International Journal of Research in Engineering, Science and Management
Volume 5, Issue 3, March 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

A Comparative Study of Various Software Development Life Cycle (SDLC) Models

Neha Dwivedi[1*], Devesh Katiyar[2], Gaurav Goel[3]
[1]MCA Student, Department of Computer Science, Dr. Shakuntala Mishra National Rehabilitation University, Lucknow, India
[2,3]Assistant Professor, Faculty of Computer & Information Technology, Dr. Shakuntala Mishra National Rehabilitation University, Lucknow, India

***Abstract*: In the software industry, the software development life cycle (SDLC) is used to design, develop, and produce high-quality, dependable, cost-effective, and on-time software products. This is also known as a model of the software development process. There are several SDLC process models to choose from. In this work, I've attempted to define various SDLC models in terms of their greatest applications. There have been numerous papers written in this area. In this work, I shall also make advantage of their information or findings. The major goal of this paper is to explain some of the most prominent SDLC models, such as Waterfall, Iterative, Spiral, V-Model, Big Bang, Agile, Rapid Application Development Model, and Software Prototype. The major goal of this paper is to describe the benefits and drawbacks of each option.**

***Keywords*: Risk analysis, software development life cycle.**

1. Introduction

We now live in a world where computers are required. A computer is responsible for 75% of a person's life. To keep up with the ever-increasing demands of the digital world, one must use computers, laptops, or personal computers. So, we need software to work on when we utilize the computer. Everything in a computer operates on the principle of the software on which it is based. So, in order to construct the appropriate software, we'll need the models.

2. Software Development Life Cycle

The abbreviation for Software Development Life Cycle (SDLC) is Software Development Life Cycle. When designing, planning, and maintaining a software project, it is a process that guarantees that all functionalities, as well as user requirements, objectives, and end goals, are accomplished. SDLC raises the quality of software programs and the software development process as a whole.

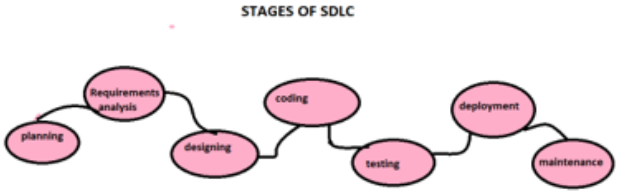

Fig. 1. Stages of SLDC

In the software industry, the SDLC is a method for designing, producing, and testing high-quality software. The SDLC is intended to provide high-quality order to meet and exceed customer requirements while staying on schedule and within budget.

There are seven stages to the SDLC Process. The steps include planning, gathering and analysing needs, designing, coding, testing, deployment, and maintenance.

Stage 1: Planning

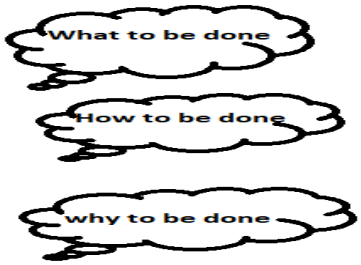

Fig. 2. Planning

The very first phase of the SDLC is "What do we want?" The team calculates the cost and describes the new program's requirements during planning, which is an important part of the software distribution lifecycle.

Stage 2: Requirement Gathering and Analysis

The SDLC's second step is gathering as much info as possible from the product's customer needs. Go over every detail and specification of the product with the consumer. The development team will then look over the requirements while keeping in mind the software's design and coding. Evaluate the validity of these needs and whether or not they can be incorporated into the software system. The main goal of this step is to ensure everybody is aware of every element of the requirement.

Stage 3: Designing

During the design phase, the programmer checks if the generated software fits all of the end-requirements. User's (3rd phase of the SDLC). Also, whether or whether the customer's project is technically, legally, and financially feasible. The developer selects the proper programming languages for the project, such as Oracle, Java, and others, after deciding on the best design plan.

*Corresponding author: narayanidwivedi25@gmail.com

After the design specification is ready, all stakeholders will review it and provide feedback. At all times, stakeholder input must also be gathered and integrated into the document, as even a tiny error could result in a budget increase.

Stage 4: Coding

choosing to start building the full system. During the coding phase, tasks are divided into sections or during this phase, developers write code in the scripting language of their modules and assigned to different developers. The Software Development Life Cycle's most time-consuming step.

Stage 5: Testing

After the software has been produced by the developers, it is deployed in the testing environment. The testing team then tests the system's functionality. The fifth phase of the SDLC is testing, which ensures that the entire application satisfies the customer's expectations.

Stage 6: Deployment

Once the test is complete and the products is ready for delivery, it is made available to clients. The scale of the project determines the difficulty of the deployment. The users are then provided training or guides to help them use the software. A final rounds production testing is carried out to ensure that there is no environmental impacts or effects from the new product.

Stage 7: Maintenance

When the user begins to utilize the designed system, the true problem appears, and must be addressed on a routine basis. In the seventh step of the SDLC, maintenance, the developed product is looked after. To stay up with changing user end situation or technology, the software is updated regularly.

3. SDLC Models

The following are a few examples of software development process models.

1) Waterfall model

The waterfall model is the most basic model of software development. It specifies that all phases of the SDLC will run in a sequential order. That is, after the first phase has been completed, the second phase will begin, and so on.

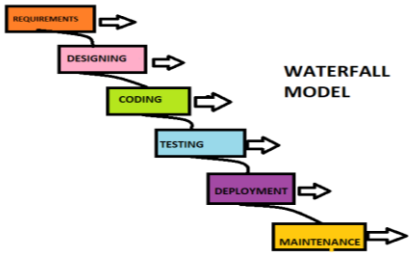

Fig. 3. Waterfall model

2) Incremental model

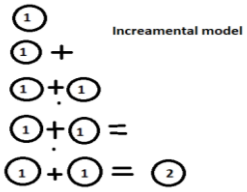

Fig. 4. Incremental model

The incremental approach divides software into independent modules (components)/increments, with each module having its own set of SDLC processes, such as requirements gathering, analysis and so on.

When a component is ready, it is delivered to the customer, and when the remaining components are ready, they are delivered to the client one at a time, merging new and old components.

3) Spiral model

The spiral model is a Software development life cycle model that combines both iterative and waterfall models, in which a product starts with a small set of requirements and then goes through the development of that simple product to meet those requirements, and is used when more frequent releases are required. Additional functionality will be added to the product based on additional requirements in an ever-increasing spiral till the product is ready for production.

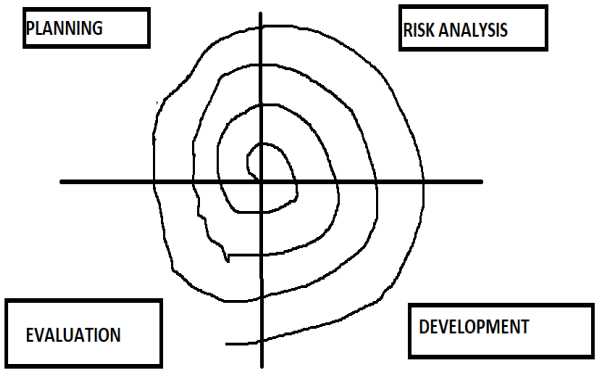

Fig. 5. Spiral model

4) V-model

The V-Model stands for verification and validation model. Each phase must be completed, before going on to the next step of the SDLC, it follows the progressive design methodology of the waterfall paradigm. The testing of the device will take place at that time as its development.

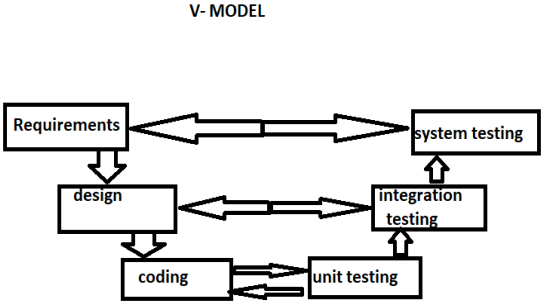

Fig. 6. V-model

Verification: It entails a review that is carried out without the use of any code. It is the process of evaluating the product development to see if the requirements given are met.

Validation: It entails the use of dynamic analysis methods (functional and non-functional), as well as code execution for testing. Validation is the process of classifying software after it has been developed to evaluate whether it satisfies the

Table 1
Comparative study of SDLC models

Models	Advantages	Disadvantages
Waterfall model	1. Presence of a clear structure. 2. Smooth transfer of transfer. 3. Easy to manage. 4. Early determination of goals. 5. Extremely stable 6. Follows a strict timeline.	1. Costly and inflexible 2. Does not prioritize the client or end user 3. Delayed testing 4. No scope for revision or reflection.
Incremental model	1. Generates operating software package quickly and early throughout the software package life cycle. 2. This model is additional versatile – more cost effective to vary scope and necessities. 3. It is easier to check and correct throughout a smaller iteration. 4. In this model client will answer every designed. 5. Lowers initial delivery value. 6. Easier to manage risk as a result of risky items are known and handled throughout it'd iteration.	1. Needs sensible coming up with and style. 2. Needs a transparent and complete definition of the complete system before it is de-escalated and engineered incrementally. 3. Total price is more than waterfall.
Spiral model	1. High quantity of risk analysis therefore, turning away of Risk is increased. 2. Good for big and mission-critical comes. 3. Strong approval and documentation management. 4. Additional practicality is additional at a later date. 5. Software is made early within the computer code life cycle.	1. Can be a expensive model to use. 2. Risk analysis needs extremely specific experience. 3. Project's success is extremely passionate about the chance analysis section. 4. Doesn't work well for smaller comes
V-model	1. Simple and straightforward to use. 2. Testing activities like coming up with, take a look at coming up with happens well before committal to writing. This protects a great deal of your time. Therefore, higher likelihood of success over the body of water model. 3. Proactive defect pursuit – that's defects are found at early stage. 4. Avoids the downward flow of the defects. 5. Works well for tiny comes wherever necessities are simply understood.	1. Very rigid and least versatile. 2. Software is developed throughout the implementation section; therefore, no early prototypes of the computer code are made. 3. If any changes happen in midway, then the take a look at documents at the side of demand documents should be updated.

Table 2
When we use models

Waterfall model	Incremental model	Spiral model	V-model
• Needs are well-defined and not subject to change. • There are no ambiguous prerequisites (no confusion). • This approach is useful when the system is completely understood. • The film is short and has a small cast. • The threat is either non-existent or insignificant.	• The system's requirements are well understood. • When there is a demand for early release of a product. • When the software engineering crew is not well-trained or skilled. • When high-risk characteristics and objectives are involved. • Web application and product-based businesses are more likely to employ this model.	• When working on a huge project, a spiral model is utilized in software engineering. • Spiral approach is utilized when frequent releases are necessary. • When it is appropriate to create a prototype. • When it's crucial to assess risk and costs • Spiral approach is excellent for projects that are medium to high-risk. • Spiral model in SDLC is beneficial when requirements are vague and complex. • Changes may be necessary at any time.	• When there is a clear and unambiguous requirement. • The V-shaped model should be utilized for small to medium-sized projects with well-defined and fixed requirements. • When sample technical resources with needed technical skills are available, the V model should be chosen.

expectations and needs of the customer.

As a result, the V-Model has Verification stages on one side and Validation phases on the other.

4. Conclusion

Software development models include the Waterfall, Prototype, Spiral, and V-Shaped models, among others. According to customer needs, the software development team will also have to determine which model to apply for their project. All of these models have their own set of benefits and drawbacks. The waterfall approach is used by an organization that wants to develop a product in a linear manner. The most important thing for businesses to remember is that they should create a "bug-free" product before putting it on the market. Some software development products may require the fusion of multiple models.

References

[1] K. K. Aggarwal, Yogesh Singh, "Software Engineering," 3rd Edition.
[2] www.google.com/amp/s/www.geeksforgeeks.org/software-development-life-cycle-sdlc/
[3] www.tryqa.com.
[4] http://www.tutorialspoint.com/sdlc/