Software Development Life Cycle Models-A Comparative Study

Article *in* International Journal of Scientific Research in Computer Science Engineering and Information Technology · July 2020

DOI: 10.32628/CSEIT206410

3 authors, including:

Gagan Gurung
The ICFAI University, Sikkim
5 PUBLICATIONS **1** CITATION
SEE PROFILE

Rahul Shah
The ICFAI University, Sikkim
4 PUBLICATIONS **1** CITATION
SEE PROFILE

International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT206410

Software Development Life Cycle Models-A Comparative Study

Gagan Gurung*, Rahul Shah, Dhiraj Prasad Jaiswal
School of Information Technology, ICFAI University Sikkim, Gangtok, Sikkim, India

Article Info
Volume 6, Issue 4
Page Number: 30-37
Publication Issue :
July-August-2020

Article History
Accepted : 01 July 2020
Published : 07 July 2020

ABSTRACT

Software Development is one of the most powerful, vital, and the need for an hour in today's generation. Every organization, industries, small firms, institutes, etc. require the software for the functionality of their system and reducing the manual work or the traditional work, which used to be insecure and had more errors. SDLC is all about the minimization of the risk and failure and maximization of the quality of the product. To make the development works in a step by step procedure and precisely SDLC came into existence. The SDLC defines the framework that includes different activities and tasks to be carried out during the software development process. There are many types of SDLC models, which have their advantages and disadvantages and will work as per their needs.

Keywords: Software Development Life Cycle, Models, Prototype, Modeling, Development, Risk Analysis and, Comparative Analysis.

I. INTRODUCTION

SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system. Software Development Life Cycle Model is a model that describes the overall area of how software development takes place with each phase describing its way of doing. There are different types of models like the Waterfall model, V shaped model, Evolutionary prototyping model, Spiral model, Iterative and Incremental model, and agile model. Therefore, it may be required to choose the right SDLC model according to the specific concerns and requirements of the project to ensure its success. Every model has its advantage and disadvantage. It is based on how our projects are and as per the requirement, we need to implement the model.

Software development life cycle - SDLC is the systematic approach to complete the software development process within the time and maintain the quality of the software. The system development life cycle provides the set of activities to be carried out during the system development and it is often called software development life cycle. Software development is divided into a set of activities that allow any software development company to control the software product easily. Software development life cycle models use the step by step approach to complete the software development process. If the process is strong, the end product will also be strong and the project can get success. While developing the good software product the developers that are

directly or indirectly included in this process should keep the following points in their mind:

- Quality
- Process
- Methods
- Tools

The software process model is the representation of process it presents and the description of a process are as:

- Specification
- Design
- Validation
- Evolution

The software development life cycle is all about:

- Understanding the problem. i.e. (problem domain)
- Decide a plan for a solution. i.e. (solution domain)
- Coding the planed solution
- Test the actual program
- Maintain the product

II. HISTORY OF THE SDLC

The profession of "software developer" has existed since the first computers, and their operators, as far back as the days of ENIAC and vacuum tubes. Practices and methods for developing software have evolved over the decades since the invention of the computer.

Those methods have adapted to the state of the art in computer hardware, development tools, and modern thinking about the organizational management of software development teams. With this progress, new methods of software development have grown out of private and public software development efforts around the world.

These methods vary widely in approach, yet they share a common goal: to develop software as cheaply, efficiently, and effectively as possible.

III. THE SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) MODELS

A. Waterfall Model

The software development pioneer, Winston Royce in 1970, proposed waterfall. It is one of the oldest SDLC model, but not used much in recent years. It follows a linear sequential flow in which progress is seen flowing downwards while in a developing phase. Here all the requirements are to be collected at the beginning of starting a project and then move to other phases. Each phase relies on Information collected in the earlier phase as it does not allow moving to the next phase until the previous phase has been completed. The waterfall approach does not allow the process to go back to the previous phase and allow changes in it. Waterfall model is used for small projects, as there is little room for revisions once a stage is completed. In waterfall model problems can't be fixed until you get to the maintenance stage. The phases in the waterfall model include phases such as Requirement analysis, System design, implementation, testing, deployment, and maintenance.

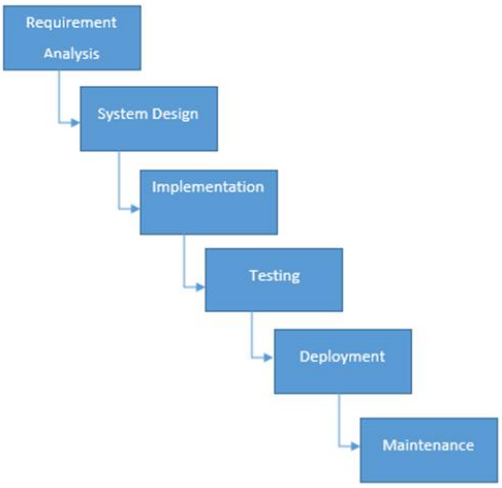

Figure 1. Waterfall Model

B. Iterative Model

Iterative model uses an iterative, which overcomes the weakness of the waterfall model. Unlike the waterfall model where the requirement was required once, iterative model requirements are gathered in every phase. The project is divided into smaller components so that the result can be utilized in the next phase. After each increment feedback is gathered from the client based on which next process is planned and modification is done. A new version of the software is produced in every phase and is repeated until the complete system is ready.

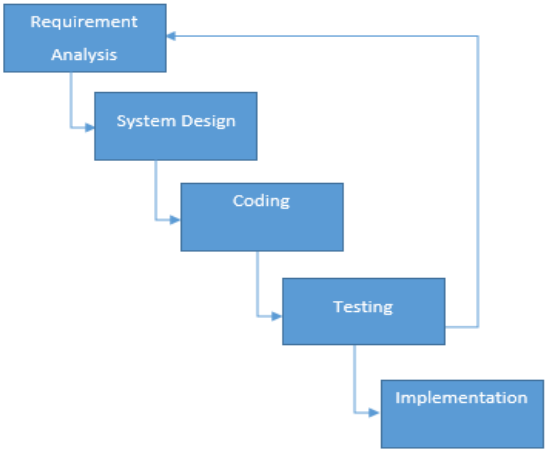

Figure 2. Iterative Model

C. V Shaped Model

The V shaped model is an extension of the waterfall model. The V-shaped model shows the relationships between each phase of development and the associated phase of testing. It's also referred to as the 'verification and validation model'. This is because each verification phase is associated with a validation phase. The main aspect of any software is to see how it performs. It needs to be tested many times. Therefore, the main focus of V shaped model is in testing. While verification focuses on development, validation ensures that all that development was carried out correctly. This process is very dynamic and a great deal of testing takes place. These steps are also referred to as the 'tester's life cycle'. This model is useful when there are no unknown requirements, as it's still difficult to go back and make changes. To every corresponding phase it has the testing phase so the model mainly focuses on testing.

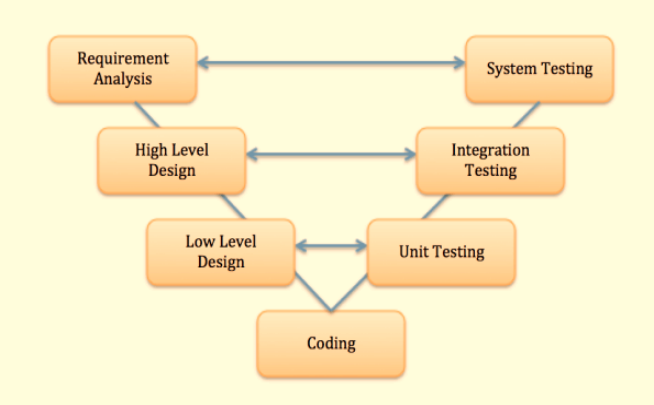

Figure 3. V Shaped Model

1) Advantages:

I. High amount of risk analysis.

II. Good for critical projects.

III. Early production.

IV. Easy to manage due to rigidity of model.

V. Easy to understand.

2) Disadvantages:

I. Not a good model for object oriented projects.

II. Not good for long and ongoing projects.

III. Not suitable were requirements have a high risk of changing.

IV. Can be a costly model.

V. Don't work well for small projects.

D. Agile Model

The Agile model was primarily designed to help a project to adapt to change requests quickly. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task agility is

required. Agility is achieved by fitting the process to the project, removing activities that may not be essential for a specific project. Also, anything that is wastage of time and effort is avoided.

Agile model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. A few Agile SDLC models are given below:

- Crystal
- Atern
- Feature-driven development
- Scrum
- Extreme programming (XP)
- Lean development
- Unified process

In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed. The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed, and deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models. Steps involve in agile SDLC models are:

- Requirement gathering
- Requirement Analysis
- Design
- Coding
- Unit testing
- Acceptance testing

The time to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.

Principles of Agile model:

To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.

Agile model relies on working software deployment rather than comprehensive documentation.

Frequent delivery of incremental versions of the software to the customer representative in intervals of a few weeks.

Requirement change requests from the customer are encouraged and efficiently incorporated.

It emphasizes on having efficient team members and enhancing communications among them is given more importance. It is realized that enhanced communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.

It is recommended that the development team size should be kept small (5 to 9 peoples) to help the team members meaningfully engage in face-to-face communication and have a collaborative work environment.

Agile development process usually deploys Pair Programming. In Pair programming, two programmers work together at one work-station. One does coding while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so.

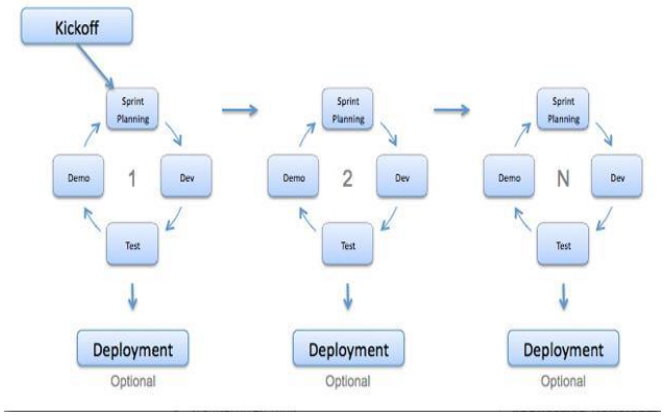

Figure 4. Agile Model

1) Advantages:

I. Working through Pair programming produce well written compact programs that have fewer errors as compared to programmers working alone.

II. It reduces the total development time of the whole project.

III. Customer representatives get the idea of updated software products after each iteration. Therefore, it is easy for him to change any requirement if needed.

2) Disadvantages:

I. Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.

II. Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

E. Prototype Model

The prototype or prototyping model is a software development model that is used to make a prototype version of the software. This model is used when the customer or user doesn't have any specific or detailed information about the product. In this model, the developer can start developing software with minimal information and requirements then take the feedback from the user and make the changes according to the user and redefine the product. It is an iterative, trial, and error process which takes place between the developer and the client. The main goal of this model is to provide a system with overall functionality so that a client can check and give the required changes to be made. In this model we have six different phases those includes requirements, quick design, build a prototype, user evaluation, refining prototype, implement and maintain as shown in the figure:

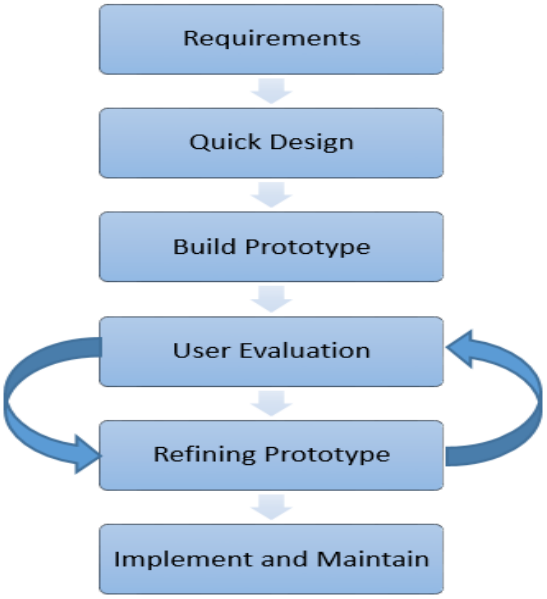

Figure 5. Prototype Model

There are four different approaches in this model that can be used by the developers based on their need:

1. Rapid Throwaway Prototypes – This model is used to get an instant feedback, ideas and changes from the user for the software by developing a quick prototype and discarded and may not be the part of the final product.

2. Evolutionary Prototype – In this model, we can make multiple versions of the prototype based on the customer feedbacks and it can save the time and efforts of a developer who has developed a software from scratch.

3. Incremental Prototype – In this model the final product is divided into multiple small parts and then the prototype of them developed and at the end all integrated into one so that the user evaluation time can be less.

4. Extreme Prototype - This model is used specifically for web development. All web prototypes are built in an HTML format with a services layer and are then integrated into the final product.

A. Spiral Model

The Spiral model is one of the most important Software Development Life Cycle models, which is used for risk management that combines the waterfall model and iterative model. The spiral model was first mentioned by Barry Boehm in his 1986 paper. In this model, every phase starts with a design goal and ends with the client reviewing the progress. This model is used for large projects which involve risk and cost on every changes. The spiral model has four different phases includes planning, risk analysis, engineering, and evaluation as shown in the diagram:

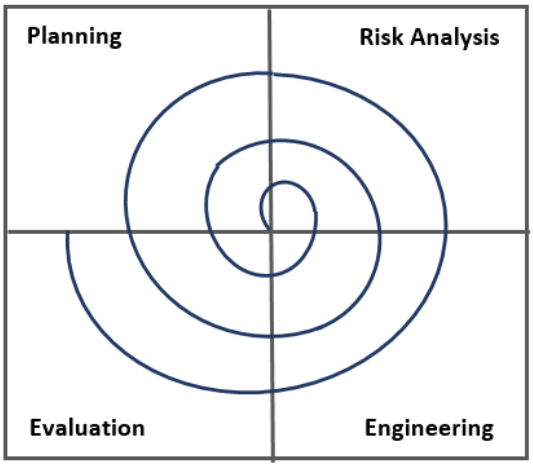

Figure 6. Spiral Model

IV. BENEFITS OF SDLC

It is very difficult to carry out a complex, team effort such as software development without some kind of plan. Each software development methodology (several will be detailed below) is a plan framework for how to develop software. There is much debate about which method is best overall, which is best suited to a particular type of software, and how to measure success in software development. One thing, however, is certain: any plan is better than no plan.

Without some kind of structured plan, software development teams tend to devolve into a "herd of cats." Developers don't know what they're supposed to create. Project managers have no idea how much progress is made towards the completion of a project. Without a plan, the business doesn't even have a way to decide whether the final product meets their requirements.

A formally defined method for software development in the form of the SDLC achieves several benefits:

- A common vocabulary for each step
- Defined communication channels between development teams and stakeholders

- Clear roles and responsibilities among developers, designers, business analysts, and project managers
- Clearly-defined inputs and outputs from one step to the next
- A deterministic "definition of done" that can be used to confirm whether a step is truly complete.

V. CONCLUSION

SDLC is a systematic process for building software that ensures the quality and correctness of the software built. The structure imposed by this SDLC is specifically designed to maximize the probability of a successful software development effort. It consists of a detailed plan which explains how to plan, build, and maintain specific software. All software begins as a concept and flows through a series of phases until a release is developed and deployed. The Software Development Life Cycle of an application or system continues, with updates and new features, until the day it is decommissioned or replaced. Several methods for software development have evolved over the decades.

VI. REFERENCES

[1] https://www.geeksforgeeks.org/software-engineering-prototyping-model/
[2] https://economictimes.indiatimes.com/definition/prototype-model
[3] https://searchcio.techtarget.com/definition/Prototyping-Model
[4] https://www.guru99.com/software-engineering-prototyping-model.html
[5] https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm
[6] https://www.w3schools.in/sdlc-tutorial/spiral-model/
[7] https://www.geeksforgeeks.org/software-engineering-spiral-model/
[8] https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html
[9] https://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm
[10] https://www.softwaretestinghelp.com/spiral-model-what-is-sdlc-spiral-model/
[11] https://raygun.com/blog/software-development-life-cycle/

AUTHORS PROFILE

Mr. Gagan Gurung did his Bachelor of Computer Application from The ICFAI University, Sikkim in the year 2012 and did his Master of Computer Application from Bangalore University in the year 2015. Currently he is working in The ICFAI University Sikkim from the year 2016 in the Department of Information Technology and has been publishing a various paper in different national and international journal. His area of interest is in the field of Data Mining, Software Engineering and Artificial Intelligence.

Mr. Rahul Shah has received his Master of Computer Application (MCA) degree from Shri Ramaswamy Memorial University, Sikkim in 2017. He is currently working as a Lecturer at ICFAI University, Sikkim. He has 3+ years of teaching experience for both undergraduate and postgraduate students. His current field of interest includes Android application development, Artificial Intelligence, Cloud computing, and Web Engineering.

Mr. Dhiraj Prasad Jaiswal has received his Master of Computer Application (MCA) degree from Shri Ramaswamy Memorial University, Sikkim in 2017. He is currently working as a Lecturer at ICFAI University Sikkim. He has 2+ years of teaching experience for both undergraduate and postgraduate students. He has also worked as a software developer at Cubiq Innovation and has 1+ year of experience. His current field of interest includes Android application development, Internet of Things, Java Programming, and Cloud computing.

Cite this article as :

Gagan Gurung, Rahul Shah, Dhiraj Prasad Jaiswal, "Software Development Life Cycle Models-A Comparative Study", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6, Issue 4, pp.30-37, July-August-2020. Available at
doi : https://doi.org/10.32628/CSEIT206410
Journal URL : http://ijsrcseit.com/CSEIT206410