Software Development Lifecycle (SDLC): is a systematic process to be followed for a software project.
A structured way to create and develop software.
The main purpose of why corporations implement SDLC is to produce software with the highest quality and lowest cost in the shortest time.
There are various phases within SDLC, and each phase has its own different process and activity. This helps the development team to design, create and deliver a high-quality product.
Every phase in a life cycle of software development needs to be deliverable from the previous phase.

1- Requirement Phase: is the first and the most critical phase of SDLC for both the developing team and the project manager. During this phase, the client specifies requirements, specifications, expectations etc. related to the product or software. The manager gathers all of this information and also prerequisites.
All the information is crucial to developing the product as per the customer requirements.
To develop the software system we should have a clear understanding of the desired product/software. To achieve this we need to continuous communication with customers to gather all requirements.
The next step is to clearly define and document the product requirements and get them approved by the customer or the market analysts.
This is done through an SRS (Software Requirement Specification) document. It consists of all the necessary requirements to be designed and developed during the project life cycle.

“SRS is a detailed description of a software system to be developed with requirements. The SRS is developed based on the agreement between customers and contractors. It may include the use cases of how a user is going to interact with a software system.”
SRS Document Structure:

2- Design Phase:
The requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.
This is the high priority phase in a system's development life cycle because the logical designing of the system is converted into physical designing. The output of the requirement phase is a list of things that are required and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as programming language like Java, .NET, PHP; database like Oracle, MySQL; a combination of hardware and software to provide a platform on which software can run without any problem is taken in this phase.
There are several tools and techniques used for describing system design, such as Flowchart, Data flow diagram (DFD), Data dictionary, Structured English, Decision table, and Decision tree.

3- Build/Development Phase:
After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system.
This phase is also known as coding phase.
Developers start to build the entire system by writing code using the chosen programming language.
Work/task is divided into small units or modules, and coding starts by the team of developers according to the design and the requirements of the client to produce the desired result.
Coding Phase is the longest phase of the SDLC process, and it requires a more focused approach for the developer.

4- Testing Phase:
Once the software is complete, it is the time for the testing phase.
This phase is where you focus on investigation and discovery. The testing team starts testing the functionality of the entire system. This is done to verify that the software works and gives the result as per the requirements addressed in the requirement phase or not.
The development team makes a test plan to start the test. This test plan includes all types of essential testing such as integration testing, unit testing, acceptance testing, and system testing.
If there is a bug/defect detected in the software, or 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 replaces it with the new one. It also needs to be verified.

5- Deployment/Deliver Phase:
When software testing is completed with a satisfying result and there are no remaining issues in the working of the software, it is delivered to the customer.
As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customers can require any changes which are not present in the software but mentioned in the requirement document to make it more user-friendly.
Besides this, if any type of defect is encountered while a customer using the software, the development team will be informed to fix this problem. If it is a critical defect, the development team solves it in a short time. Otherwise, it will wait for the next version.
After the solution of all types of bugs and changes, the software finally deployed to the end-user.

6- Maintenance Phase:
The last phase of the process SDLC is the maintenance phase where the process continues until the software's life cycle comes to an end. When a customer starts using software, actual problems start to show up. At that time, there's a need to solve these problems.
Maintenance Phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve its performance, enhance security features and address customer's requirements.

SDLC Models

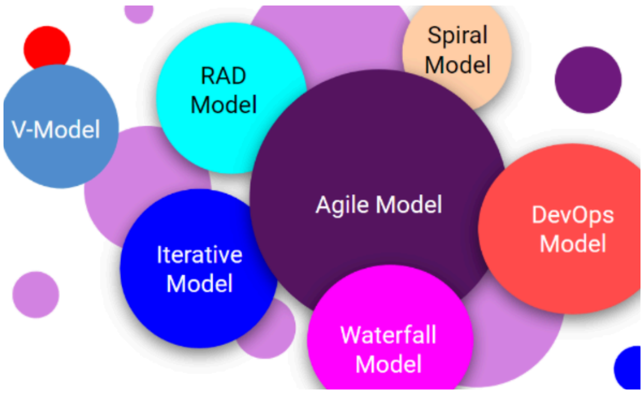
A SDLC model describes the types of activities performed in a software development project at each stage, and how the activities relate logically and chronologically to each other.
There are many different SDLC models, each of which requires different approaches to testing.

Software Development and Software Testing:
It is an important part of a tester's role to be familiar with the common SDLC models so that appropriate test activities can take place. In any SDLC model, there are several characteristics of good testing:

- For every development activity, there is a corresponding test activity
- Each test level has test objectives specific to that level
- Test analysis and design for a given test level begin during the corresponding development activity
- Testers participate in discussions to define and refine requirements and design, and they are involved in reviewing work products as soon as drafts are available

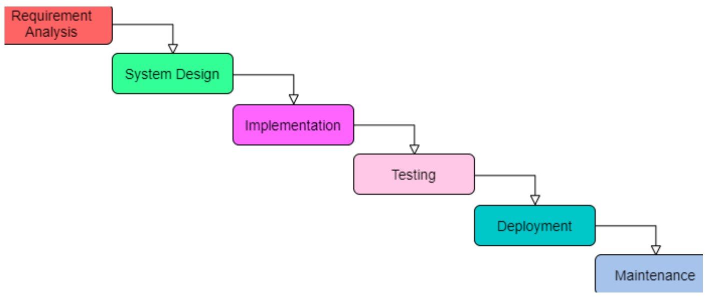
No matter which SDLC model is chosen, test activities should start in the early stages of the life cycle, adhering to the testing principle of early testing.

Verification & Validation
In every development life cycle, a part of testing is focused on verification testing, and another part is focused on validation testing.
Verification is concerned with evaluating a work product, component, or system to determine whether it meets the requirements set. Verification focuses on the question Is the deliverable built according to the specification?.
Validation is concerned with evaluating a work product, component, or system to determine whether it meets the user needs and requirements. Validation focuses on the question Is the deliverable fit for purpose, and does it provide a solution to the problem?



1- The Waterfall Model: is the earliest approach and a widely accepted SDLC model that was used for software development to ensure the success of some type of projects.
This model is very simple to understand and use. The model formed the transition between the phases of the software development process like a waterfall pouring. It illustrates the process of software development in a linear sequential flow. This means that each phase must be completed before the beginning of the next phase. Each phase finds what the previous phase produces at the starting point and there is no overlapping in the phases.

Design:
In The Waterfall approach, the entire software development process is divided into separate phases. Usually, In this model, the result of one phase acts sequentially as the input for the next phase.



- Requirement Gathering and analysis − During this process, all possible system requirements to be created are collected and recorded in a requirement specification document.
- System Design − In this phase, the requirements from the first phase will be reviewed, and the device design will be prepared. This system design helps determine the specifications of the hardware and system and helps define the overall system architecture.
- Implementation − The system is first developed with inputs from the system design into small programs called units, which are integrated into the next phase. Each unit is developed and tested for their functionality, known as Unit Testing.
- Integration and Testing − After testing each unit, all the units built during the implementation process are incorporated into a system. Any flaws and deficiencies are checked for the post-integration of the entire system.
- Deployment of system − Once testing is done, the product is deployed in the customer environment or released into the market.
- Maintenance − Some issues arise in the client environment. Patches are released to fix those issues. Some improved versions are released to enhance the product. Maintenance is done in the consumer environment to make such improvements.

Application:
Each software developed is different and requires an appropriate SDLC approach based on both internal and external factors to be followed. Many situations where the most effective use of the Waterfall model are;
- The requirements are documented very well.
- The definition of a product is stable.
- Technology is comprehensible.
- Requirements are not ambiguous.
- Ample resources are available with the required expertise to support the product.
- The project is short.

Advantages:
- It allows control and departmentalization.
- A schedule may be set with deadlines for each stage of development, and a product may proceed one by one through phases of the development process model.
- Every development phase proceeds in strict order.
- Easy and simple to use and understand.
- Easy to manage because of model rigidity. There are specific deliverables and a review process for each phase.
- Phases are processed one at a time and completed.
- Works well for smaller projects where requirements are very well understood.
- Phases clearly defined.
- The coding and testing steps are very short, as the requirements and design are clearly defined during the analysis and design phases.
- The number of errors during the test phase is very small.
- Tasks are easy to arrange.
- There is good documentation of the process and results.

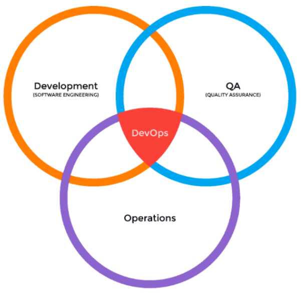
Disadvantages:
- It does not allow a great deal of reflection or revision.
- Once an application is in the phase of testing, it is very hard to go back and alter something that was not well-documented.
- No working software is produced until late during the life cycle.
- There is a high amount of risk and uncertainty.
- The time-loss due to the upper phase errors is quite high.
- It is not a good model for complex and object-oriented projects.
- It is a poor model for long and ongoing projects.
- It is not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty are high with this process model.
- The progress in phases can be difficult to measure.
- The product has to wait until the end of all the phases.
- It can not adapt to changing requirements.
- Scope adjustment over the life cycle can end a project.

2- DevOps Model (Development and Operations):

It focuses on collaboration between developers and other roles.
DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.
DevOps aims to shorten the system's development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.
DevOps is an evolution of the Agile Model of software development.
As the Agile model addressed the gap between clients and developers, DevOps addressed the gap between Developers and Operations.
The development team will submit the application to the operations team for implementation.
The operations team will monitor the application and provide relevant feedback to developers.

DevOps Phases:
Plan - Business owners and software development team discuss project goals and create a plan.
Code-Programmers then design and code the app and use tools like Git to store application code.
Build - Build tools like Maven and Gradle, take code from different repositories and combine them to build the complete application.
Test - Application is tested using automation testing tools like Selenium and Junit to ensure software quality.
Integrate - When testing is complete, new features are integrated automatically to the already existing codebase.
Deploy - Application is packaged after release and deployed from the development server to the production server.
Operate - Once the software is deployed, the operations team performs activities such as configuring servers and provisioning them with the required resources.
Monitor - Monitoring allows IT organizations to identify specific issues of specific releases and understand the impact on end-users.

- The DevOps is a new SDLC model that focuses on communication, collaboration, integration between Developers and Operations teams to accelerate innovation and the deployment of higher-quality and more reliable software products.
- DevOps is a culture that promotes collaboration between Development and Operations teams. This allows deploying code to production faster and in an automated way. It helps to increases an organization's speed to deliver applications and services.
- DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.



DevOps Processes:
DevOps Model consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, continuous monitoring, and continuous feedback.

Continuous Development: This is the phase that involves planning and coding of the software. The vision of the project is decided during the planning phase and the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are a number of tools for maintaining the code.
Continuous Testing: This is the stage where the developed software is continuously tested for bugs. For Continuous Testing, automation testing tools like Selenium, TestNG, JUnit, etc are used.
Continuous Integration: This stage is the heart of the entire DevOps life cycle. It is a software development practice in which the developers require to commit changes to the source code more frequently.
This may be on a daily or a weekly basis. Every commit is then built and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.
The code supporting new functionality is continuously integrated with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end-users.
Jenkins is a very popular tool used in this phase.
Continuous Deployment: This is the stage where the code is deployed to the production servers.
It is also important to ensure that the code is correctly deployed on all the servers.
Continuous Monitoring: This is a very crucial stage of the DevOps life cycle, where it is continuously monitored the performance of the application.
Here vital information about the use of the software is recorded.
This information is processed to recognize the proper functionality of the application.
The system errors such as low memory, server not reachable, etc. are resolved in this phase.
The root cause of any issue is determined in this phase.
It maintains the security and availability of the services.

DevOps Principles
Some of the DevOps principles are;

- Create production-like systems for development and testing environment
- Deployments need to be iterative and frequent. Ensure a reliable and repeatable process
- Continuously monitor and validate operational quality characteristics
- Amplify feedback loop

Advantages

Some of the DevOps advantages are;

- Time taken to create and deliver software is reduced
- The complexity of maintaining an application is reduced
- Improved collaboration between developers and operations team
- Continuous integration and delivery ensure faster time to market

Disadvantages

Some of the DevOps disadvantages are;

- Technology investments in the automation tools required for DevOps are costly and will take a great deal of time to identify and implement.
- It needs a specialist who can cover each stage of the software delivery pipeline rather than investing in a smaller number of full-stack developers.
- It is easy to add new features, but more features may not always better, even if they can be implemented efficiently.