Project Report

SDLC Waterfall Model

Written By

Name: Pasang Chand

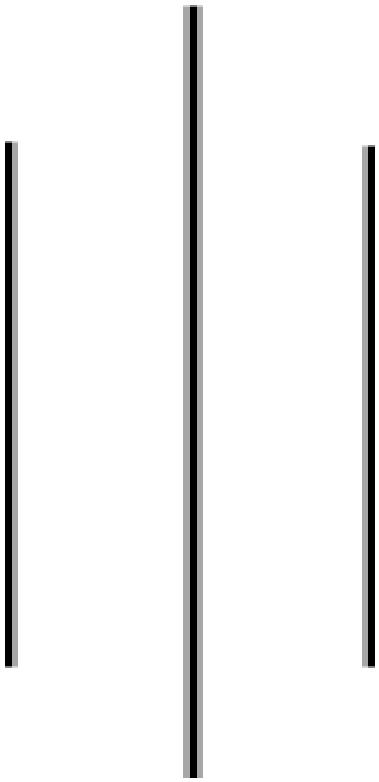
Roll Number:126

Class:12(XII)

Date of Submission: 10th july,2025

PADMODAYA SECONDARY SCHOOL

RamShahPath,Kathmandu



Lab Report Number: 01
Lab Report On: SDLC waterfall Model

Submitted By

Name: Pasang Chand
Roll Number: 126
Class: 12(XII)
Subject: Computer
Faculty: Science

Submitted To

.....

Department of Computer Science

Date of Submission:10th July,2025

CONTENTS

1. Introduction of SDLC

The waterfall model was discovered by Winston W.Royce in 1970. It Became widely adopted due to its clear structured and disciplined approach to software development. SDLc is an organized way to develop a software/system. It is a systematic process of developing any software. It helps in establishing a system,software,project or plane. SDLC starts when development personnel feel that new software or an improvement in the existing software is required.

One of the best and most traditional models of the SDLC. The report highlights it's characteristics,working process,advantages and limitations to provide a better understanding of its role in software engineering.

Software project management refers to the planning, organizing, and controlling of resources,activities, and tasks involved in developing and delivering software projects. It involves applying project management principles and techniques to ensure the successful completion of software within the defined scope, budget, and timeline.

2. Phases of the SDLC Waterfall Model

Types of Phases of the SDLC waterfall Model:

- Requirements Gathering and Analysis
- System Design
- Implementation/Coding
- Testing
- Deployment/Installation
- Maintenance

Explanation of this phases

- Requirements Gathering and Analysis

The requirement is the first stage in the SDLC process.It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry.Planning for the quality assurance requirements and recognition of the risks involved is also done at this stage.Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

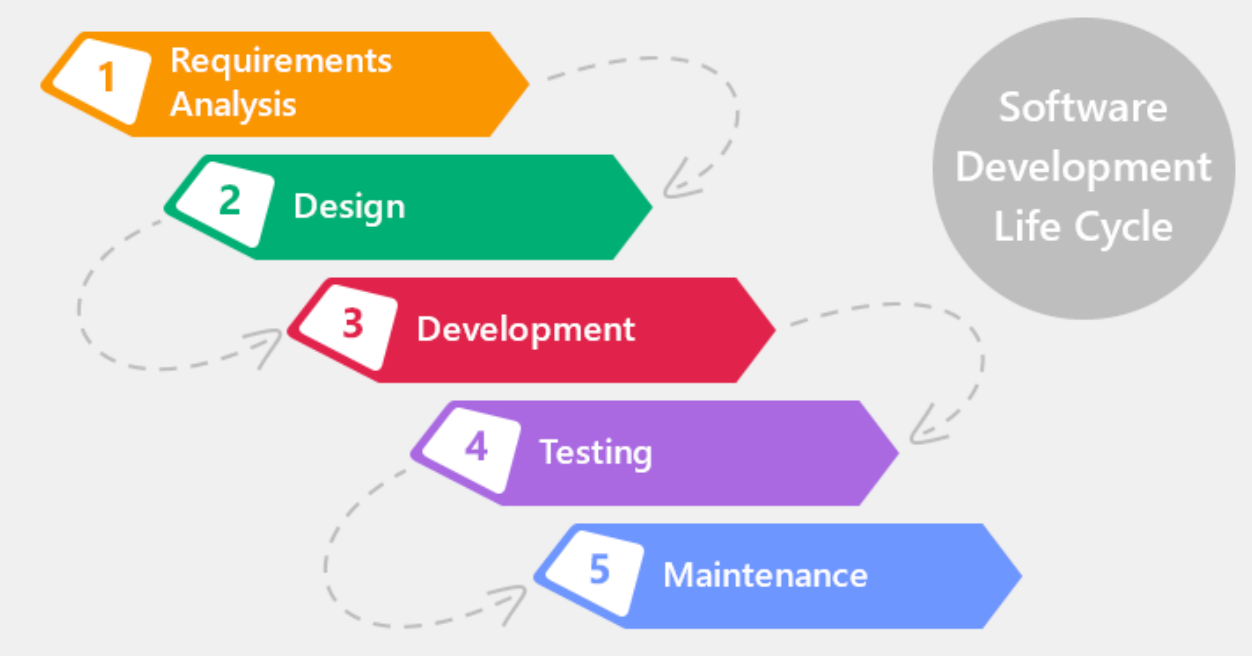


Fig:Phases of the SDLC Waterfall Model

- System Design

  In this second phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

  This design phase serves as input for the next phase of the model.There are two kinds of design documents developed in this phase:

  - ➢ High-Level Design:-Brief description and name of each module.An outline about the functionality of every moduleInterface relationship and

dependencies between modules. Database tables identified along with their key elements. Complete architecture diagrams along with technology details

- ➢ Low-Level Design:-This is the functional logic of the modules. Its database tables, which include type and size. It does not give complete detail of the interface. Addresses all types of dependency issues. Listing of error messagesComplete input and outputs for every module.

- Implementation/Coding

  In this phase, developers start build the entire system by writing code using the chosen programming language.In the coding phase, tasks are divided into units or modules and assigned to the various developers.It is the longest phase of the Software Development Life Cycle process.They need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

- Testing

  It is an iterative process that helps identify defects, errors, or bugs in the software before it is released to end-users.The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement. It helps to identify and fix any defects or bugs. Different types of testing, such as unit testing, integration testing, system testing, and user acceptance testing, are performed to ensure the software functions as intended.

- Deployment/Installation

  Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

- Maintenance

Once the system is deployed, and customers start using the developed system, following 3 activities occur

- ➢ Bug fixing – Bugs are reported because of some scenarios which are not tested at all .
- ➢ Upgrade – Upgrading the application to the newer versions of the Software
- ➢ Enhancement – Adding some new features into the existing software.

The focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

3. Key Characteristics/Principles

Key Characteristics / Principles of the Waterfall Model:

- Sequential Flow

  The process flows strictly from one phase to the next (e.g., Requirements → Design → Implementation → Testing → Deployment). Each phase must be completed before the next begins.

- No Overlapping Phases (in theory)

  Ideally, phases do not overlap or iterate.You move forward in a linear fashion without returning to earlier stages.

- Early Freezing of Requirements

  Requirements are gathered and finalized at the beginning of the project.Changes to requirements after this point are discouraged or costly.

- Emphasis on Documentation

  Each phase produces detailed documents (e.g., requirement specs, design docs).These documents serve as formal records and communication tools.

- Difficulty in Accommodating Changes

  Since the model assumes stable requirements, adapting to change later in the process is challenging and expensive.
  There is minimal flexibility for iterative revisions.

4. Advantages and Disadvantages of the waterfall Model

Advantages of the waterfall Model:

1. Simplicity and ease of understanding.
   The linear structure makes it easy to follow, especially for beginners or teams with less experience.
2. Good for small, well-understood projects.
   Works best when requirements are clear, fixed, and not expected to change.
3. Clear deliverables and milestones.
   Each phase has defined outputs, making project tracking straightforward.
4. Easy to manage and control (especially for managers).
   The structured nature allows better control over timelines, budgets, and resources.
5. Strong documentation.
   Comprehensive documentation at each phase supports future maintenance and helps new team members understand the project easily.

Disadvantages of waterfall Model:

1. Lack of flexibility (difficulty in accommodating changes).
   Once a phase is completed, revisiting it is difficult and costly, making the model rigid.
2. High risk and uncertainty (especially for complex projects)
   Problems may not surface until later stages, increasing the chances of failure in large or undefined projects.
1. Late discovery of errors/defects.
   Since testing happens only after the development phase, errors found late are harder and more expensive to fix.
2. Customer feedback is limited to early stages.
   Clients are involved mainly during the requirements phase and may not see the product until it is fully developed.
3. Long development cycles.
   The entire product must be built before delivery, which can delay deployment and business value.
4. Not suitable for projects with evolving requirements.
   The model struggles to adapt to changing needs or emerging user feedback.

5. Projects/Real Life Example of SDLC Waterfall Model

Projects/Real Life Example of SDLC Waterfall Model:

1.Government or Defense Projects with Strict Regulations Example:

- Military software systems
- Tax filing systems
- National ID databases

Why Suitable:

- ➢ These projects require strict documentation,
- ➢ compliance with laws, and formal approval processesChanges are minimal once development begins,
- ➢ making the rigid structure of Waterfall ideal.

2. Embedded Systems Development Example:

- Software for washing machines, printers, or ATMs
- Firmware for microcontrollers

Why Suitable:

- ➢ Requirements are fixed and must match specific hardware constraints.
- ➢ Extensive testing and documentation are essential.
- ➢ Changes are costly after production, so a linear approach works well.

3. Simple Utility Applications Example:

- Basic calculators
- Note-taking apps
- File conversion tools

Why Suitable:

- ➢ These projects are small in scale, easy to understand, and have clear goals.
- ➢ No need for frequent user feedback or iterative development.

4. Database or Backend Systems with Stable Requirements Example:

- Inventory management systems
- Payroll processing systems

Why Suitable:

- ➢ Functional requirements are well-defined from the beginning.
- ➢ Minimal UI or user-driven changes expected during development.

5. Educational Projects or Proof-of-Concepts Example:

- Projects done for academic purposes
- Demos used to illustrate specific technologies

Why Suitable:

- ➢ The focus is on following a structured process, learning, and documentation.
- ➢ Limited scope and predefined objectives match Waterfall's strengths.

6. Conclusion

The Waterfall Model is one of the earliest and most structured approaches to software development within the SDLC framework. Its sequential flow, strong emphasis on documentation, and clear phase-by-phase deliverables make it a suitable choice for projects with well-defined, stable requirements. While it is simple to understand and manage, especially for small-scale or regulatory-bound projects, it also comes with limitations such as inflexibility, late error detection, and limited adaptability to change.

Despite the rise of more iterative and agile methodologies, the Waterfall Model still holds value in specific domains—such as government systems, embedded software, and educational projects—where control, predictability, and formal process adherence are critical. Understanding its characteristics, strengths, and limitations enables project managers and development teams to select the right model for their specific needs, ensuring better project planning, execution, and outcome.

7.References

Winston W. Royce (1970) – ***Managing the Development of Large Software Systems***

Pressman, R. S. – Software Engineering: A Practitioner's Approach

Sommerville, Ian – Software Engineering

IEEE Standard 1074-1997

Boehm, Barry W. **Asmita publication**

Er. Prachandra Ram Shrestha Asmitapublication

Roshan Bhusal Asmita Publication

Ridip Khanna Asmita Publication