What Is Software Development Life Cycle (SDLC)?

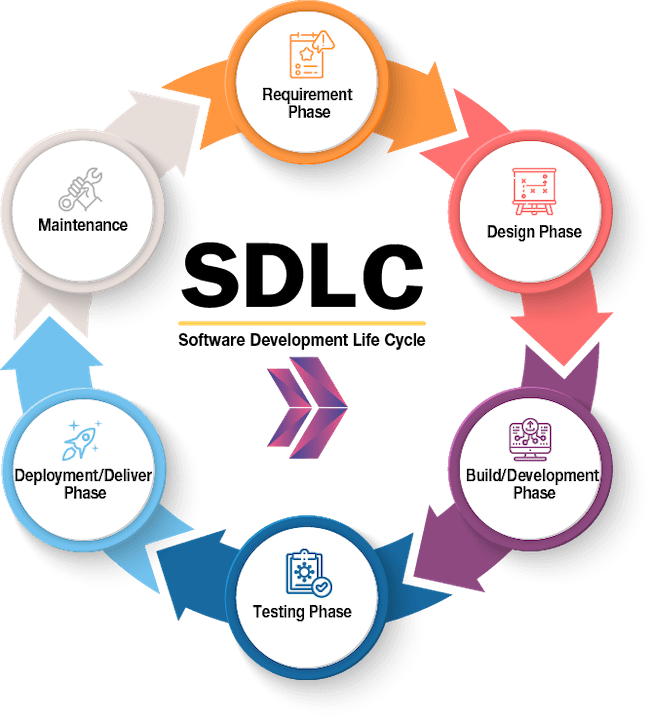
Oct 4, 2021 | [Full Stack Development](https://clarusway.com/full-stack-develop/) | [1 comment](https://clarusway.com/what-is-software-development-life-cycle/#respond)

Have you ever heard about **SDLC**? If you are new to the IT industry or a newbie in your IT career, you might be curious about how all that software is developed. The answer leads us to SDLC. SDLC stands for “**Software Development Life Cycle**.” As the name suggests, it is a process for developing software. In other words, SDLC is a structured way to create and develop high-quality software. The process starts from planning but not finishes after the delivery of the product. As you may understand from the name (software development life cycle), SDLC is not a linear project management framework. Instead, it becomes a cycle where feedback is made, problems are solved through repetitions, or new features are acquired. **SDLC** process consists of phases like ***planning, designing, implementing, testing, deploying, and maintenance***. The stages of the software development life cycle are which describe how to develop and maintain particular software. Each phase has different processes and activities.

Why is SDLC Important?

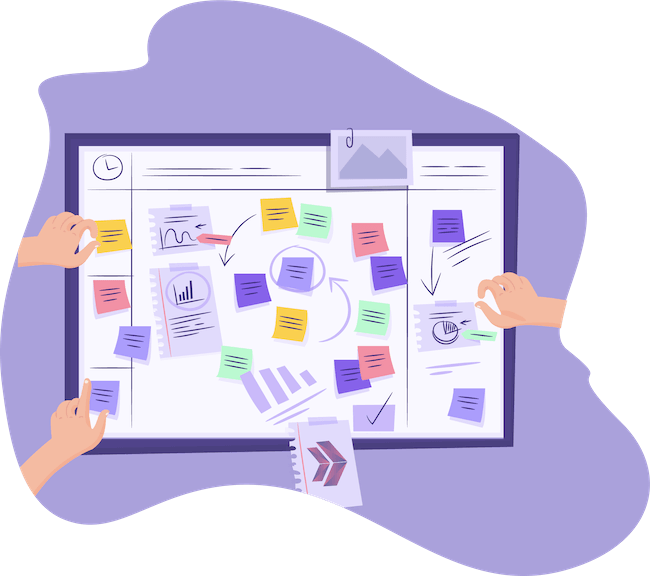
SDLC consists of a plan that describes how the software will be developed, maintained, and improved. The software is actually a product, and that there is a life process in that product. The purpose of SDLC is to produce software with high quality that meets customer needs. **Software development life cycle** reduces the hassle of developing a new product. It’s important to have an **SDLC** in place as it helps to transform a product from a design state into a complete product. This production process consists of phases that follow each other. As the requirements for software functions are constantly changing and expanding, these phases are handled in a continuous loop. At any stage in the cycle, it is possible to go back and forward again. This helps the development team to produce high-quality software.

What are the SDLC Phases?

SDLC consists of time-based and content-divided phases. Every phase needs to be deliverable from the previous step. For example, design gets its inputs from the requirement phase. In the build phase, developers use the designated product template to develop the software. The SDLC process consists essentially of the following phases:

* Requirement Phase
* Design Phase
* Build/Development Phase
* Testing Phase
* Deployment/Deliver Phase
* Maintenance

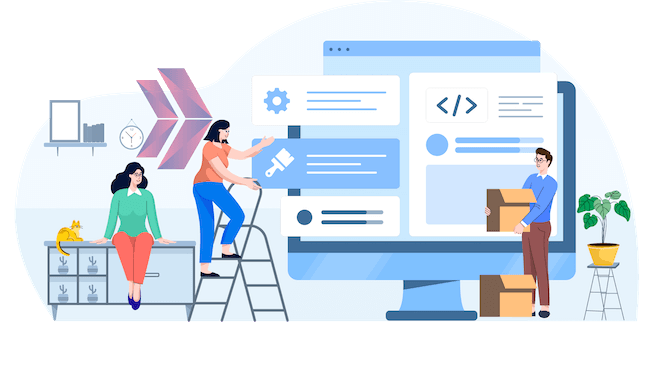
**Requirement Phase:**

**The requirement phase** is the first and the most critical stage of SDLC for both the developing team and the project manager. In this phase, the client or stakeholders specify all the information about the product they want. The product features, specifications, expectations, and all other requirements related to the product or software. The business manager or project manager gathers all of this information and also prerequisites. Here are some examples:

* What customer wants
* Who is the end-user
* Product’s target group
* Opportunities and risks

All data collected in this phase is critical in shaping the software according to customer needs. To develop the software system, you should have a clear understanding of the desired product/software. Actually, the desired product is in the customer’s mind, and because of this fact, the communication between customer and product owner is vital. The next step is preparing a document that includes all product requirements. This document is called [SRS](https://medium.com/analytics-vidhya/what-is-srs-software-requirement-specification-document-and-its-requirements-b57bf87bb868) (Software Requirement Specification) and must be approved by the business owner, stakeholders, and the customer. The final SRS has all details about the product to be developed with requirements. So we can say the SRS is an agreement between customers and software developers. It is also a checklist for the developers in creating the software.

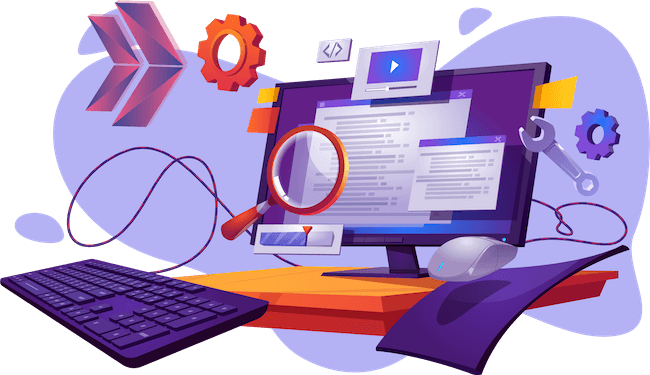
**Design Phase:**

**The design phase**is the second step of the SDLC process. As mentioned before, the outputs of the requirement phase will be used as the inputs in this step. The main output of the requirement phase was the SRS document, so that it will be used as a base document in the design phase. It would not be wrong to call this stage the highest priority stage because this is the stage where the system is transformed from logical design to physical design. This design also reveals the tools with which the software will be developed. For example, which programming language will be used, such as Python, Java, .NET, or PHP? Which database to be worked, such as Oracle, MySQL, Postgres, relational or non-relational database? All those questions and more need to be answered to create a detailed design since all of the back-end and front-end programming tools must work smoothly together without bugs.

**Build/Development Phase:**

**The build/development phase** is also known as **the coding phase**. Developers start coding in this stage. In this phase, the smallest piece of software is called a **unit**. Units make up **modules**. Therefore, developers develop units and modules step by step in a weekly or monthly work plan. Additionally, the coding phase is the longest phase of the SDLC process, and it requires a focused approach for the developers.

**Testing Phase :**

Once the software is complete, it is time for **the testing phase**. This phase is where you focus on investigation and discovery. Then, the testing team starts testing the functionality of the entire system. This is done to verify whether the software works and gives the result as per the requirements addressed in the requirement phase. The development team makes a test plan. This test plan includes;

* Integration testing
* System testing
* Acceptance test

If there is a bug/defect detected in the software, it is not working as expected. The testing team gives detailed information to the development team about the issue. If the defect is valid or worth fixing, it will be fixed, and the development team will replace it with the new one.

**Deployment/Deliver Phase :**

After completing all tests successfully, the product will be delivered to the customer. Now, customers should do beta testing for the product after the delivery. Beta, or beta version, refers to the system testing and vulnerability testing of the software’s initial release. This release aims to run the software smoothly under different hardware. After the customer starts testing the software, the necessary checks are made to correct it if any problem is encountered. **Maintenance Phase :** The last phase of the process SDLC is **the maintenance phase**, where the process continues until the software’s life cycle ends. The product alone is not sufficient as a delivery output. Also, user manual and version difference documents should be created for end-users. Additionally, the maintenance phase begins with the delivery. There are different maintenance activities such as debugging, preventing, improving the infrastructure, adding new features to the product.

**The Most Known SDLC Models**

An SDLC model describes the types of activities performed in a software development project at each stage and how the activities relate logically and chronologically to each other. There are many different [SDLC models](https://www.forbes.com/sites/forbestechcouncil/2021/08/10/13-tech-leaders-share-their-favorite-software-development-life-cycle-methodologies/?sh=260d3f9e6157), each of which requires different approaches. We can specify models under two mainframes. The first one is traditional models; the second one is modern models. The traditional project management approaches can be called linear, that each stage of the SDLC should be completed before stepping next stage. They are documentation-oriented. Modern approaches can be defined as models that are more flexible and open to change. Communication channels are always kept open to meet customer expectations fully. **Most common traditional SDLC models:**

* Waterfall Model
* V-Shaped Model
* Spiral Model

**Modern SDLC models:**

* Agile Model
* Lean Model
* Extreme Programming

In this article, we want to mention two main models. The first one is **the waterfall model**, which is a traditional one, the other is **the agile model**, which is a modern model.

**Waterfall Model**

The waterfall model can be defined as a solid model. In this model, each stage continues step by step like a waterfall. It flows linearly and sequentially. It is also known as the Linear-Sequential SDLC model. The Waterfall Model is the earliest approach and a widely accepted SDLC model used for software development to ensure the success of the specific type of project. In the waterfall model, strict documentation of customer requirements before the beginning of the project may create problems in the future. As customer requirements may change, it may be inevitable for the software system to undergo structural changes. Such a situation might increase the cost because the phases in the model may need to be implemented several times. On the other hand, **the waterfall model** can be called a disciplined approach because it directs the team from the beginning till the end of the project in every step.

**Agile Model**

Software projects face the risk of failure during the life cycle due to misuse of resources, wrong prioritization, and inaccurate understanding of customer demands. This has led companies to experiment with a variety of new methods to produce more productive projects. Agile is a new approach to project management methods resulting from these experiments. This method is based on an incremental and iterative approach that helps teams deliver a product to their customers faster and more desirable. In this respect, the framework of agile promises significant advantages over traditional methods. In recent years, most companies have adopted the agile approach, which they find more successful than traditional methods. In this context, we can say that almost every software developer has used the agile method in some form.

**The Agile Manifesto**

In 2001, 17 independent Software Leaders met in America for brainstorming. They aimed to find out how to develop software better by using different knowledge and approaches. After two days of brainstorming, they released the **[Agile Manifesto](https://agilemanifesto.org/" \t "_blank)**. There are four values in the manifesto. Individuals and interactions, working software, customer collaboration, and responding to the changes are the values. Stemming from these values, there are 12 principles in agile. These principles can be summarized as satisfying the customer, welcoming changing requirements, good cooperation between business people and developers (working together), face-to-face conversation, motivated individuals, and simplicity.

**Agile Modeling**

Agile is not a prescriptive process, unlike the waterfall model. Agile methods break the product into small incremental builds. In addition, the model provides a light framework for development teams. That helps them focus on fast delivery while maintaining functionality. Also, the agile software development model is highly flexible, has a low error rate, is fast, and offers inexpensive solutions.

**Agile Methods**

There are several different methodologies used to implement the **agile model**. The two most common agile methodologies are Scrum and Kanban.

●      ***Scrum***

Scrum is the most common methodology to implement Agile. It is an iterative development model used in complex software development processes. In scrum, larger projects are divided into smaller parts to manage them with sprints. Sprints are the periods from one to four weeks. It can be even a few days when needed. Continuous sprint length reduces variability; a scrum team can safely predict what they can do on each sprint based on what they have done in previous sprints.

●      ***Kanban***

Kanban is another popular methodology for implementing the agile system in the business. The word “kanban” is in Japanese and can be translated as “***the card you can see***” or “***signboard***“. Kanban focuses on visualizing the workflow process. The primary goal of kanban is to identify potential flaws in the process and correct them to ensure that works progresses at the desired pace. When a bottleneck is detected in the flow, action is taken against it, and improvements are made to prevent it from recurring. There are no short-term targets set. Development continues continuously. At Clarusway, you will have a chance to learn details of the **SDLC process** by participating in project-based group study methods once you attend [full-stack web development](https://clarusway.com/full-stack-developer/), [front-end development](https://clarusway.com/front-end-development/), or [back-end development](https://clarusway.com/back-end-development/) courses.