IJARETY

International Journal of Advanced Research in Education and TechnologY (IJARETY)

Volume 12, Issue 6, November-December 2025

Impact Factor: 8.152

www.ijarety.in editor.ijarety@gmail.com

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in | | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

A Review of SDLC Approaches and Their Conceptual Perspectives

Fasseela Mol KJ, Gokula Krishna, Smitha C Thomas

PG Student, Department of Computer Science and Engineering, APJ Abdul Kalam Technological University, Kerala, India

Assistant Professor, Department of Computer Science and Engineering, APJ Abdul Kalam Technological University, Kerala, India

Professor, Department of Computer Science and Engineering, APJ Abdul Kalam Technological University, Kerala, India

ABSTRACT: The successful execution of any software project necessitates a well-defined and structured development process. Software Development Life Cycle (SDLC) methodologies offer diverse models designed to align with the requirements of various proposed projects, providing predictable and efficient system development scenarios. This paper provides a comprehensive review of the major SDLC methodologies, illuminating the core Traditional (Heavy-weight) and Agile (Light-weight) approaches, along with their respective models. A detailed analytical study of the strengths and weaknesses of popular models—including Waterfall, Iterative, Spiral, V-Model, Big Bang, eXtreme Programming (XP), Scrum, Feature-Driven Development (FDD), and Kanban—is presented. The ultimate goal of SDLC is to deliver a high-quality program that meets customer expectations within a defined duration and reasonable cost.

I. INTRODUCTION TO SDLC METHODOLOGIES

Software Development Life Cycle (SDLC) methodologies are promising to be adapted in the development of all sectors and projects. They are fundamentally designed to provide a systematic and logical framework for project development, with applications across virtually all sectors. The SDLC methodology is meticulously crafted within the contours of an organizational structure, delineating a constellation of indispensable stages arranged within a systematic and cogent framework.

The typical outline of the SDLC process consists of several sequential phases:

- **Requirements Analysis and Plans**
- **Product Design**
- **Implementation**
- **Testing**
- **Maintenance**

SDLC methodologies are broadly classified into two principal types: the **Traditional (Heavy-weight)** approach and the **Agile (Light-weight)** approach to software development. The choice of the appropriate model is contingent upon the nature and features of the specific project.

II. ESSENTIAL STEPS APPLIED IN SDLC MODELS

To ensure the success of the software development process, each SDLC model must include a series of unique, essential steps, such as requirements analysis, designing, implementation, testing, and maintenance of high-quality software.

- **Step 1: Requirements Analysis** This critical initial stage involves a specialized team collecting and analyzing customer requirements, documenting them clearly in the **Software Requirements Specification (SRS)** document. Developers then analyze the information, draw up a project development plan, and conduct a feasibility study according to economic and technical factors.

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in | | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

- **Step 2: Designing** The design phase follows requirements analysis, depicting the structural form of the requirements within a clear programming language. This design, including how the project will be implemented, is formally documented in the **Software Design Description (SDD)**, which is then reviewed by experts for risk, quality, cost, and schedule.
- **Step 3: Implementation and Unit Testing** This stage includes the **coding** part that was precisely documented in the SDD. Testing is conducted for all small and working units in the project (**unit testing**), and modifications can be made during the inspection stage if necessary.
- **Step 4: Testing** The dedicated testing phase is vital for identifying and addressing errors to improve product quality. A full **system test** must be performed, as unit testers may not properly check the program interface between units. This stage is important to build trust in the developers before the product is delivered.
- **Step 5: Maintenance** The final stage involves delivering the product, operating it, and publishing it in the market. The product is often first deployed for testing in a real business environment called **User Acceptance Testing (UAT)**. During maintenance, coding errors are corrected, gradual improvements are made, and old parts of the product are deleted.

III. COMPARATIVE ANALYSIS OF SDLC MODELS

SDLC models, also known as **Software Development Process Models (SDPM)**, are designed with specific, sequential steps to ensure successful development.

3.1 Traditional SDLC Approach (Heavy-Weight)

The traditional model is one of the oldest approaches, characterized by its **sequential approach**, meaning all its steps are carried out step by step. It is mainly based on **planning, analysis, and detailed discussion** of the project, documenting all steps from gathering requirements to deploying the product.

3.1.1 Key Traditional Models

- **Waterfall Model** The waterfall model is the first model within the SDLC approach, proposed by Royce in 1970. It is known as the **sequential linear life cycle model**. The next stage only begins once the current stage is fully completed sequentially.

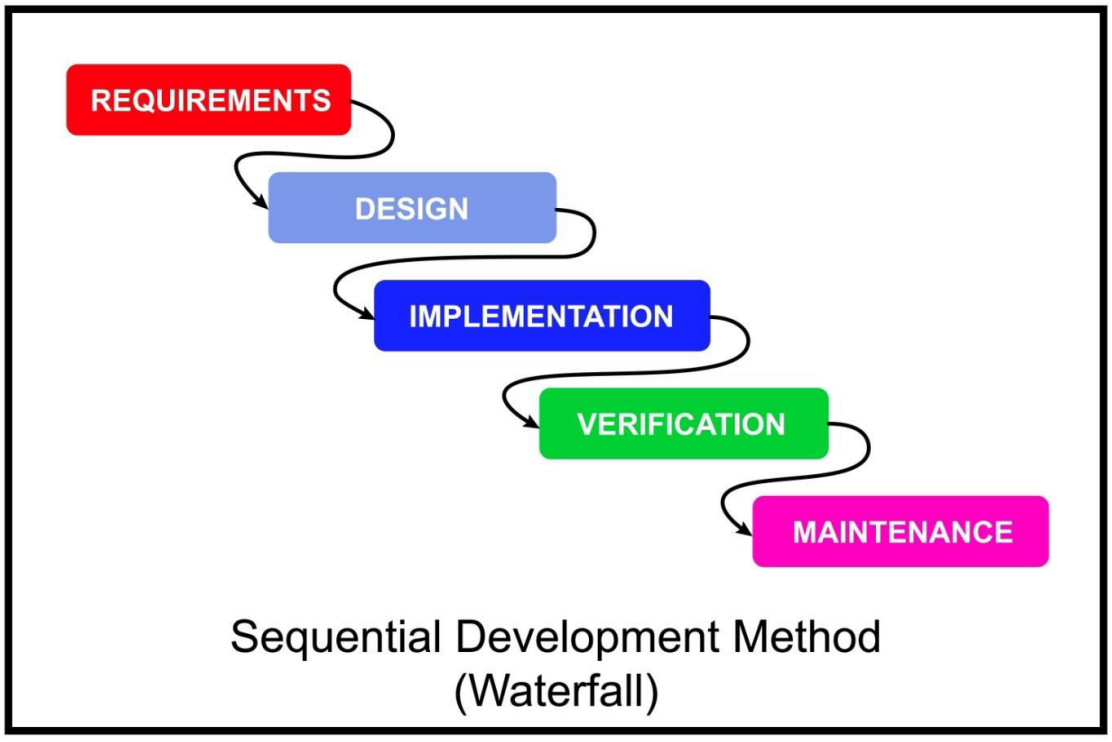

FIG 1. Waterfall model

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

Strengths	Weaknesses
• Simple and easy to understand and use	• The amount of risk and ambiguity is high
• Easy to manage due to the strength of the model	• Unsuitable for complex, object-oriented, or long-term projects
• Each stage has a specific output and review process	• Changes in project requirements cause problems
• Suitable for small projects because the requirements are understandable	• Difficulty identifying technical and commercial obstacles early

• **Iterative Model** :This model implements software requirements through **small, repeated units** (iterations/increments) to reach a completed system. Each architecture unit is comprised of design, development, testing, and implementation. All requirements must be checked and tested at each cycle of development.

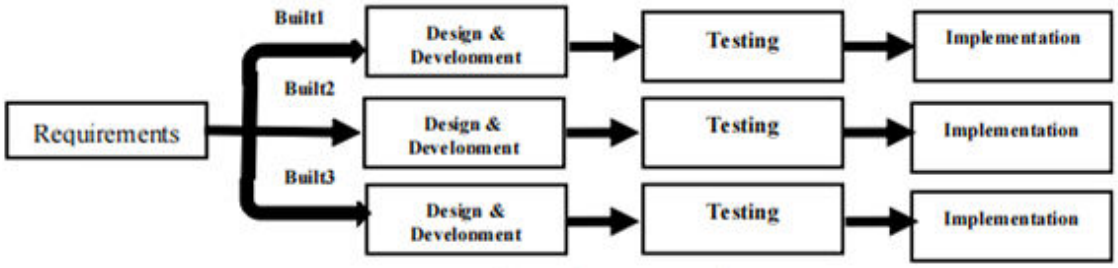

FIG 2 Iterative model

Strengths	Weaknesses
• Scalable quickly and early, with quick and periodic results	• Requires more resources
• Easy to measure progress and limit cost of requirement changes	• System architecture design problems often arise because not all requirements are collected initially
• Ease of conducting tests and handling errors through iterations	• Not suitable for small-scale projects

• **Spiral Model** Developed by Barry Boehm, this model integrates the **iterative** development model with the sequential (waterfall) model, focusing strongly on **risk**

• **analysis**. The product progressively improves through an iterative process around the coil.

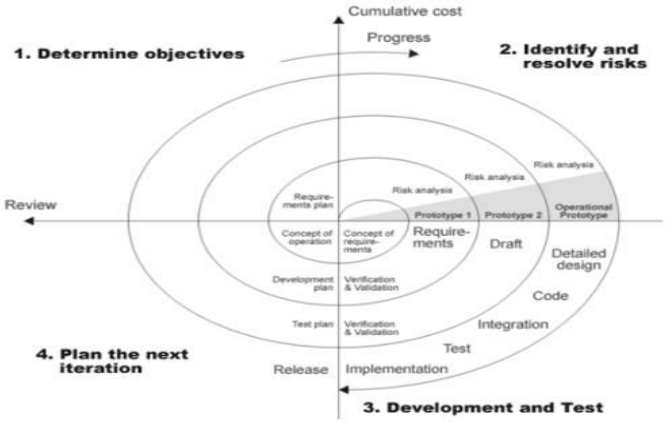

FIG 3. Spiral model

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in | | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

Strengths	Weaknesses
• Able to deal with changes in project requirements	• Model management is too complicated
• It handles a large number of prototypes	• An early end to the project is uncertain, and steps are complicated
• Quick to display the system to users and good risk management	• Often the spiral continues indefinitely, requiring documentation and intermediate stages

- **V-Model** Similar to the Waterfall model, stages are executed through a sequence of stages. It is strictly disciplined, and no stage begins until the completion of the one before it. It is characterized by associating the **testing process** with each development stage, linking verification on one side with validation on the other.

Strengths	Weaknesses
• This model is very disciplined, sequential, and easy to understand/manage	• Ambiguous and highly risky
• Perfect for small projects	• Not suitable for complex, object-oriented, long-term, or change-prone projects
	• It is not possible to return and change its functions when the test stage is reached

- **Big Bang Model** Assigned for **small projects**, this model does not follow specific steps and relies heavily on inputs (money and efforts) and outputs (the product). It is fast, requires less analysis, and involves less or no planning.

Strengths	Weaknesses
• Simple and easy model, not requiring planning	• Ambiguous and the risk is high
• Model management is easy, and it is very flexible	• Not suitable for object-oriented, complex, ongoing, or long-term projects
• It is a suitable educational platform for new students	• Less understanding of the requirements leads to high costs

3.2 Agile SDLC Approach (Light-Weight)

The agile approach is a lightweight, 'moving quickly' methodology that relies on developing software in **small parts** called 'iterations' or 'increments'. The focus is on **customer collaboration and interaction** with the product development process, rather than traditional detailed planning.

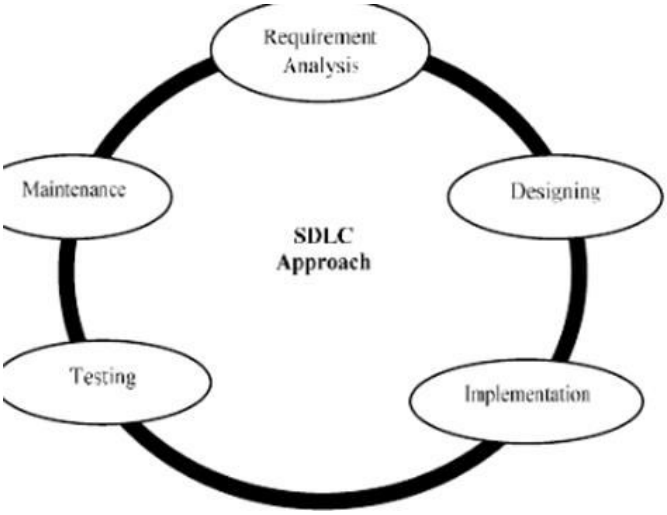

FIG 4 Steps of the sdlc approach

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in | | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

3.2.1 Key Agile Models

- **eXtreme Programming (XP) Model** Conceptualized in the 1990s, XP is rooted in the collaborative efforts of the development team, focusing on creating the necessary program through systematic, tested iterations and customer feedback. Its four main stages are planning, coding, design, and testing. It is suitable for small enterprises.
  - **Weaknesses:** Lack of documentation and poor design, making it unsuitable for medium and large projects. The structure is complex, making it difficult to adopt a specific design activity.
- **Scrum Model** Developed by Ken Schwaber and Jeff Sutherland in 1993, Scrum is the most prominent agile model, defined as a framework for solving complex problems and delivering high-quality, innovative products. Activities are organized within a standard framework consisting of several **sprints**. Scrum is continuously adaptable to all changes in requirements.
  - **Feature-Driven Development (FDD) Model** Developed by Jeff DeLuca and Peter Coad, FDD focuses primarily on the **design and construction phases**, as well as product quality during development. It works incrementally and iteratively, but is best suited for **small and medium projects**. The project progress steps are monitored during the product delivery process.
  - **Weaknesses:** Not suitable for large and long-term projects.
- **Kanban Model** Originally developed by Toyota for manufacturing, this approach is widely approved for systems development and is based on the **Just-In-Time (JIT)** concept. It provides information on what and how much is needed within a schedule. It maintains transparency, helps monitor workflow, and aims to reduce the project completion time.
  - **Principles:** Each worker should know their current task, the developer must make developmental changes, and employees must be prepared to be leaders within the program development.

IV. COMPARATIVE ANALYSIS OF TRADITIONAL VS. AGILE APPROACHES

A comparative analysis highlights the primary differences, weaknesses, and strengths between the Traditional (Sequential) and Agile (Adaptive) approaches.

Table 1. Major Differences Between Traditional and Agile Approaches

Key Criteria	Traditional Approach	Agile Approach
• Requirements	• Fixed, changing is difficult	• Frequently changing, easily changeable
• Development Type	• Stable	• Easily changeable
• Project Size	• Suitable for Large	• Suitable for Small and medium
• **Potential Risks**	• Limited effect	Great effect
• **Changes Cost**	• High	Low
• **Documentation**	• More documentation	Minimal documentation
• **Customer Role**	• Limited	Large interaction with clients
• **Team Size**	• Large team	Small team

Key Strengths and Weaknesses:

- **Traditional Approach:** It is a **predictive approach** that relies on a detailed development plan and more documentation. It is suitable for large projects and emphasizes a specific requirement model before implementation. However, weekly meetings are not suitable for developers, the time frame for each iteration is short, and there is a lack of communication between developers and customers.
- **Agile Approach:** It is an **adaptive approach** that adapts to dynamic requirement changes, with the product being frequently tested, resulting in minimum risks. It features open communication and team collaboration. Conversely, it does not depend on a detailed development plan, is difficult for beginner developers, and frequent changes can cause design problems. Delays in completing iterations can also increase development costs.

| ISSN: 2394-2975 | www.ijarety.in | | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 6, November-December 2025 ||

DOI:10.15680/IJARETY.2025.1206011

V. LIMITATIONS AND FUTURE DIRECTIONS

5.1 Limitations

- **Traditional Approach Limitations**
  - The regression method applied does not allow returning to previous steps, meaning the step must be completed.
  - It is **inflexible** as it does not allow changes to be made according to customer requirements.
  - Any change in the approach structure leads to the project starting again, incurring additional cost.
  - Increased time and expenses may result if the customer keeps adding requirements to the list.
  - Team members not in the current stage often remain idle.
- **Agile Approach Limitations**
  - Stakeholders are often used in project management if there is no specific timetable for the end of the project.
  - Project costs are clearly affected if tasks are not specified accurately.
  - Requires a **highly experienced team** to complete the project within the specified period.
  - Leaving a team member during the development period negatively affects the project.

5.2 Future Directions

Future research on SDLC is poised to integrate advanced methodologies to enhance model selection and performance prediction.

- **Multi-Criteria Decision Making (MCDM):** Techniques such as AHP, ANP, TOPSIS, VIKOR, and others represent a promising research area. These techniques can be used to calculate the weights and priority of multiple criteria (e.g., cost, time, complexity) and determine the most appropriate SDLC model for a given case study.
- **Machine Learning (ML):** ML algorithms represent an ideal direction for the **classification and prediction** of the SDLC model's performance.

VI. CONCLUSION

Software Development Life Cycle (SDLC) methodologies represent the best solution to the software development process in all areas of life. This study conducted a comprehensive review of the most important SDLC approaches: the **Traditional approach** (Waterfall, Spiral, Iterative, V-Model, and Big Bang) and the **Agile approach** (XP, Scrum, FDD, and Kanban). Future works should emphasize new research directions using methodologies such as **decision-making techniques and machine learning** in software development.

REFERENCES

1. Kute, Seema Suresh, and Surabhi Deependra Thorat. "A review on various software development life cycle (SDLC) models." *International Journal of Research in Computer and Communication Technology* 3.7 (2014): 778-779.
2. Alshamrani, Adel, and Abdullah Bahattab. "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model." *International Journal of Computer Science Issues (IJCSI)* 12.1 (2015): 106.
3. Gupta, Aparna, Achint Rawal, and Yamini Barge. "Comparative study of different SDLC models." *Int. J. Res. Appl. Sci. Eng. Technol* 9.11 (2021): 73-80.
4. Akinsola, Jide ET, et al. "Comparative analysis of software development life cycle models (SDLC)." *Computer science on-line conference*. Cham: Springer International Publishing, 2020.
5. Iqbal, Syed Zaffar, and Muhammad Idrees. "Z-SDLC model: a new model for software development life cycle (SDLC)." *International Journal of Engineering and Advanced Research Technology (IJEART)* 3.2 (2017): 8.
6. Ragunath, P. K., et al. "Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC)." *International Journal of Computer Science and Network Security* 10.1 (2010): 112-119.
7. Olorunshola, Oluwaseyi Ezekiel, and Francisca Nonyelum Ogwueleka. "Review of system development life cycle (SDLC) models for effective application delivery." *Information and Communication Technology for Competitive Strategies (ICTCS 2020) ICT: Applications and Social Interfaces*. Singapore: Springer Singapore,
2021. 281-289.
8. Tuteja, Maneela, and Gaurav Dubey. "A research study on importance of testing and quality assurance in software development life cycle (SDLC) models." *International Journal of Soft Computing and Engineering (IJSCE)* 2.3 (2012): 251-257.
9. Arora, Ritika, and Neha Arora. "Analysis of SDLC models." *International Journal of Current Engineering and Technology* 6.1 (2016): 268-272.
10. Okesola, Olatunji J., et al. "Software requirement in iterative SDLC model." *Computer Science On-line Conference*. Cham: Springer International Publishing, 2020.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975 Impact Factor: 8.152

www.ijarety.in editor.ijarety@gmail.com