International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

Relevance Of Software Development Life Cycle Models For Effective Programming

Dr. Mohit Kumar Sharma

Associate Professor & Head, Post Graduate Department of Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India

***Abstract*-- Software Development Life Cycle process models are involved the activities in creation of quality software. Software Development Life Cycle is an important procedure used in software engineering to describe a plan for planning, creating, coding, testing and implementation of user requirement specification. SDLC is part of an overall product lifecycle, which includes maintenance and changes to user requirements. SDLC is step by step process for creating quality software for users. The lifecycle model is the first step in project planning. It determines the organization of development activities, delivery timing, frequency, and resource allocation. Modern project planners need more options in lifecycle models, methods, and tools to deliver higher-quality software faster and cheaper. This paper includes different phases of SDLC and analysis of different process models for effective programming.**

***Keywords*-- SDLC models, Waterfall, Spiral, Quality**

I. INTRODUCTION

Software Development Life Cycle is an important procedure used in software engineering to describe a plan for planning, creating, coding, testing and implementation of user requirement specification. Modern project planners need more options in lifecycle models, methods, and tools to deliver higher-quality software faster and cheaper. SDLC is part of an overall product lifecycle, which includes maintenance and changes to user requirements. SDLC is step by step process for creating quality software for users [1].

The lifecycle model is the first step in project planning. It determines the organization of development activities, delivery timing, frequency, and resource allocation. Modern project planners need more options in lifecycle models, methods, and tools to deliver higher-quality software faster and cheaper [2].

Software Development Life cycle (SDLC) phases listed are: requirements specification, feasibility study, design, analysis, coding, testing, implementation and maintenance. System specification acts as an interaction between client and producer. Activities include specifying the system, compiling requirements definition, establishing an exact project schedule, validating the specification, justifying economic feasibility.

Feasibility study is a phase that analyses risks, costs, and benefits related to economics, technology, and user organizations. All feasibilities are equally important and guide whether a project should proceed. Analysis establishes the software's requirements, services, constraints, and include completing requirements analysis, delimiting the problem domain, sketching target system components, creating a rough project schedule.

System Design Focuses on how the system will be built and Produces design specifications for the overall system and individual components. Different activities include designing the system architecture, defining component interfaces and creating detailed design documents. The practical development of software design starts in terms of writing program code in the suitable programming language and developing error free executable programs efficiently. Software may need to be integrated with the libraries, databases, and other program. The software developer team should be expert in programming skills required to develop a good software product.

Software testing is done after creating software to remove errors or mistakes, bugs to make it error free good quality software product. While coding by the developers and testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, object-oriented testing, and testing the product at static and dynamic levels. Testing time consumes more time as compare to other phases of SDLC. Implementation includes installing the software on user computer. At times, software needs post-installation configurations at user end. It includes all hardware and software requirements to run the developed and tested software. Software is tested for portability and adaptability is solved during implementation. It also includes training of software to the user for efficient working.

Software maintenance is to remove errors or bugs, to change platforms requirements and add new features to existing software. There are corrective, perfective and adaptive maintenance for correcting, adding and change platform to existing software to make more reliable.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

II. Objectives Of The Study

1. To study the relevance of software development life cycle models for effective programming.
2. To encourage software developers to use SDLC models for developing quality software.
3. To explore the comparison of different software development life cycle models.

III. Review Approach

Software development life cycle model is a process of high-level tasks that are utilized to create software in systematic way. The lifecycle model selection is the first step in the project planning process. There are a number of different models for software development lifecycles. The choice of a lifecycle model determines the organization of software development activities over time, effecting when software will be delivered, how often, and with what resources. These are different types of software lifecycle models [3] listed as:

1. Build and Fix Lifecycle Model
2. Waterfall Model
3. V-shaped Lifecycle Model
4. Iterative Lifecycle Model
5. Incremental Lifecycle Model
6. Evolutionary Lifecycle Model
7. Prototyping Lifecycle Model
8. Boehm's Spiral Lifecycle Model
9. Win-Win Spiral Model
10. Object-oriented Lifecycle Model

IV. Comparitive Study Of Software Development Life Cycle Models

Build-and-Fix Lifecycle Model is also called the ad-hoc model, it is the original and widely used software lifecycle model. Process involves build first version of the software [3]. It is unfortunate that many software products are developed using what might be termed as the build-and-fix approach. The software is developed without first addressing specifications or design. Instead, the developers simply build a product that is reworked any times as necessary until it satisfies its client [4].

Waterfall Lifecycle Models was introduced by Winston Royce in 1970 and is currently the most commonly used model for software development. The waterfall model is an engineering model designed to apply to the development of software [5]. There are two variations of Waterfall Lifecycle model as Original Waterfall Lifecycle Model and Iterative Waterfall Lifecycle Model [6].

In V-shaped Lifecycle Model, the system and functional testing of requirements can capture any requirements changes at the beginning. In this model, it is possible to obtain user satisfaction by showing them the result of requirements functional testing. The drawback is that here also the functional unit can be seen only at the end of the development.

An Iterative lifecycle 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 [7].

This process is then repeated, producing a new version of the software for each cycle of the model. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance [8].

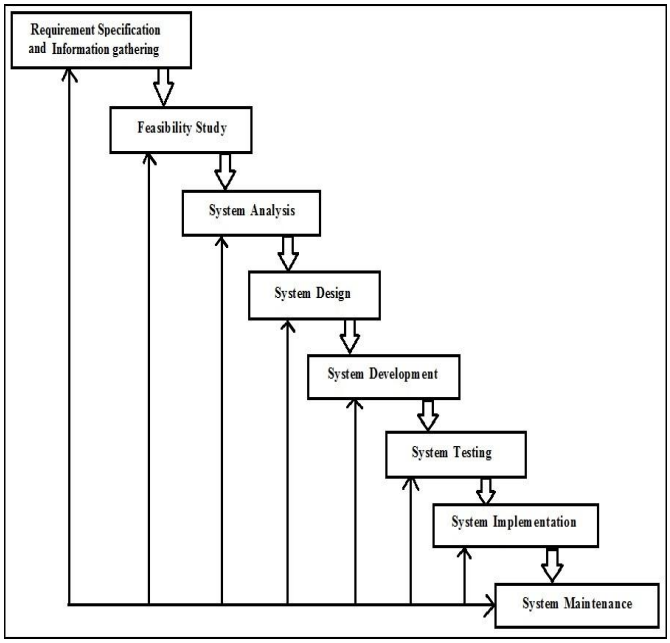

Fig. 1.1 – Software Development Waterfall Model

In Incremental model, an overall architecture of the total system is developed first; the detailed increments and releases are planned. Each increment has its own complete lifecycle. The increments may be built serially or in parallel depending on the nature of the dependencies among releases and on availability of resources. Each increment adds additional or improved functionality to the system.

Evolutionary Lifecycle Model accommodates incremental development using experience from earlier increments to help define requirements for subsequent increments.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

Increments are developed in a sequential manner, rather than in parallel, and within each incremental development cycle, there is a normal progression through analysis, design, code, test and implementation, followed operations and maintenance.

In Prototyping Lifecycle Model based on the idea of developing an initial implementation for user feedback, and then refining this prototype through many versions until a satisfactory system emerges. Prototyping is also referred to as evolutionary development. Prototyping aims to enhance the accuracy of the designer's perception of the user's requirements. The specification, development and validation activities are carried out concurrently with rapid feedback across the activities.

In Boehm's Spiral Lifecycle Model combines elements of the waterfall lifecycle model, along with an emphasis on the use of risk management techniques. Boehm proposed a spiral model, where each round of the spiral. Identifies the sub-problem which has the highest risk associated with it and finds a solution for that problem. Its chief distinguishing feature is that it is primarily risk-avoidance driven rather than document-driven, code driven or release driven. The primary advantage of the spiral model is that its range of options allows it to accommodate the best features of existing software process models, while its risk-driven approach helps it to avoid most of their difficulties [3].

Win-Win Spiral Process Model is a model of a process based on Theory W, which is a management theory and approach based on making winners of all of the system's key stakeholders as a necessary and sufficient condition for project success.

The main difference between object-oriented development and the other development lifecycles is in the modelling of the system components [9]. Other development lifecycles operate on a separation of events and data, while the object-oriented development encapsulates events with the specific data for which the event is valid. Object-oriented analysis develops an object-oriented model of the application domain. Object-oriented design develops an object-oriented model of the software system. Object-oriented programming realizes the software design with an object-oriented programming language that supports direct implementation of objects, classes, and inheritance.

Even though incremental development is planned when using the evolutionary model, the increments are not likely to be developed in parallel. This is because the purpose of each increment is to deliver a portion of the system with known requirements for subsequent increments.

Unlike the waterfall and incremental models, the system requirements & feasibility and software requirements processes are not done once for the entire project, but these are revised at the start of each evolutionary development cycle to incorporate the latest requirements.

Evolutionary Lifecycle Model is preferable lifecycle model when requirements are not fully known, but a subset of the requirements is known, stable and understood. Benefits include the early delivery of some functions and the early testing of some assumptions before the entire system is built around them. The major weakness of this model is related to the inability to plan in detail at the outset of the project. Because the requirements are not fully known, problems with the scope creep, inaccurate estimating and less than optimal architecture are possible.

The predominately sequential nature of this lifecycle makes it not particularly rapid or cost efficient for complex systems. Project managers using the evolutionary model must plan to revise the overall architecture as the system evolves. Flexibility must also be built-in at the individual software module level. Time that appears to be gained on the front end of a project because of the early release of the first increment may be spent on later increments on modification and integration efforts [10].

V. Results And Discussion

The Software Development Life cycle models ensure software quality as essential to keep up the standard level for user's acceptance worldwide through different reputed bodies such as ISO, IEEE, ANSI, etc [3]. Software quality depicts the attributes of software ensuring reliability and productivity. It is a procedure for assessment, evaluation and improvement for programming execution. Programming quality incorporates user and execution prerequisites and documentation. Software quality aims to provide guaranteed products including all characteristics required by the user. It guarantees updates in the product item normally according to the changing necessity of equipment arrangements and needs of the client.

Programming quality can be kept up through scientific estimations by programming measurements use. These measurements give quality outcomes to support and further programming improvement. The procedure for programming quality beginnings from necessity and data assembling by the user. This information gathering should be accurate and correct for the fruitful fulfillment of programming. An analysis is required to remove anomalies and ensuring user prerequisites.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

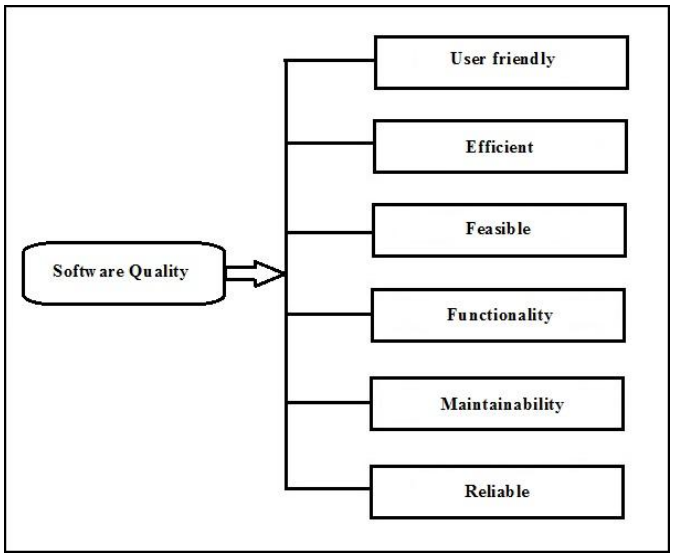

Fig. 1.2 – Software Quality Attributes

VI. CONCLUSION

The Software Development Life cycle models each suit different project needs. The Waterfall model works best for projects with stable and well-understood requirements but lacks flexibility. The Incremental model improves on this by delivering parts of the system early and supporting parallel development, though it requires careful coordination. The Evolutionary model is ideal when requirements are not fully known and evolve over time, allowing early feedback but risking scope creep and estimation issues. The Spiral model focuses on risk management and is most suitable for large, complex, high-risk projects. In short, no single model fits all projects — the choice depends on requirement clarity, risk, project size, and the need for flexibility and early delivery.

REFERENCES

[1] Roger S. Pressman and B.R. Maxim, Software Engineering, Tata-McGraw Hill Publishing House, 9th edition, 2023

[2] Ali Behforooz and Frederick J.H,, Software Engineering Fundamentals, Oxford University Press. 1996

[3] Nasib S. Gill, Software Engineering-Software Reliability and Testing, K.B. Publishing, 2020

[4] Shubhmeet Kaur, "A Review of Software Development Life Cycle Models", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 11, November 2015 ISSN: 2277 128X

[5] Shubham Dwivedi, "Software Development Life Cycle Models - A Comparative analysis", International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 2, February 2016

[6] Pargaonkar, S. "A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering", International Journal of Scientific and Research Publications, 13(08)

[7] Wadkar, V., & Ghorpade, S., "A Comparative Analysis of Software Life Cycle Models: Phases, Activities, and Order of Execution", Iconic Research and Engineering Journals, 8(3), 13-17

[8] Taya, S., & Gupta, S.," Comparative Analysis of Software Development Life Cycle Models", International Journal of Computer Science and Technology, Vol. 2, Issue 4, ISSN: 0976-8491

[9] Rodriguez-Martinez, L. et al., "Review of Relevant System Development Life Cycles (SDLCs) in Service-Oriented Software Engineering", Journal of Applied Research and Technology, 10(2), 94-113. ISSN 1665-6423

[10] Kumar, M., & Rashid, E., "An Efficient Software Development Life Cycle Model for Developing Software Project", International Journal of Education and Management Engineering, Vol.8, No.6, pp.59-68.

ABOUT AUTHOR

Dr. Mohit Kumar Sharma (Ph.D., M.Phil., M.Sc.) is working as Associate Professor & Head, Post Graduate Department of Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India. He has 18 years teaching and research experience. He has 21 research publications with book chapters in international refereed journals/publishers, 20 paper presentations at international/national level conferences/ seminars, 2 books authored and 3 edited books. He is also member, undergraduate and postgraduate board of studies, Computer Science and Applications in Panjab University, Chandigarh. His research interests are in Software Engineering and Object-Oriented Programming.