**SEMINAR REPORT**

ON  
**“SOFTWARE DEVELOPMENT LIFE CYCLE”**

Submitted to

**Sant Gadge Baba Amravati University, Amravati.**

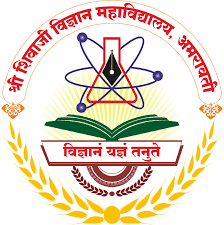
In partial fulfillment of the requirements of

**M.Sc. (Computer Software) Final Year Examination**

Submitted by

**Sakshi V. Shekar**

**(Department of Computer Science)**



**Shri Shivaji Education Society Amravati’s**

**SHRI SHIVAJI SCIENCE COLLEGE, Amravati.**

**2023-2024**

**Internal Examiner External Examiner**

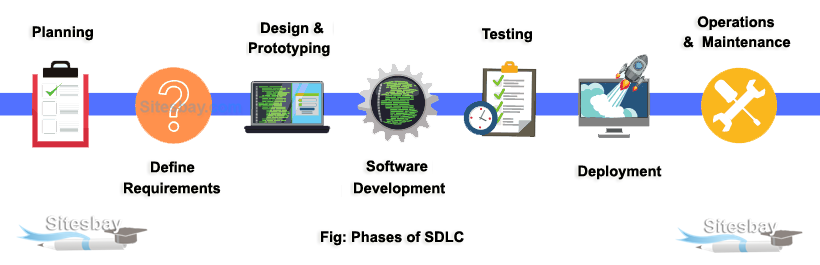
| **INDEX** | | |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page No.** |
| 1 | Introduction | 1 |
| 2 | Models:   * Agile * Waterfall * V-Shaped * Iterative * Spiral | 4 |
| 3 | Challenges and Future Trends | 9 |
| 4 | Conclusion | 10 |
| 5 | References | 10 |

**INTRODUCTION**

Software Development Life Cycle (SDLC) is a structured process followed by software developers to design, develop, test, and deploy high-quality software efficiently. It encompasses a series of stages that ensure systematic progression from the conception of an idea to the delivery of a fully functional software product. This seminar report provides an in-depth analysis of the SDLC, its various phases, methodologies, best practices, and challenges.

Phase of SDLC

SDLC (Software Development Life Cycle) is a term used in software development to narrate a procedure for planning, designing, creating, testing, and deploying software or application.



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. SDLC mainly contains following phases.

* Requirement gathering
* Analysis
* Designing
* Coding and Implementation
* Testing
* Deployment
* Maintenance

**Planning and requirement analysis:** Requirement Gathering is the most important stage in SDLC. Business analyst and project organizer do client meeting to gather all the requirements of the clients like what to build, who will be the end-user, what is the purpose of the product, etc. Before creating a product, a core understanding or knowledge of the product is very necessary.

**Defining requirements:** Once the requirement analysis is done, the next stage is to surely document the software specifications and get them approved by the project stakeholders. This can be accomplished through the "SRS"- Software Requirement Specification document, which embraces all the product elements to be created and developed during the project life cycle.

**Designing:** In this stage, the requirements gathered in the SRS document is used as information to obtain the software architecture. Moreover, this phase also consists of storyboarding or wire framing software that is needed for functionality. Through this, the developers then create either rough working models, or illustrates how the software will work, how it will look, how usage flows will move from screen to screen, and more.

**Implementation or Coding**: In this stage of SDLC, the exact development begins, and the programming is built. The execution of design begins concerning script code. Developers have to follow the coding guidelines defined by their management, and programming tools like compilers, interpreters, debuggers, etc. are used to generate and implement the code.

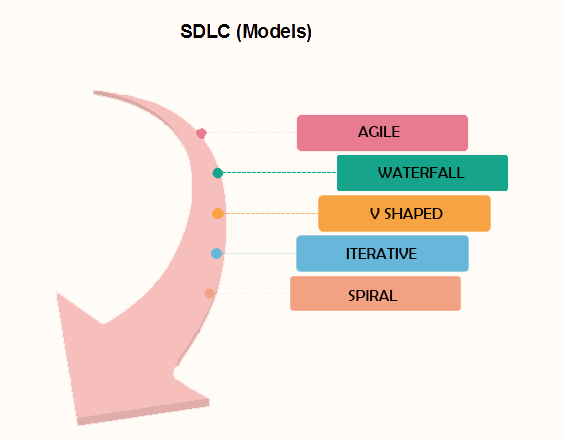
**Testing:** After the code is generated, it is tested against the specifications to ensure that the products are solving the needs directed and inferred during the requirements stage. During this phase, several testing like unit testing, integration testing, system testing, and approval testing are done.

**Deployment:** Once the software is approved, and no bugs or errors are asserted, then it is deployed. Later, based on the assessment, the software is delivered as it is or with suggested augmentation in the object segment. After the software is deployed, then its sustenance begins.

**Maintenance:** Once the client starts using the developed software, then the real issues start coming up. In this stage, the team is required to fix the issues, rollout new features and refine the functionalities as required. The method where the care is taken for the finished product is thus known as maintenance.

**Software Development Life Cycle Model**

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.



There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "Software Development Process Models." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

* Agile Model
* Waterfall Model
* V-Model
* Iterative Model
* Spiral Model

**Agile Model**

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like.

* Planning
* Requirements Analysis
* Design
* Coding
* Unit Testing and
* Acceptance Testing.

**When to use Agile Model**

Agile works really well when the product vision or features are not well defined. Agile allows product owners to adjust requirements and priorities along the way to take advantage of opportunities and ultimately deliver a better product to all of the project stakeholders.

**Advantages of Agile Model**

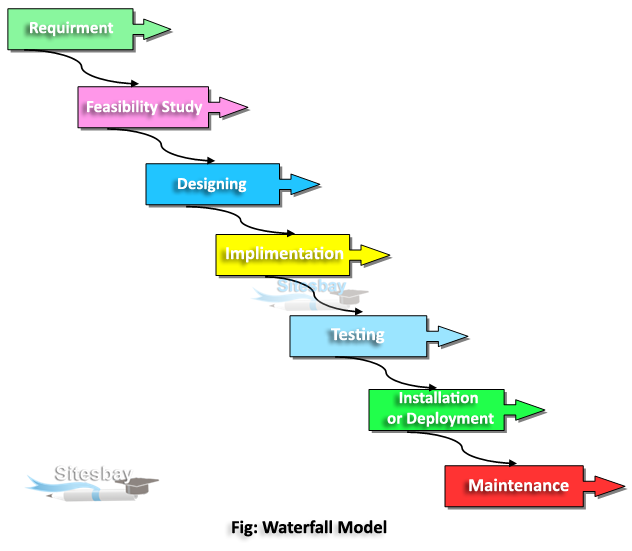
* Frequent Delivery
* Face-to-Face Communication with clients.
* Efficient design and fulfils the business requirement.
* Anytime changes are acceptable.

**Disadvantages of Agile Model**

* + Not suitable for handling complex dependencies
  + More risk of sustainability, maintainability and extensibility.

**Waterfall Model**

The waterfall model is a linear, sequential approach to the software development life cycle (SDLC) that is popular in software engineering and product development. The waterfall model emphasizes a logical progression of steps. Similar to the direction water flows over the edge of a cliff, distinct endpoints or goals are set for each phase of development and cannot be revisited after completion.



When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

* When the requirements are constant and not changed regularly.
* A project is short
* The situation is calm

**Advantages of Waterfall Model**

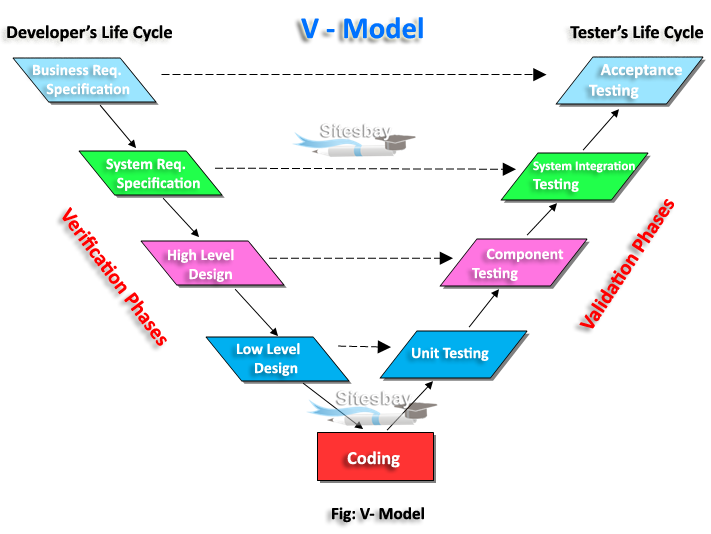
* Before the next phase of development, each phase must be completed
* Suited for smaller projects where requirements are well defined
* They should perform quality assurance test (Verification and Validation) before completing each stage

**Disadvantages of Waterfall Model**

* Error can be fixed only during the phase
* not desirable for complex project where requirement changes frequently

**V-Shaped Model**

V- Model is also known as Verification and Validation Model. In this model Verification & Validation goes hand in hand i.e. development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.



**When to use V-Model?**

* When the requirement is well defined and not ambiguous.
* The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
* The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

**Advantages of V-Model**

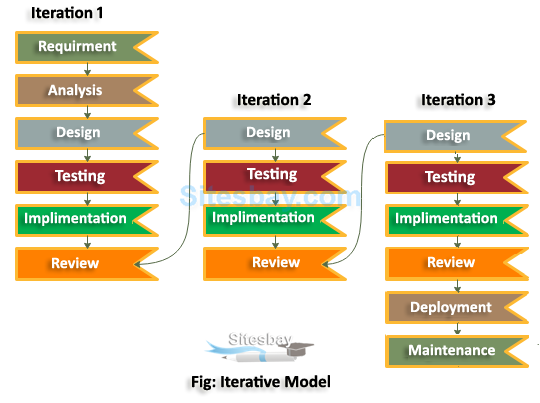
* It is a simple and easily understandable model.
* Testing Methods like planning, test designing happens well before coding.
* Works well for small plans where requirements are easily understood.

**Disadvantages of V-Model**

* Very rigid and least flexible.
* Not a good for a complex project.
* V-shaped model is not good for Ongoing projects

**Iterative Model**

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.



The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process. Consider an iterative life cycle model which consists of repeating the following phases in sequence:

* Requirements
* Analysis
* Design
* Testing
* Implementation
* Review
* Maintenance

**Advantages of Iterative Model**

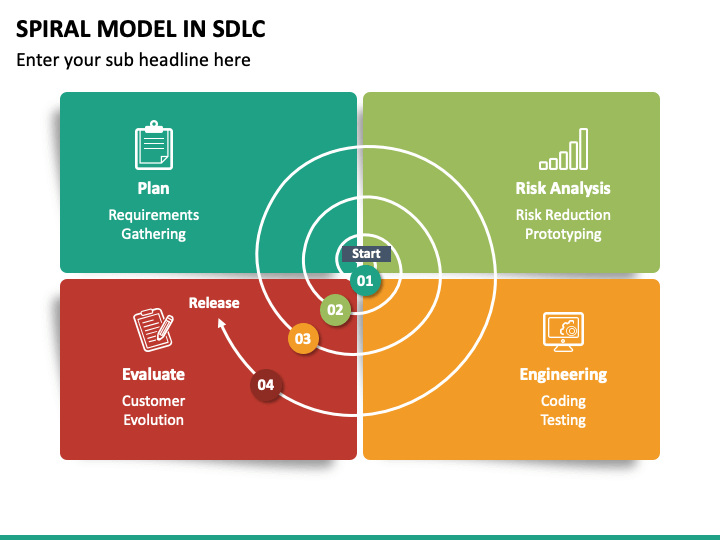
* Generates working software quickly and early during the software life cycle.
* More flexible - less costly to change scope and requirements.

**Disadvantages of Iterative Model**

* Each phase of an iteration is rigid and do not overlap each other.
* Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

**Spiral Model**

Spiral Model is one of the mostly usages process model for development of large scale application or enhanced application. Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model.



Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process. Each phase of spiral model in software engineering begins with a design goal and ends with the client reviewing the progress.

**Advantages of Spiral Model**

* Risk Handling
* Good for large projects
* Flexibility in Requirements
* Customer Satisfaction

### Disadvantages of Spiral Model

* Complex
* Expensive
* Too much dependability on Risk Analysis

**CHALLENGES**

1. Changing Requirements: One of the most significant challenges in SDLC is managing changing requirements throughout the development process.

2. Resource Constraints: SDLC projects often face resource constraints in terms of skilled personnel, budget limitations, and infrastructure requirements.

3. Technical Complexity: Software projects are becoming increasingly complex due to advancements in technology, integration of multiple systems, and evolving user expectations.

4. Communication and Collaboration: Effective communication and collaboration among team members, stakeholders, and cross-functional teams are critical for project success.

**FUTURE SCOPE**

1.Integration of AI and Machine Learning: The future of SDLC is likely to see increased integration of artificial intelligence (AI) and machine learning (ML) technologies.

2. DevOps and Continuous Delivery: DevOps practices and m continuous delivery are expected to become more prevalent in SDLC. Automation, collaboration, and feedback loops will be key focus areas for organizations looking to streamline their development, testing, and deployment processes.

3. Agile at Scale: Agile methodologies such as Scrum and Kanban will continue to evolve to support larger and more complex projects. Agile at scale frameworks like SAFe (Scaled Agile Framework) and LeSS (Large Scale Scrum) will gain popularity as organizations seek to achieve agility across their entire enterprise.

4. Low-Code/No-Code Development: With the growing demand for rapid application development, low-code/no-code platforms are expected to gain traction in SDLC.

**CONCLUSION**

In conclusion, the Software Development Life Cycle (SDLC) remains a fundamental framework for guiding the development of high-quality software products. Despite facing challenges such as changing requirements, resource constraints, and technical complexity, SDLC continues to evolve to meet the demands of a rapidly changing technological landscape.

Throughout its history, SDLC has adapted to new methodologies, tools, and practices to address emerging challenges and opportunities. From traditional waterfall models to agile methodologies, DevOps practices, and beyond, SDLC has continuously evolved to improve efficiency, quality, and agility in software development.

**REFERENCES**

* Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
* Sommerville, I. (2015). Software Engineering. Pearson Education Limited.
* McConnell, S. (2013). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.
* Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes, 11(4), 14-24.
* Schwaber, K., & Sutherland, J. (2017). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org.