

SOFTWARE DEVELOPMENT LIFE CYCLE

Table of Contents

1. Introduction:	3
1.1. Purpose:	3
1.2. System Overview:	4
2. Phases of SDLC:	4
2.1. Phase-1: Plan	4
2.2. Phase-2: Design	5
2.3. Phase-3: Implementation (Coding)	5
2.4. Phase-4: Testing	5
2.5. Phase-5: Deployment	5
2.6. Phase- 6: Maintenance	5
3. SDLC Models:	6
3.1. Waterfall Model:	6
3.2. RAD Model:	8
3.3. V-Model	9
3.4. Spiral Model:	11
3.5. Big Bang Model	12

List of Figures

Figure 1: The six phases of SDLC	4
Figure 2: Waterfall Model	7
Figure 3: RAD Model	8
Figure 4: V-model flow diagram	10

1. Introduction:

SDLC Definition – The software development life cycle is the process of **Planning, Writing, Modifying, and Maintaining software**.SDLC began as the “systems development lifecycle” in the 1960s.

Software Development Life Cycle process (SDLC) is a systematic and structured approach to developing software applications. It outlines the series of steps and activities that software development teams follow to design, create, test, deploy, and maintain software products.

The primary goal of the SDLC is to ensure that the software is developed efficiently, with high quality, and in a predictable manner.

The SDLC serves as a roadmap that guides software developers, project managers, and stakeholders throughout the development process, helping them work together cohesively and achieve project objectives effectively. It provides a framework to manage risks, track progress, and maintain transparency throughout the development life cycle.

1.1. Purpose:

Software Development Life Cycle is the structured process that enables the production of high quality, low-cost possibility, low-cost software, in the shortest possible production time.

The goal of the SDLC is to produce superior software that meets and exceeds all the customer expectations and demands.

1.2. System Overview:

Software Development life Cycle is a process that produces the software with the highest quality and lowest cost in the shortest time possible.

2. Phases of SDLC:

The **SDLC model involves six phases or stages** while developing any software. SDLC is a collection of these six stages, and the stages of SDLC are as follows:

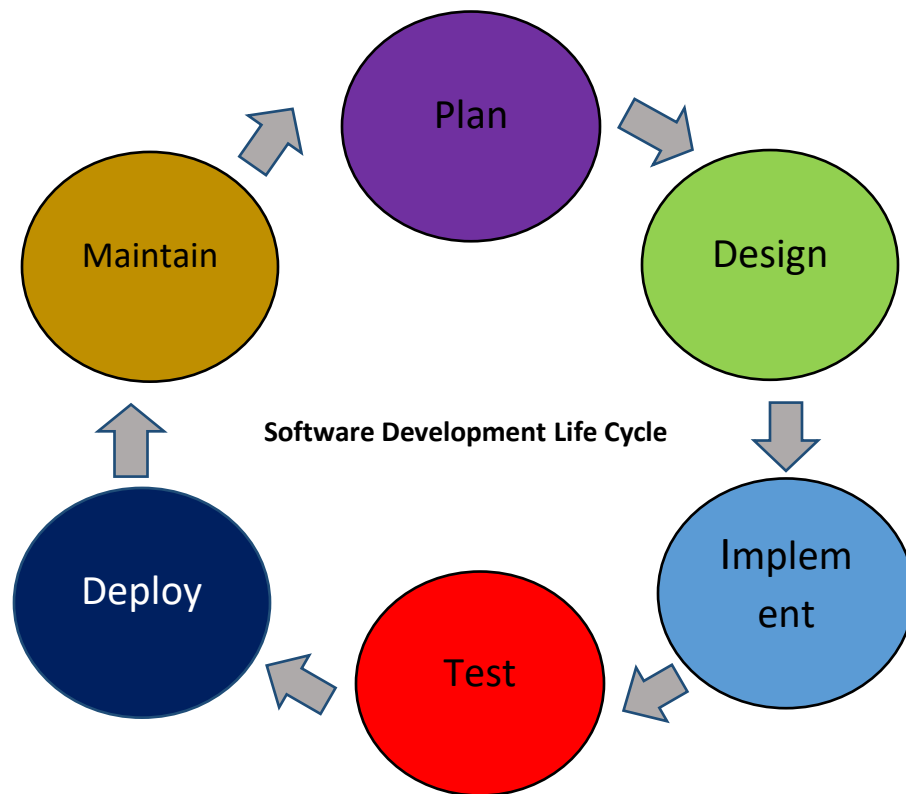


Figure 1: The six phases of SDLC

2.1. Phase-1: Plan

In this initial phase, the development team works with stakeholders to identify and gather detailed requirements for the software. This involves understanding the project's goals, user needs, functionality and constraints.

2.2. Phase-2: Design

Once the requirements are clear, the design phase will begin. During this stage, the development team creates a comprehensive design for the software, including architecture, data structures, interface and modules.

2.3. Phase-3: Implementation (Coding)

In this phase, developers write the code according to the specifications provided in the design phase. It involves converting the design into actual software using programming languages and development tools.

2.4. Phase-4: Testing

After the implementation phase is completed, the software undergoes various testing procedures. This includes unit testing, integration testing and system testing.

2.5. Phase-5: Deployment

Once the testing is successfully completed, the software is deployed to the production environment or made available to end users.









2.6. Phase- 6: Maintenance

After deployment, the software requires ongoing maintenance to address bugs, new features and adapt to changes in the operating environment.

3. SDLC Models:

There are different Software Development Life Cycle models specify and design, which are followed during the development phase. These models are also called as “Software development process models”. Each process model follows a series to ensure the success in the step of software development.

Some Important phases of SDLC Models are:

-  Waterfall Model
-  RAD Model
-  Spiral Model
-  V- Model
-  Incremental Model
-  Agile Model
-  Iterative Model
-  Big bang Model

3.1. Waterfall Model:

Waterfall model is a sequential development process that flows like a waterfall through all phases of a project (Analysis, Design, development and Testing).

For Example: With each phase completely wrapping up before the next phase begins.

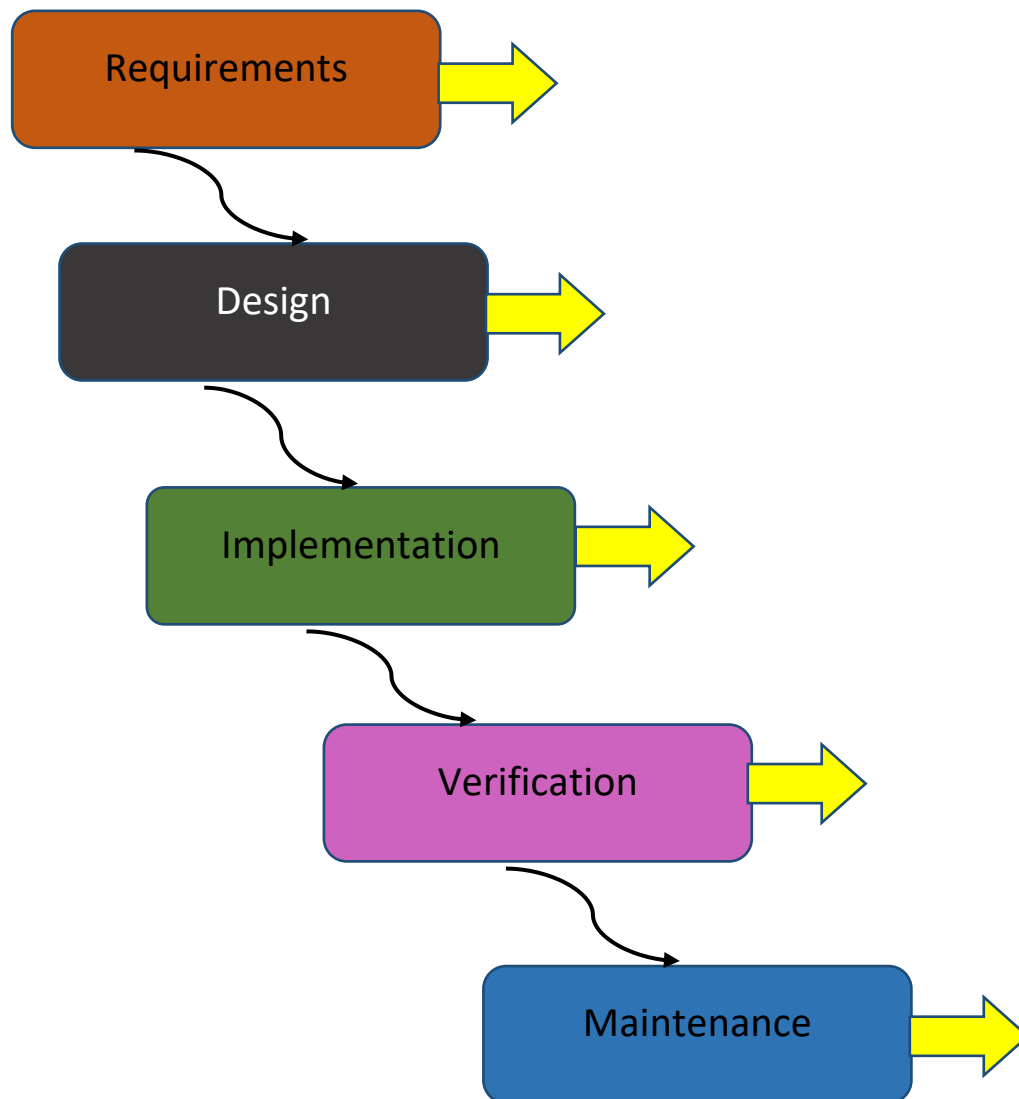


Figure 2: Waterfall Model

- **Advantage:**

1. The Waterfall model depends upon the sequential approach in which each stage should complete itself to start the next stage.
2. The development moves from concept, complete design, deployment, testing, installation, troubleshooting, and ends up in maintenance and operation. Each phase of development has its own value and should be worked properly.

- **Dis –advantage:**

- 1.The waterfall model breaks down project activities into linear sequential phases.
Each set of circumstances depends on the deliverables that came from the previous step as it corresponds to a specialization of task-oriented approaches.
- 2.This method is typically seen in the areas of construction development and engineering design because each step must get fulfilled before the next one can follow.
- 3.Waterfall model does not support making changes.It delays testing until after the completion of the project.

3.2. RAD Model:

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

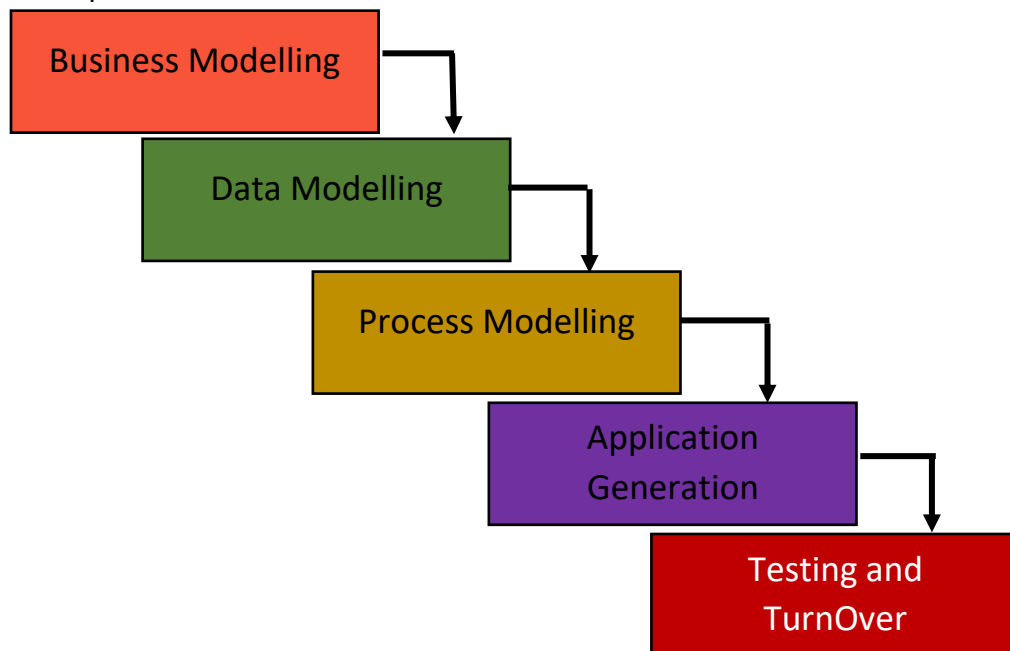


Figure 3: RAD Model

- **Advantages:**

1. RAD Model is flexible for change and reduces the development time.
2. Each phase in RAD brings highest priority functionality to the customer and increases the reusability of features.

- **Dis-Advantages:**

1. It required highly quality designs and all application is not compatible to RAD.
2. For smaller project, we cannot use the RAD Model and required user involvement.
On highly technical risk it is not suitable.

3.3. V-Model

The V-model is type of SDLC model where process is executes in sequential manner in V-shape. It is also known as verification and validation model. It is based on the association of testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase.

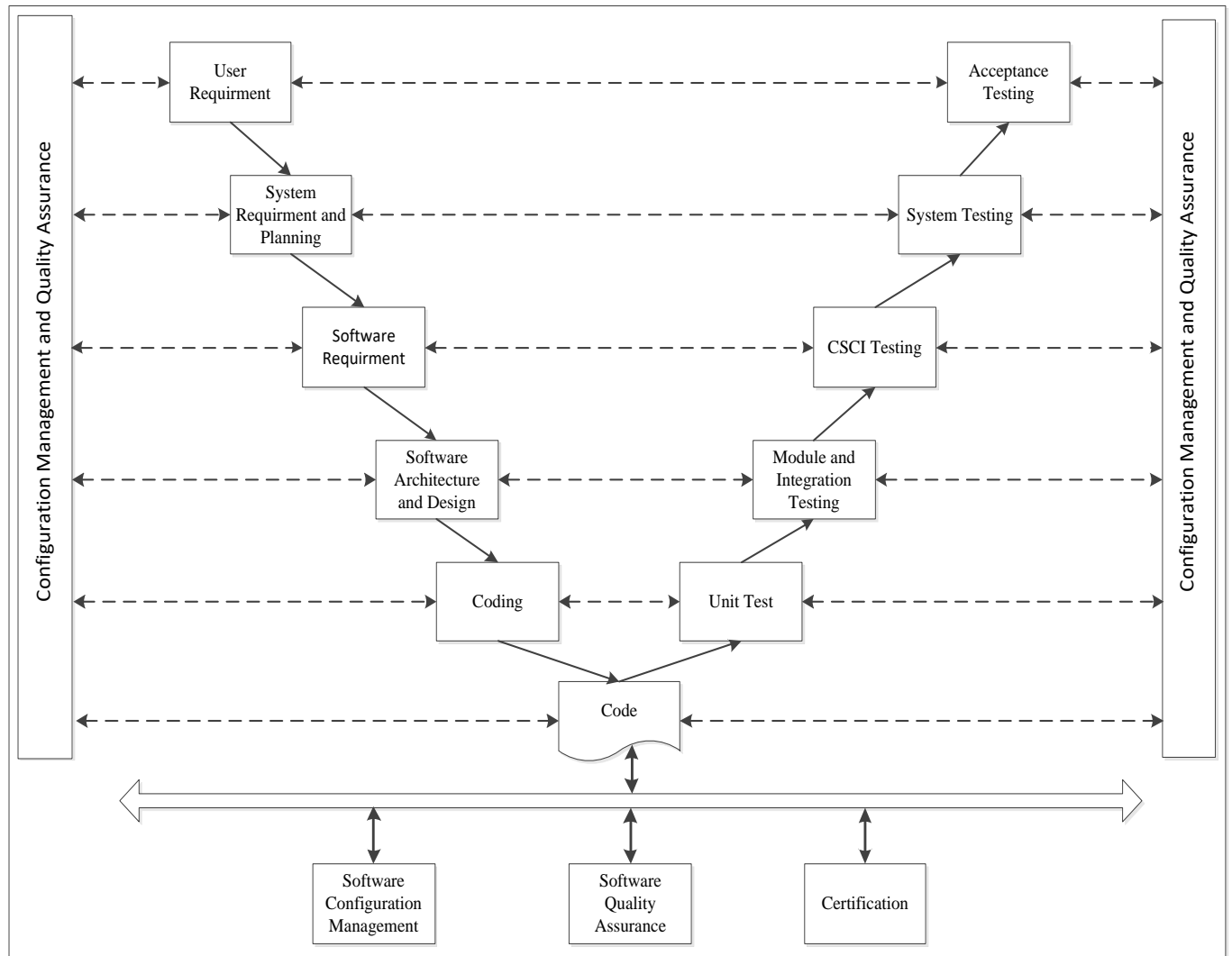


Figure 4: V-model flow diagram.

- **Advantages:**

1. **Clear Documentation and Planning:** The V-Model encourages thorough documentation of requirements, designs, and test cases at each stage, which leads to better planning and understanding of the project.
2. **Early Detection of Defects:** Testing activities are initiated early in the development process, which means defects are identified and addressed sooner, reducing the cost of fixing them later in the project.

3. **Easy to Manage and Control:** Because the phases are clearly defined and correspond directly to testing activities, it is relatively easy to manage and control the project.

- **Disadvantages:**

1. **Inflexible to Changes:** Like the Waterfall model, the V-Model is less adaptable to changing requirements. Once a phase is completed, it's difficult and costly to make changes.

2. **Not Suitable for Complex or Large-scale Projects:** The V-Model may not be suitable for projects with high complexity or those where requirements are subject to frequent changes.

3. **Potential for Late Testing:** If there are delays in the development phase, it can lead to a backlog of testing activities, potentially causing project delays.

3.4. Spiral Model:

Spiral Model is a software development life cycle model that combines the elements of both the iterative and waterfall models. It was proposed by Barry Boehm in 1986 and is designed to address some of the limitations of traditional linear development approaches like the waterfall model. The Spiral Model is particularly well-suited for large and complex projects with changing requirements and significant risk factors.

- **Advantages:**

1. **Risk Management:** The model explicitly addresses and manages risks at every phase of the project. This helps in identifying potential issues early, reducing the likelihood of project failure.

2. **Flexibility and Adaptability:** The Spiral Model allows for flexibility in accommodating changes to requirements. It is well-suited for projects where requirements are not well-understood or may evolve over time.

3. **Client Involvement:** There are frequent opportunities for client involvement and feedback, which can lead to a product that better meets the client's needs and expectations.

- **Disadvantages:**

1. **Complexity:** The Spiral Model can be more complex to manage compared to linear models like Waterfall. It requires a well-structured risk assessment process, which can be resource-intensive.

- **Resource Intensive:** The model requires significant resources for risk assessment, prototyping, and frequent iterations. This can lead to higher development costs.

- **Time-Consuming:** The iterative nature of the model can lead to longer development times, particularly if there are multiple iterations.

3.5. Big Bang Model

The Big Bang model in SDLC is a term used to describe an informal and unstructured approach to software development, where there is no specific planning, documentation, or well-defined phases.

- **Advantages:**

1. **Rapid Start:** Development can begin immediately without the need for extensive planning or documentation. This can lead to quick initiation of the project.

2. **Flexibility:** The Big Bang Model is highly flexible and can accommodate changing requirements and priorities as the project progresses.

3. **Low Initial Cost:** Since there is no formal planning phase, the initial cost of starting a project using the Big Bang Model may be lower compared to other models.

- **Disadvantages:**

1. **High Risk of Failure:** Due to the lack of formal planning and structure, there is a high risk of failure. Requirements may be misunderstood, leading to a product that does not meet user needs.
2. **Challenging to Scale:** The Big Bang Model may not be suitable for large, complex projects with multiple teams, as the lack of structure can lead to coordination and communication challenges.
3. **Customer Satisfaction Risks:** Since the client's involvement is typically limited until the end of the project, there is a higher risk of delivering a product that does not meet their expectations.