Module: Software Development Lifecycle

Module Overview

This module introduces participants to the different phases of Software Development Lifecycle. In this module, participants will also explore the different software development models.

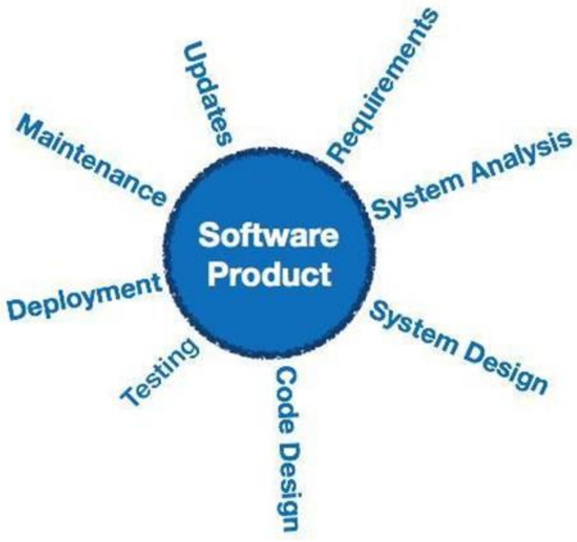
Module Objective

At the end of this module, students should be able to:

- Discuss Software development life cycle
- Demonstrate the different software development models

Overview of Software Engineering

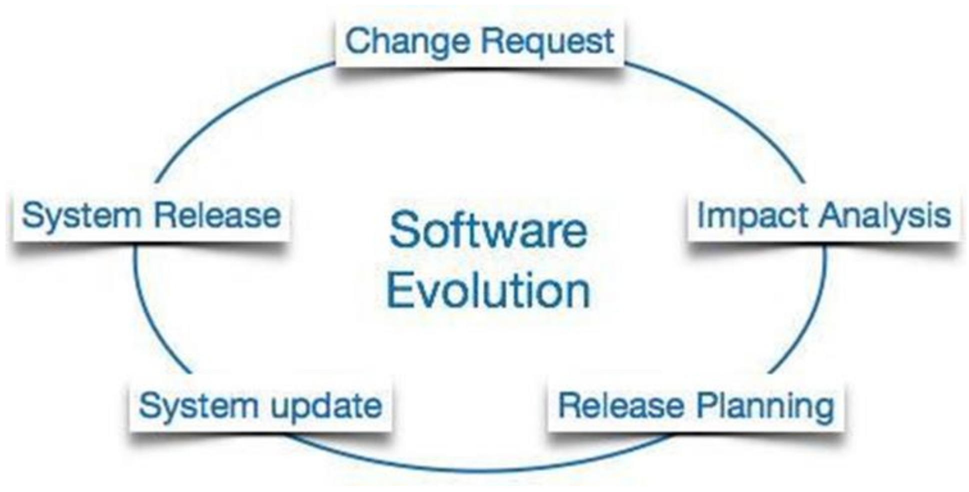
Let us understand what Software Engineering stands for. The term is made of two words, software and engineering. Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.
Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.



Software engineering is an engineering branch associated with development of software product using well- defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

Software evolution

The process of developing a software product using software engineering principles and methods is referred to as Software Evolution. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.



Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of the software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has the desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with the requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

Software Evolution Laws

Lehman has given laws for software evolution. He divided the software into three different categories:

- **Static-type (S-type) -** This is a software, which works strictly according to defined specifications andsolutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.

- **Practical-type (P-type) -** This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obviously instant. For example, gaming software.
- **Embedded-type (E-type)** - This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real-world situations. For example, online trading software.

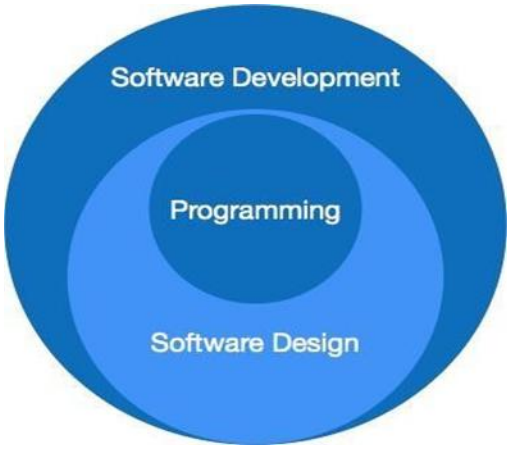
E-TYPE software evolution

Lehman has given eight laws for E-Type software evolution

- **Continuing change -** An E-type software system must continue to adapt to the real-world changes, else it becomes progressively less useful.
- **Increasing complexity -** As an E-type software system evolves, its complexity tends to increase unlesswork is done to maintain or reduce it.
- **Conservation of familiarity -** The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc., must be retained at any cost, to implement the changes in the system.
- **Continuing growth-** In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
- **Reducing quality -** An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
- **Feedback systems-** The E-type software systems constitute multi-loop, multi-level feedback systemsand must be treated as such to be successfully modified or improved.
- **Self-regulation -** E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
- **Organizational stability -** The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

Software paradigms

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are implemented. But, we need to see where in the software engineering concept, these paradigms stand. These can be combined into various categories, though each of them is contained in one another:



Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development Paradigm

This paradigm is known as software engineering paradigms; where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering whichhelps the software product to build. It consists of

- Requirement gathering
- Software design
- Programming

Software Design Paradigm

This paradigm is a part of Software Development and includes

- Design
- Maintenance
- Programming

Programming Paradigm

This paradigm is related closely to programming aspect of software development. This includes –

- Coding

- Testing
- Integration

Need of Software Engineering

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working. Following are some of the needs stated:

Large software - It is easier to build a wall than a house or building,likewise, as the size of the software becomes large, engineering has to step to give it a scientific process.

Scalability- If the software process were not based on scientific andengineering concepts, it would be easier to re-create new software than to scale an existing one.

Cost- As hardware industry has shown its skills and huge manufacturinghas lower down the price ofcomputer and electronic hardware. But, cost of the software remains high if proper process is not adapted.

Dynamic Nature- Always growing and adapting nature of the softwarehugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where the software engineering plays a good role.

Quality Management- Better process of software development providesbetter and quality softwareproduct.

Characteristics of good software

A software product can be judged by what it offers and how well it can be used. This software must satisfyon the following grounds:

- Operational
- Transitional
- Maintenance

Well-engineered and crafted software is expected to have the following characteristics:

Operational

This tells us how well the software works in operations. It can be measured on

- Budget

- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

Transitional

This aspect is important when the software is moved from one platform to another:

- Portability
- Interoperability
- Reusability
- Adaptability

Maintenance

This aspect briefs about how well the software has the capabilities to maintain itself in the ever-changing environment:

- Modularity
- Maintainability
- Flexibility
- Scalability

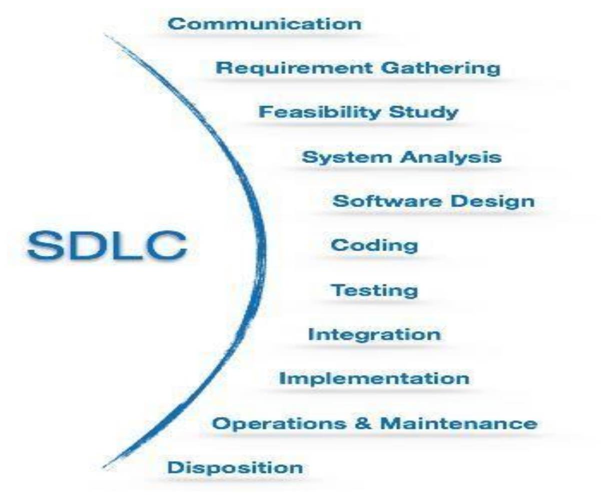
In short, Software engineering is a branch of computer science, which uses well-defined engineering conceptsrequired to produce efficient, durable, scalable, in-budget, and on-time software products.

Software Development Life Cycle

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLCframework includes the following steps:



Communication

This is the first step where the user initiates the request for a desired software product. The user contacts the service provider and tries to negotiate the terms, submits the request to the service providing organization in writing.

Requirement Gathering

This step onwards the software development team works to carry on the project. The team holds discussionswith various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given –

- studying the existing or obsolete system and software,
- conducting interviews of users and developers,
- referring to the database or
- collecting answers from the questionnaires.

Feasibility Study

After requirement gathering, the team comes up with a rough plan of software process. At this step the teamanalyzes if a software can be designed to fulfill all requirements of the user, and if there is any possibility of software being no more useful. It is also analyzed if the project is financially, practically, and technologicallyfeasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyzes the scope of the projectand plans the schedule and resources accordingly.

Software Design

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design, and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams, and in some cases pseudo codes.

Coding

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers andthorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing, and testing the product at user’s end. Early discovery of errors andtheir remedy is the key to reliable software.

Integration

Software may need to be integrated with the libraries, databases, and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

Implementation

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

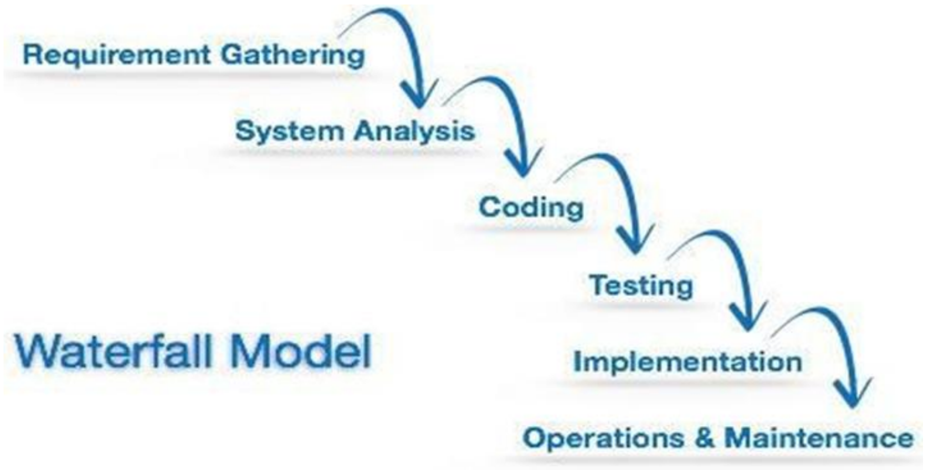
Software Model

The software development paradigm helps a developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods, and procedures, which are expressed clearly and defines software development life cycle. A few of software development paradigms or process models are defined as follows.

Software Development Life Cycle Models

Waterfall Model

Waterfall model is the simplest model of software development paradigm. All the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.



This model assumes that everything is carried out and taken place perfectly as planned in the previous stageand there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us to go back and undo or redo our actions.

Advantage:

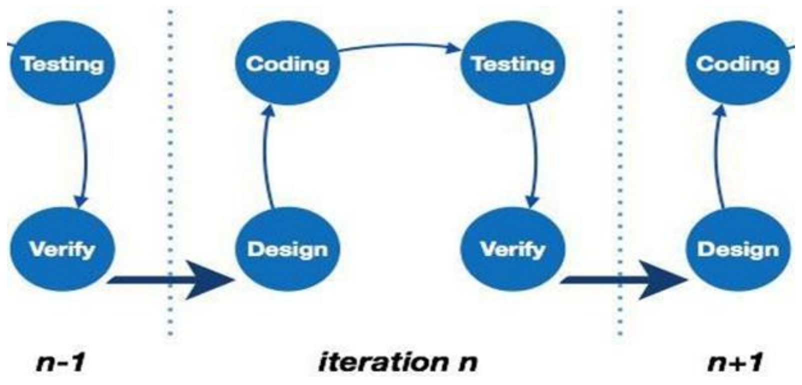
This model is best suited when developers already have designed and developed similar software in the pastand are aware of all its domains.

Drawback:

The sequential nature of model does not allow to go back and undo or redo the actions

Iterative Model

This model leads the software development process in iterations. It projects the process of development incyclic manner repeating every step after every cycle of SDLC process.



The software is first developed on very small scale and all the steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested, and added to the software. Every cycle produces a software, which is complete in itself and has more features and capabilities than that of the previous one.

After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

Advantage:

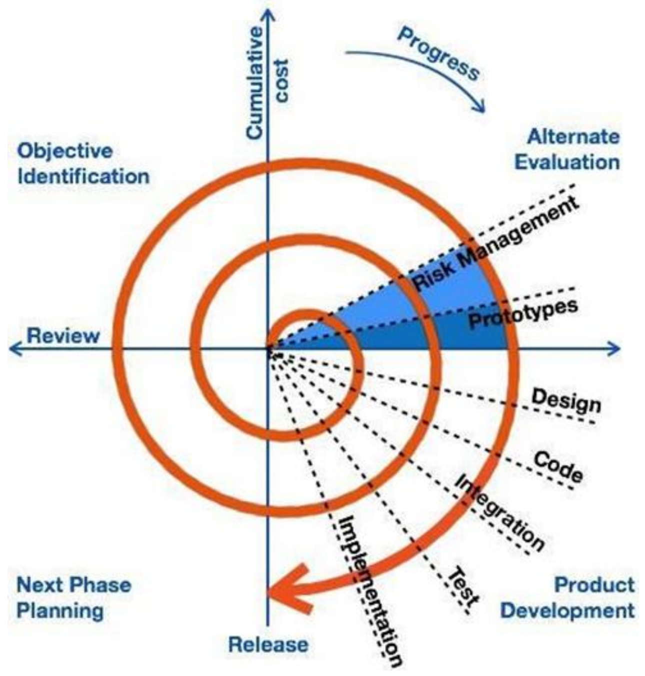
Because a cycle includes small portion of whole software process, it is easier to manage the developmentprocess.

Drawback:

Since more features are added to the software on every iteration, it consumes more resources.

Spiral Model

Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if youchoose one SDLC model and combined it with cyclic process (iterative model).



This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

Advantage:

1. Additional functionality or changes can be done at the later stage

2. Cost Estimation becomes easy Drawback:

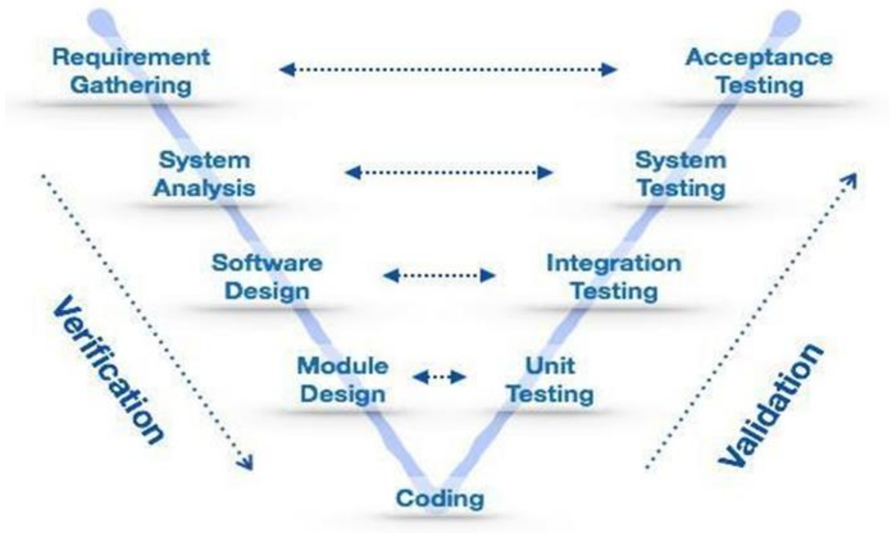
1. Not advisable for smaller projects, as it might cost more

2. Demands Risk assessment expertise

V – model

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.

The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.



At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.

This makes both verification and validation go in parallel. This model is also known as verification and validation model.

Advantage:

1. Each phase has specific deliverables.
2. Works well for small projects where requirements are easily understood.
3. Utility of the resources is high.

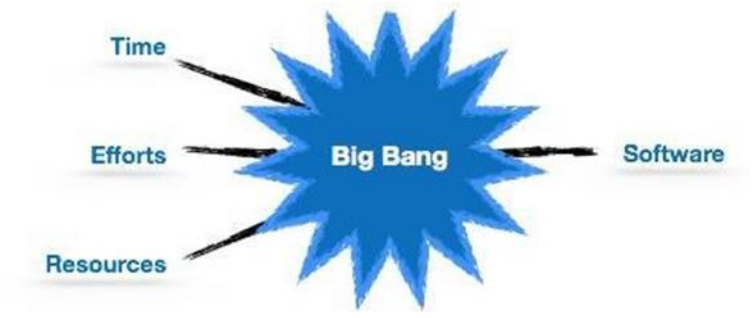
Drawback:

1. Very rigid, like the waterfall model.

2 Little flexibility and adjusting scope is difficult and expensive.

Big Bang Model

This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets, and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.



Advantage:

1. Simple and easy to implement
2. Requires very little or no planning
3. Easy to manage since, no formal procedure required before starting any project

Drawback:

1. Risky model
2. Changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project.