**Branch :- Computer Science and Engineering Class :- III Year**

**Subject :- C-Skill Lab-IV Sem :- VI**

**Teacher Manual**

**PRACTICAL NO. 3**

**Aim:** Study software development life cycle and DevOps life cycle.

**Theory:**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software’s. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* SDLC is the acronym of Software Development Life Cycle.
* It is also called as Software Development Process.
* SDLC is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

**SDLC:**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure 1 is a graphical representation of the various stages of a typical SDLC.



**Stages of SDLC**

A typical Software Development Life Cycle consists of the following stages −

**Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

**Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

**Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

**Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high-level programming

languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

**Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**SDLC Models:**

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

* Waterfall Model
* Iterative Model
* Spiral Model
* V-Model
* Big Bang Model

1. **Waterfall Model**

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. Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.



**Waterfall Model**

1. **Iterative Model**

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model. Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."



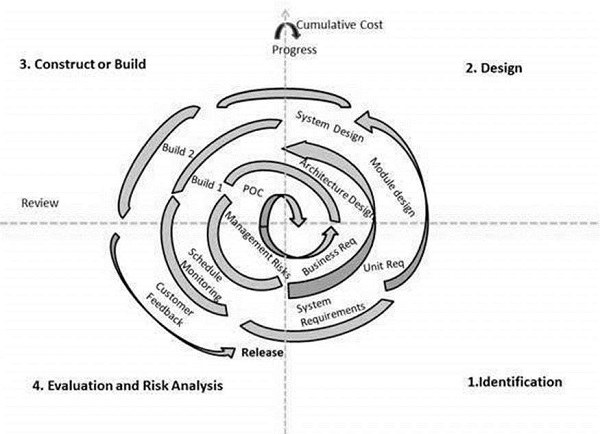
**Iterative and Incremental**

1. **Spiral Model**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

**Spiral Model - Design**

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.



**Spiral Model**

1. **V-Model**

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

**V-Model - Design**

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the ‘V’ and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The figure 5 illustration depicts the different phases in a V-Model of the SDLC.



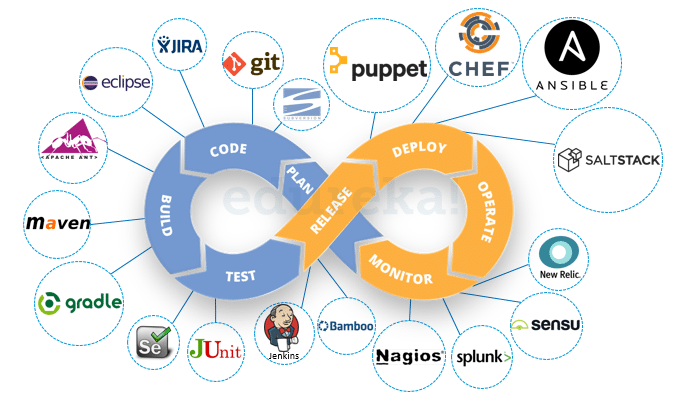
**V model**

1. **Big Bang Model**

The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis. Usually, this model is followed for small projects where the development teams are very small.

**DevOps**

* The term DevOps is a combination of two words namely Development and Operations. DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.
* The aim of DevOps is to shorten the system’s development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.
* DevOps is a software development approach through which superior quality software can be developed quickly and with more reliability.



**DevOps Life cycle**

* It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring.

#### Continuous Development

* This phase 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](https://www.edureka.co/blog/devops-tools" \t "_blank) that are required for planning, but there are a number of tools for maintaining the code.
* The code can be written in any language, but it is maintained by using Version Control tools.
* Maintaining the code is referred to as Source Code Management.
* The most popular tools used are Git, SVN, Mercurial, CVS, and JIRA.
* Also, tools like Ant, [Maven](https://www.edureka.co/blog/create-selenium-maven-project/" \t "_blank), Gradle can be used in this phase for building/ packaging the code into an executable file that can be forwarded to any of the next phases.

#### 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.
* These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality.
* In this phase, Docker Containers can be used for simulating the test environment.
* Selenium does the automation testing, and the reports are generated by [TestNG](https://www.edureka.co/blog/selenium-webdriver-tutorial" \t "_blank). This entire testing phase can be automated with the help of a Continuous Integration tool called Jenkins.
* Suppose you have written a selenium code in Java to test your application. Now you can build this code using ant or maven. Once the code is built, it is tested for User Acceptance Testing (UAT). This entire process can be automated using [Jenkins](https://www.edureka.co/blog/jenkins-tutorial/" \t "_blank).
* Automation testing saves a lot of time, effort and labor for executing the tests instead of doing this manually. Besides that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. We can also schedule the execution of the test cases at predefined times.
* After testing, the code is continuously integrated with the existing code.

1. **Continuous Integration**

* 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](https://www.edureka.co/blog/continuous-integration/" \t "_blank) 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.

1. **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. Configuration management and [Containerization tools](https://www.edureka.co/blog/docker-tutorial" \t "_blank) help in achieving Continuous Deployment (CD).
* [Configuration Management](https://www.edureka.co/blog/what-is-puppet/" \t "_blank) is the act of establishing and maintaining consistency in an application’s functional requirements and performance.
* In simpler words, it is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
* Since the new code is deployed on a continuous basis, configuration management tools play an important role in executing tasks quickly and frequently. Some popular tools that are used here are Puppet, [Chef](https://www.edureka.co/blog/what-is-chef/" \t "_blank), Salt Stack, and Ansible.
* Containerization tools also play an equally important role in the deployment stage. Docker and Vagrant are the popular tools used for this purpose.
* These tools help produce consistency across Development, Test, Staging and Production environments. Besides this, they also help in scaling-up and scaling-down of instances swiftly.
* Containerization tools help in maintaining consistency across the environments where the application is developed, tested and deployed.
* Using these tools, there is no scope of errors/ failure in the production environment as they package and replicate the same dependencies and packages used in the development/ testing/ staging environment. It makes application easy to run on different computers.

1. **Continuous Monitoring**

* In this stage performance of application is continuously monitored.
* In this stage 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.
* Also, if there are network issues, they are resolved in this phase. It helps us automatically fix the problem as soon as they are detected.
* This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system.
* The popular tools used for this are [Splunk](https://www.edureka.co/blog/what-is-splunk/" \t "_blank), [ELK Stack](https://www.edureka.co/blog/elk-stack-tutorial/" \t "_blank), [Nagios](https://www.edureka.co/blog/nagios-tutorial/" \t "_blank), NewRelic and Sensu.
* These tools help to monitor the application’s performance and the servers closely and also enable you to check the health of the system proactively.
* They can also improve productivity and increase the reliability of the systems, which in turn reduces IT support costs. Any major issues if found are reported to the development team so that it can be fixed in the continuous development phase. This leads to a faster resolution of the problems.

These DevOps stages are carried out on loop continuously till it achieves the desired product quality. Therefore, almost all of the major IT companies have shifted to DevOps for building their products.

**Result:** Thus, I have studied software development life cycle and DevOps life cycle.