Volume 3, 2019 ISSN: 2617-4537
DOI: 10.31058/j.data.2019.34002

A Review on Models of Software Development Life Cycle

Bindia Tarika[1*]

[1] Computer Science & Engineering, PTU, Punjab, India

Email Address
bindiatarika11@gmail.com (BindiaTarika)
*Correspondence: bindiatarika11@gmail.com

Received: 10 November 2019**;** **Accepted:** 1 December 2019**;** **Published:** 15 December 2019

Abstract:
One of the basic notions of the software development process is SDLC models which stand for Software Development Life Cycle models. SDLC – is a continuous process, which starts from the moment, when it's made a decision to launch the project, and it ends at the moment of its full remove from the exploitation. There is no one single SDLC model. They are divided into main groups, each with its features and weaknesses. This paper describes the various models of SDLC and also gives result of each.

Keywords:
Software, SDLC, Models, Phases

1. Introduction

Evolving from the first and oldest "waterfall" SDLC model, their variety significantly expanded. The SDLC models diversity is predetermined by the wide number of product types – starting with a web application development to complex medical software. And if you take one of the SDLC models mentioned below as the basis – in any case, it should be adjusted to the features of the product, project, and company. Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

i. SDLC is the acronym of Software Development Life Cycle.

ii. It is also called as Software Development Process.

iii. SDLC is a framework defining tasks performed at each step in the software development process.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

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.

The most used, popular and important SDLC models are given below:

i. Waterfall model

ii. Iterative model

iii. Spiral model

iv. V-shaped model

v. Agile model

2. Basic Stages Of Software Development Life Cycle

Stage 1. Planning and requirement analysis

Each software development life cycle model starts with the analysis, in which the stakeholders of the process discuss the requirements for the final product. The goal of this stage is the detailed definition of the system requirements. Besides, it is needed to make sure that all the process participants have clearly understood the tasks and how every requirement is going to be implemented. Often, the discussion involves the QA specialists who can interfere the process with additions even during the development stage if it is necessary.

Stage 2. Designing project architecture

At the second phase of the software development life cycle, the developers are actually designing the architecture. All the different technical questions that may appear on this stage are discussed by all the stakeholders, including the customer. Also, here are defined the technologies used in the project, team load, limitations, time frames, and budget. The most appropriate project decisions are made according to the defined requirements.

Stage 3. Development and programming

After the requirements approved, the process goes to the next stage – actual development. Programmers start here with the source code writing while keeping in mind previously defined requirements. The system administrators adjust the software environment, front-end programmers develop the user interface of the program and the logics for its interaction with the server.

The programming by itself assumes four stages

Algorithm development

Source code writing

Compilation

Testing and debugging

Stage 4. Testing

The testing phase includes the debugging process. All the code flaws missed during the development are detected here, documented, and passed back to the developers to fix. The testing process repeats until all the critical issues are removed and software workflow is stable.

Stage 5. Deployment

When the program is finalized and has no critical issues – it is time to launch it for the end users. After the new program version release, the tech support team joins. This department provides user feedback; consult and support users during the time of exploitation. Moreover, the update of selected components is included in this phase, to make sure, that the software is up-to-date and is invulnerable to a security breach.

3. SDLC Models

3.1. Waterfall SDLC Model

Waterfall – is a cascade SDLC model, in which development process looks like the flow, moving step by step through the phases of analysis, projecting, realization, testing, implementation, and support. This SDLC model includes gradual execution of every stage completely. This process is strictly documented and predefined with features expected to every phase of this software development life cycle model.

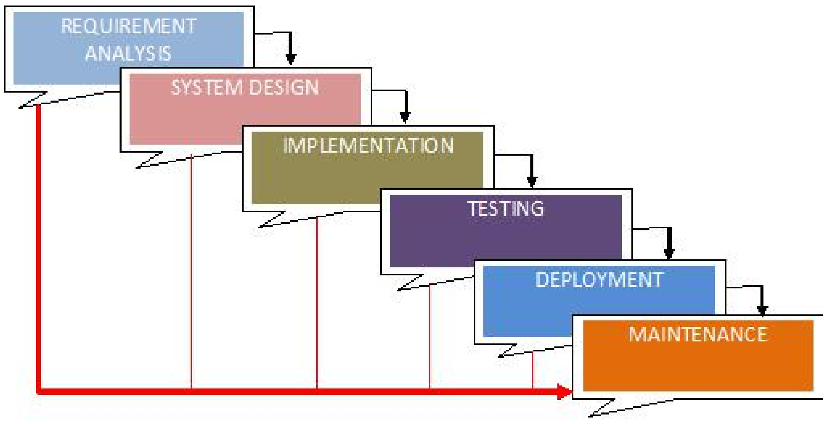

Figure 1. *Waterfall Model.*

Table 1. *Advantages/Disadvantages of Waterfall Model.*

ADVANTAGES	DISADVANTAGES
Simple to use and understand	The software is ready only after the last stage is over
Management simplicity thanks to its rigidity: every phase has a defined result and process review	High risks and uncertainty
Development stages go one by one	Not the best choice for complex and object-oriented Projects
Perfect for the small or mid-sized projects where requirements are clear and not equivocal	Inappropriate for the long-term projects
Easy to determine the key points in the development cycle	The progress of the stage is hard to measure while it is still in the development
Easy to classify and prioritize tasks	Integration is done at the very end, which does not give the option of indentifying the problem in advance

Use cases for the Waterfall SDLC model:

The requirements are precisely documented

Product definition is stable

The technologies stack is predefined which makes it not dynamic

No ambiguous requirements

The project is short

3.2. Iterative SDLC Model

The Iterative SDLC model does not need the full list of requirements before the project starts. The development process may start with the requirements to the functional part, which can be expanded later. The process is repetitive, allowing to make new versions of the product for every cycle. Every iteration (which last from two to six weeks) includes the development of a separate component of the system, and after that, this component is added to the functional developed earlier. Speaking with math terminology, the iterative model is a realization of the sequential approximation method; that means a gradual closeness to the planned final product shape.

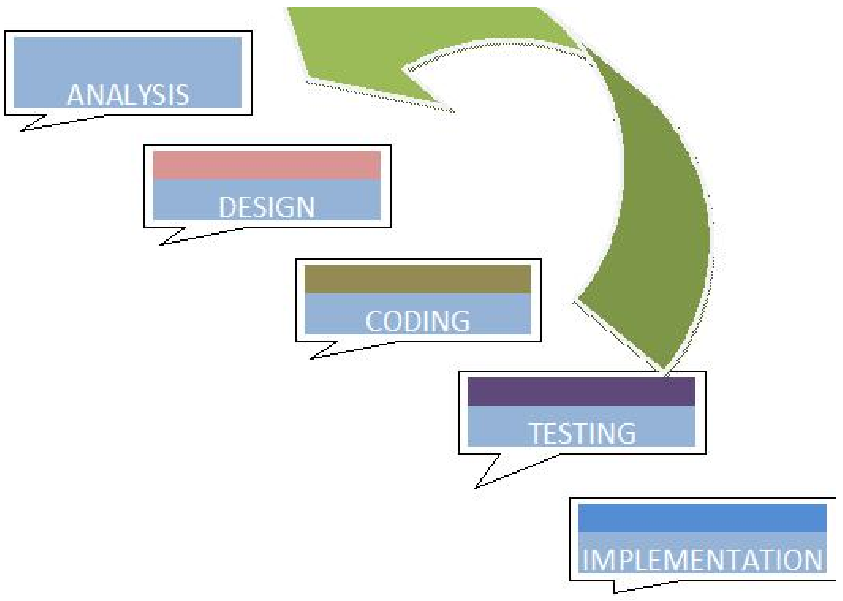

Figure 2. *Iterative Model.*

Table 2. *Advantages/Disadvantages of Iterative Model.*

ADVANTAGES	DISADVANTAGES
Some functions can be quickly developed at the beginning of the development lifecycle	Iterative model requires more resources than the waterfall model
The paralleled development can be applied	Constant management is required
The progress is easy measurable	Issues with architecture or design may occur because not all the requirements are foreseen during the short planning stage
The shorter iteration is - the easier testing and debugging stages are	Bad choice for the small projects

It is easier to control the risks as high-risk tasks are completed first	The process is difficult to manage
Problems and risks defined within one iteration can be prevented in the next sprints	The risks may not be completely determined even at the final stage of the project

Use cases for the Iteration model:

The requirements to the final product are strictly predefined

Applied to the large-scale projects

The main task is predefined, but the details may advance with the time

3.3. Spiral SDLC Model

Spiral model – is SDLC model, which combines architecture and prototyping by stages. It is a combination of the Iterative and Waterfall SDLC models with the significant accent on the risk analysis. The main issue of the spiral model – is defining the right moment to make a step into the next stage. The preliminary set time frames are recommended as the solution to this issue. The shift to the next stage is done according to the plan, even if the work on the previous stage isn't done yet. The plan is introduced basing on the statistic data, received during the previous projects even from the personal developer's experience.

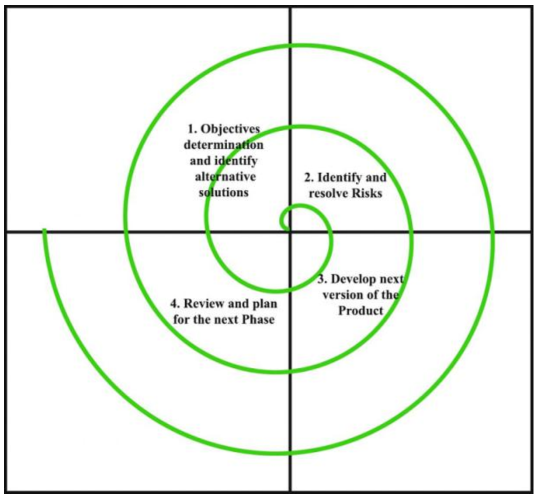
1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

Figure 3. *Spiral Model.*

Table 3. *Advantages/Disadvantages of Spiral Model.*

ADVANTAGES	DISADVANTAGES
Lifecycle is divided into small parts, and if the risk concentration is higher, the phase can be finished earlier to address the treats	Can be quite expensive
The development process is precisely documented yet scalable to the changes	The risk control demands involvement of the highly-skilled professionals
The scalability allows to make changes and add new functionality even at the relatively late stages	Can be ineffective for the small projects
The earlier working prototype is done - sooner users can point out the flaws	Big number of the intermediate stages requires excessive documentation

Use cases for the Spiral model:

Customer isn't sure about the requirements

Major edits are expected during the development cycle

The projects with mid or high-level risk, where it is important to prevent these risks

The new product that should be released in a few stages to have enough of clients feedback

3.4. V-shaped SDLC Model

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Verification: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

Validation: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

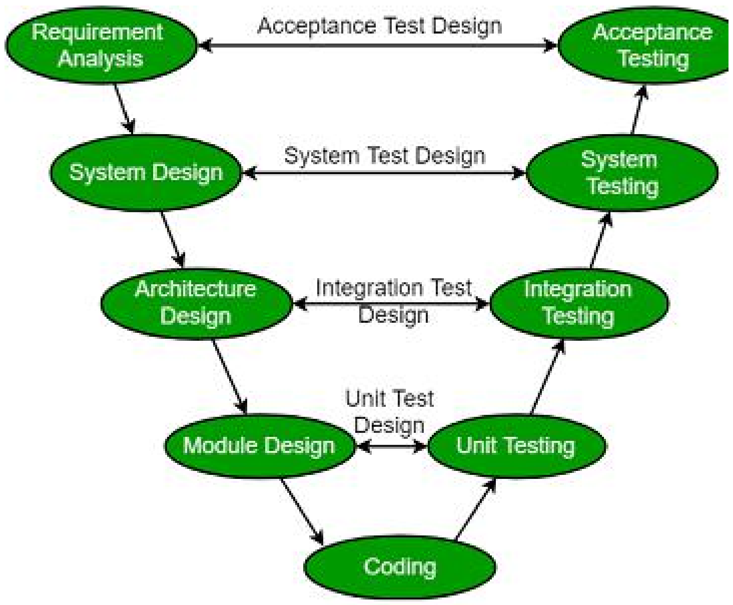

Figure 4. *V-Shaped Model.*

Table 4. *Advantages/Disadvantages of V-Shaped Model.*

ADVANTAGES	DISADVANTAGES
Every stage of V-shaped model has strict results so it's easy to control	Lack of the flexibility
Testing and verification take place in the early stages	Bad choice for the small projects
Good for the small projects, where requirements are static and clear	Relatively big risks

Use cases for the V-shaped model:

For the projects where an accurate product testing is required

For the small and mid-sized projects, where requirements are strictly predefined

The engineers of the required qualification, especially testers, are within easy reach.

3.5. Agile SDLC 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.

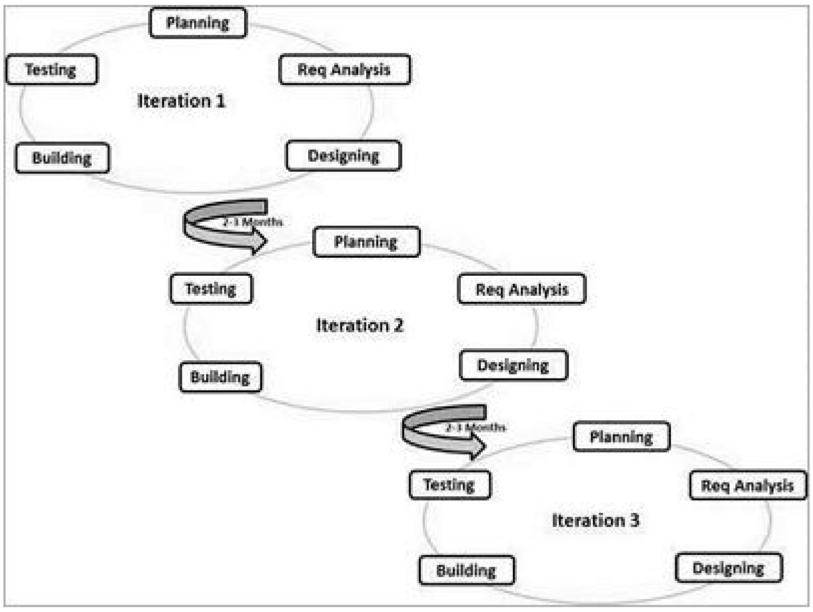

Figure 5. *Agile Model.*

Table 5*. Advantages/Disadvantages of Agile Model.*

ADVANTAGES	DISADVANTAGES
Corrections of functional requirements are implemented into the development process to provide the competitiveness	Difficulties with measuring the final cost because of permanent changes
Project is divided by short and transparent iterations	The team should be highly professional and client-oriented
Risks are minimized thanks to the flexible change process	New requirements may conflict with the existing architecture
Fast release of the first product version	With all the corrections and changes there is possibility that the project will exceed expected time

Use cases for the Agile model:

The users' needs change dynamically

Less price for the changes implemented because of the many iterations

Unlike the Waterfall model, it requires only initial planning to start the project

4. The Importance of SDLC Methodologies

It is rather unrealistic to think of a software development environment where structure and defined processes aren't needed. In an atmosphere where there are so many variables, players and things that can go wrong, it is imperative to have the help of guidelines and a "rulebook" you can use as a standard to guide you through the process of delivering a successful software application.

The importance of having and following prescribed methodologies in software development lies in the predictability of having a controlled environment for all development efforts. Software Development cycles or methodologies, in essence, are a series of stages or steps through which an organism, or in this case, a software application, passes through in a series of recurrences to reach the desired outcome. The lifecycle in software development follows the life of a software application from its inception to its maintenance, and developers need a level of control to ensure the solution is consistent with the original requirements and the release of the solution is properly managed.

Methodologies in software development are repeatable processes that can be reused as many times as necessary with a strong likelihood of delivering successful results if applied correctly.

Working under SDLC methodologies provides the opportunity to deploy solutions faster because it is a consistent, repeatable and systematic approach. Additionally, it allows organizations to respond better to market pressure and deliver high-quality business applications due to its structure and systematic nature which enables developers to work in a controlled environment.

5. Conclusion

During the years of the SDLC evolution, different models were developed from the basic cascade model to meet a huge variety of development requirements and expectations. There is no only one suitable model for all the projects, starting conditions and payment model. Even at the first sight, multi-purpose Agile cannot be used widely because of some customers' unpreparedness to scale the budget. The SDLC models often cross in the solutions and particularly look similar.

Each of these SDLC methodologies offers unique process for the variety of project challenges you will encounter in your career.

Finding the right one depends heavily on not just the expected outcome, but the parameters by which the project is executed.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this article.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

[1] Jovanovi, M. Software testing methods and techniques. *IPSI BgD Journals*, 2009, 5, 30-41.

[2] K.M.; K.R. Comparative study of automated testing tools: Testcomplete and quicktest pro. *International Journal of Computer Application*, 2011, 24, 1-3.

[3] Ian, Sommerville. Software Engineering, Addison Wesley, 7th edition, 2004.

[4] Nabil Mohammed Ali Munassar, Govardhan, A. A Comparison between Five Models of Software Engineering. *IJCSI International Journal of Computer Science*, 2010, 7(5), 1694-0814.

[5] Roger S. Pressman. Software Engineering A Practitioner's Approach. McGraw-Hill International Edition, 5th edition.

[6] Whitgift, David. Methods and Tools for Software Configuration Management. J. Wiley, 1991.

[7] Petersen K.; Wohlin C.; Baca D. The Waterfall Model in Large-Scale Development. Lecture Notes in Business Information Processing. Springer, Berlin, Heidelberg, 2009, 32.

[8] ShubhmeetKaur. A Review of Software Development Life Cycle Models. *In International Journal of Advanced Research in Computer Science and Software Engineering*, 2015, 5(11).

[9] Boehm, Barry. A Spiral Model of Software Development and Enhancement. In Proceedings of an International Workshop on the Software Process and Software Environments, 1985.

[10] DeGrace, Peter, Stahl, Leslie Hulet. *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms.* pp. 116, 117, 127. Reprinted with permission of Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[11] Coad Peter, Edward Yourdan, Object-Oriented Design, 1991.

[12] Software Management Guide, Vol. I, Software Technology Support Center, October 1993; pp. 23.

[13] Dyer, Mike. he Cleanroom Approach to Quality Software Development. 1993.

[14] Blum Bruce I. Software Engineering: A holistic View. 1992.

[15] Booch, Grady, Software Engineering with Ada, 1994; pp. 25