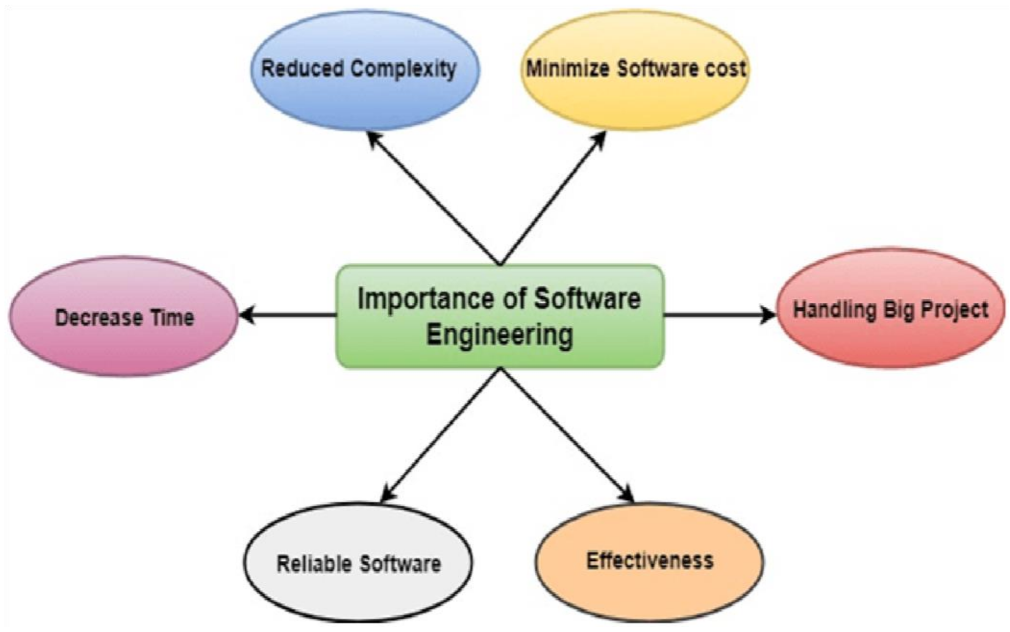
Importance of Software Engineering



The importance of Software engineering is as follows:

1. **Reduces complexity:** Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. Software engineering **divides big problems into various small issues**. And then start **solving each small issue one by one**.
2. **To minimize software cost:** Software needs a lot of hard work and **software engineers are highly paid experts**. **A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed.** In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.
3. **To decrease time:** Anything that is not made according to the project always wastes time. And if you are **making great software**, then you may need to **run many codes** to get the definitive running code. This is a **very time-consuming procedure**, and if it is **not well handled**, then this **can take a lot of time**. So if you are **making your software** according to the **software engineering method**, then it will **decrease a lot of time**.
4. **Handling big projects:** Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires a master planning, direction, testing, and maintenance. So to handle a big project without any problem, the company has to go for a software engineering method.

5. **Reliable software:** Software should be secure, means if **you have delivered the software, then it should work for at least its given time** or subscription. And **if any bugs** come in

the software, the **company is responsible** for **solving all these bugs**. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.

6. **Effectiveness:** Effectiveness **comes if anything has made according to the standards**. Software standards are the big target of companies to make it more effective. So **Software becomes more effective in the act with the help of software engineering.**

What is SDLC?

SDLC stands for **Software Development Life Cycle** and is also referred to as the **Application Development life-cycle**. SDLC is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.

Need of SDLC

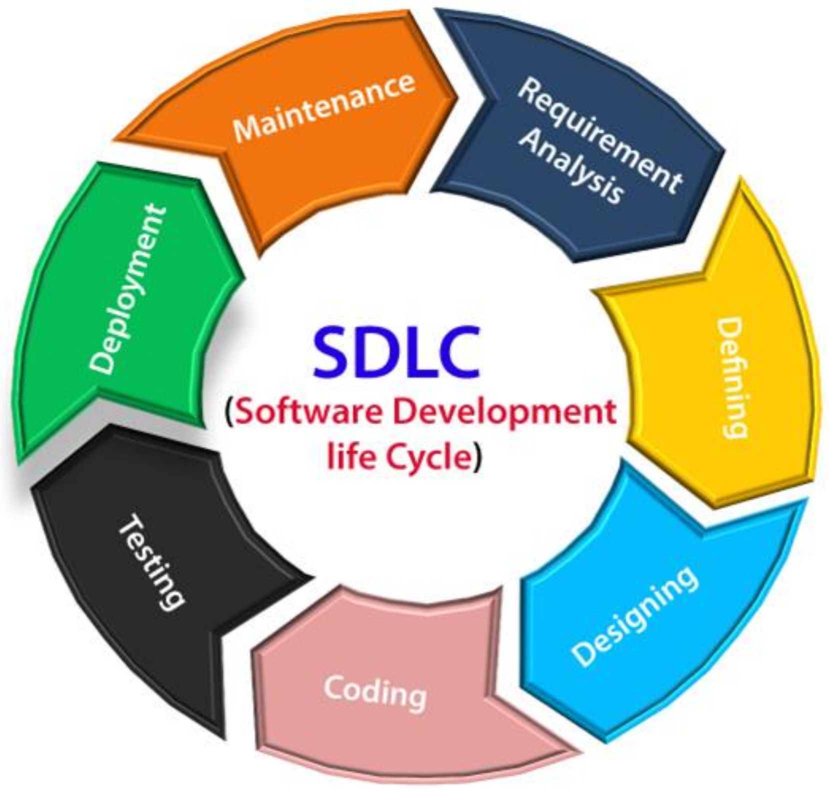
The development team must determine a suitable life cycle model for a particular plan and then observe to it.

Without using an **exact life cycle model**, the development of a software product **would not** be in a **systematic and disciplined manner**. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do. Otherwise, project would be failed.

A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

SDLC Cycle

SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



The stages of SDLC are as follows:

Stage1: Requirement analysis

Requirement Analysis is the most important and necessary stage in SDLC. The senior members of the team perform it with inputs from all the stakeholders.

Business analyst and **Project organizer set up a meeting** with the client to **gather all the data** like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

For Example, A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Stage 2: Defining Requirements

Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.

Stage 3: Designing the Software

The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project.

Stage 4: Coding

The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

Stage 5: Testing

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

Stage 6: Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.

Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.

Stage7: Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

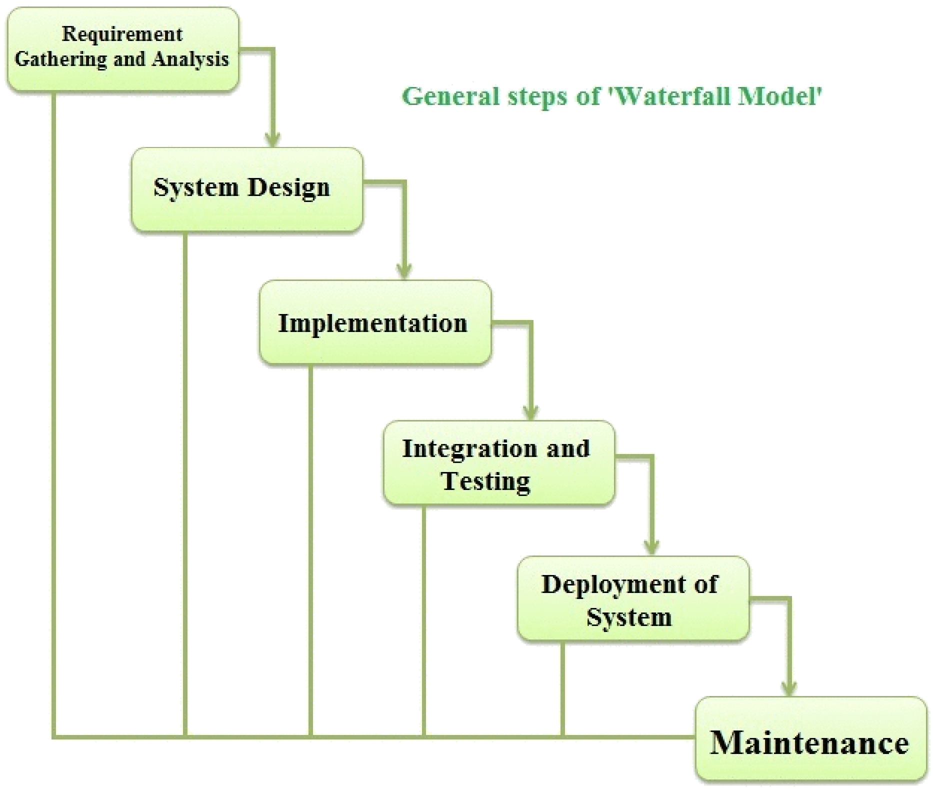
Some SDLC Models:

- Waterfall Model
- V-Model
- Spiral Model
- Iterative Model (Incremental and Evolution)
- Prototype Model, etc.

- **<u>Waterfall Model:</u>**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.



The sequential phases in Waterfall model are −

- **<u>Requirement Gathering and analysis</u>** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **<u>System Design</u>** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.