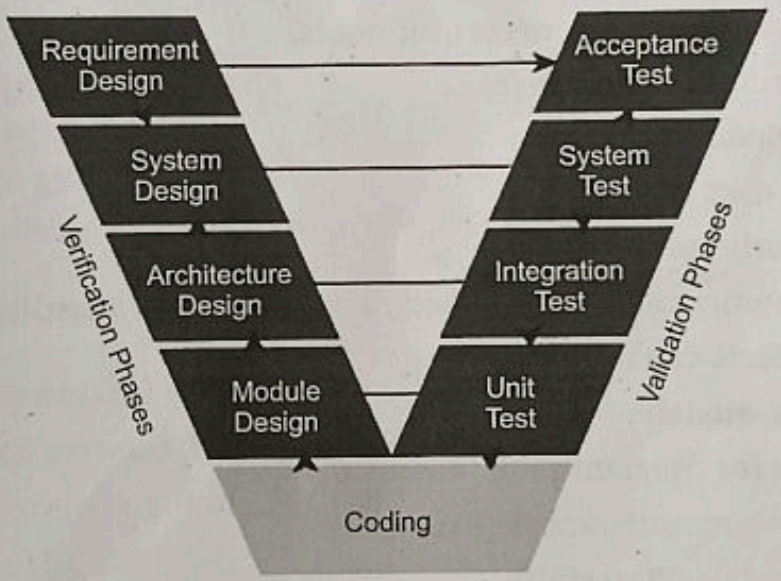

Fig. 2.6: V-V Model

- The V-Model is divided into two main sides:

(i) Left Side – Verification (Development Phases):

1. **Requirements Analysis:**
   - **Corresponds to:** Acceptance Testing
2. **System Design:**
   - **Corresponds to:** System Testing
3. **Architectural Design (High-Level Design):**
   - **Corresponds to:** Integration Testing
4. **Module Design (Low-Level Design):**
   - **Corresponds to:** Unit Testing.
5. **Coding (Implementation Phase)**
   - The base of the V (bridge between design and testing).

(ii) Right Side – Validation (Testing Phases):

1. **Unit Testing:**
   - Tests individual modules.
   - Based on module design.
2. **Integration Testing:**
   - Tests interaction between modules.
   - Based on architectural design.
3. **System Testing:**
   - Tests the whole system functionality.
   - Based on system design.
4. **Acceptance Testing:**
   - Confirms the system meets user needs.
   - Based on user requirements.

Advantages of V-Model

1. Simple and easy to use.
2. Clear and early test planning.
3. Each development activity is directly associated with a testing activity.
4. Reduces defects due to early testing.

Disadvantages of V-Model:

1. Not suitable for frequent requirement changes.
2. No early working software (unlike Agile).
3. High rigidity (like Waterfall).

2.4.4 Concurrent Models

- Activities in different phases occur simultaneously.
- Allows parallel development and testing.
- Suitable for dynamic and large projects.
- Promotes flexibility and fast adaptation to changes.

How it Works?

- The process is divided into several subsystems or workflows, each undergoing its own life cycle.
- These workflows run concurrently but synchronize periodically to integrate components.
- Feedback loops allow continuous updates and refinements throughout development.

Advantages:

1. Reduces time to market by overlapping phases.
2. Increases flexibility to accommodate changes at any stage.
3. Allows early detection of errors due to parallel testing.
4. Improves resource utilization by enabling multitasking.

Disadvantages:

1. Requires strong coordination and communication among teams.
2. Complex to manage due to parallel activities.
3. Potential integration issues if synchronization is not carefully planned.

Check Your Understanding

1. What does SDLC stand for?
   (a) Software Data Life Cycle
   (b) System Development Life Cycle
   (c) Software Design Life Cycle
   (d) System Design Life Cycle
2. Which of the following is not a phase in the SDLC?
   (a) Planning (b) Implementation
   (c) Wedding (d) Testing
3. What is one major advantage of using the SDLC process?
   (a) Structured and systematic development
   (b) No documentation required
   (c) Random development
   (d) Skipping testing

4. Which of the following is a descriptive model?
   (a) Waterfall Model
   (b) Spiral Model
   (c) Agile Model
   (d) All of the above
5. Prescriptive models are focused on:
   (a) Guidelines and steps to follow
   (b) Only describing the problem
   (c) Hardware configurations
   (d) Post-maintenance issues
6. The Waterfall model follows which kind of flow?
   (a) Non-linear
   (b) Iterative
   (c) Sequential
   (d) Random
7. The Incremental model is best described as:
   (a) All modules delivered at once
   (b) Building the system piece by piece
   (c) No coding required
   (d) Entire system deployed without testing
8. In Prototyping model, the key focus is on:
   (a) Final product at once
   (b) Ignoring user feedback
   (c) Full documentation before coding
   (d) Developing an early version for feedback
9. The Spiral model combines elements of:
   (a) Agile and RAD
   (b) Waterfall and Prototyping
   (c) Incremental and Throwaway
   (d) Scrum and Kanban
10. The Concurrent Development Model is most useful when:
   (a) All processes must be completed one after another
   (b) No overlap of processes is allowed
   (c) Activities occur in parallel
   (d) Waterfall model is used

Answers

1. (b)	2. (c)	3. (a)	4. (d)	5. (a)	6. (c)	7. (b)	8. (d)	9. (b)	10. (c

Practice Questions

Q.I Answer the following questions in short.

1. What are the major activities carried out during the SDLC?
2. What are the benefits of using SDLC?
3. Define prescriptive process model.
4. What is a descriptive process model?
5. In what kind of projects is the Waterfall model suitable?
6. What is prototyping in software development?
7. What are the types of prototypes used in SDLC?
8. Define Verification and Validation.

Q.II Answer the following questions in detail:

1. What is SDLC? Explain its purpose.
2. Define Software Development Life Cycle and its significance in software engineering.
3. List and explain the various phases of SDLC.
4. Describe each phase of the SDLC with a neat diagram.
5. How does SDLC improve software quality and project management?
6. Explain with an example how descriptive models help in understanding software processes.
7. Explain with a diagram.
   (a) Waterfall model
   (b) Spiral model
   (c) Prototyping model
   (d) V-V model
   (e) Incremental model
   (f) Concurrent Development model
8. What are the advantages and disadvantages of the
   (a) Waterfall model
   (b) Spiral model
   (c) Incremental process model

9. Compare
   (a) Descriptive and Prescriptive process models
   (b) Prototyping and Waterfall models
   (c) Incremental model with the Waterfall model.
   (d) Waterfall and Spiral models.
10. Explain the prototyping model and its use in software development.
11. Discuss the advantages of the Spiral model over the Waterfall model.
12. How does the Spiral model address risk in software development?
13. How does the Concurrent model differ from other SDLC models?
14. Explain the need for concurrent development in modern software engineering.
15. Write short notes on:
    (a) Prototyping model
    (b) Concurrent model

***