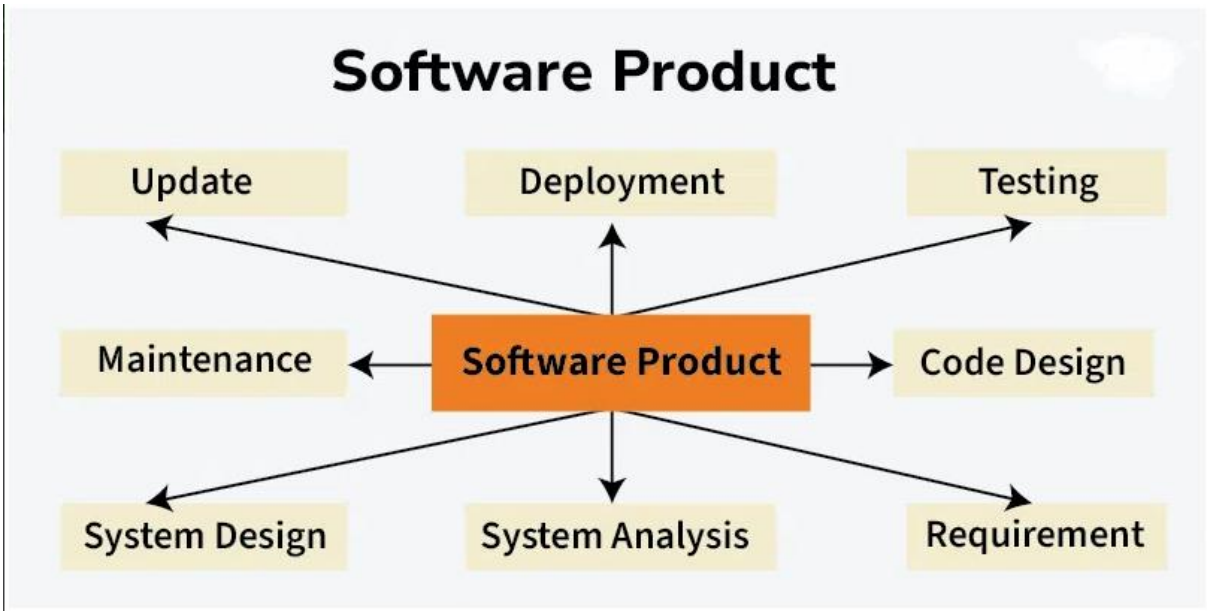
Unit -1 Introduction to Software Engineering

➢ Definition and Scope: Software engineering

What is Software Engineering?
Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.



Software Engineering is a systematic, disciplined, quantifiable study and approach to the design, development, operation, and maintenance of a software system.

Objectives of Software Engineering

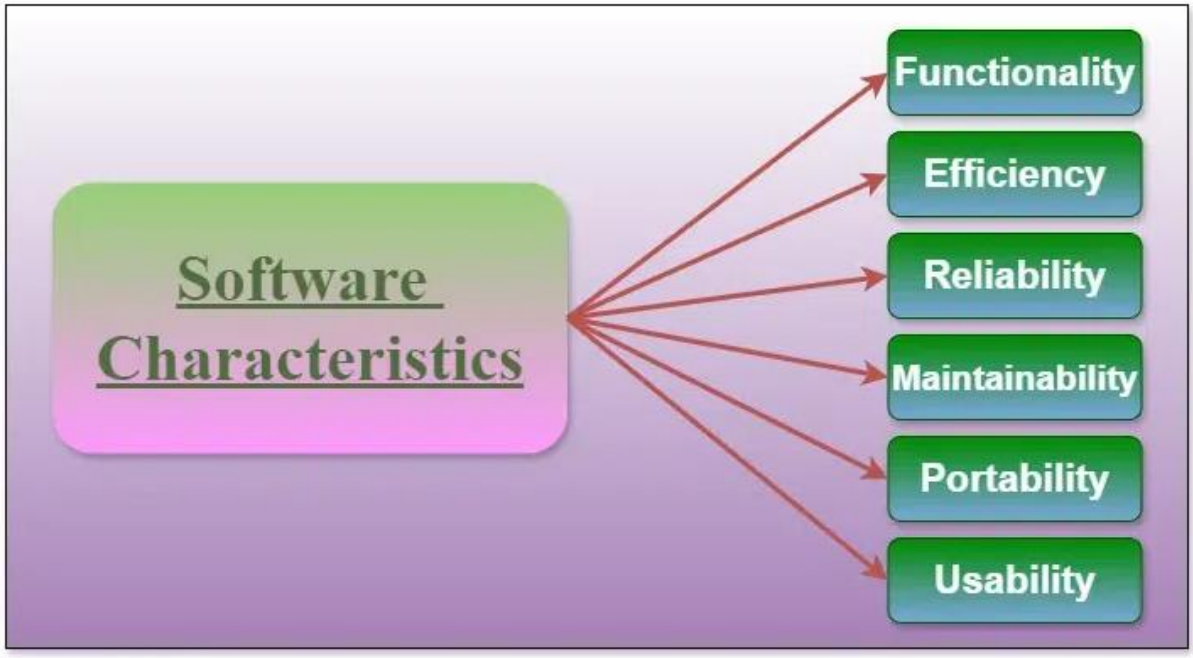
1. **Maintainability:** It should be feasible for the software to evolve to meet changing requirements.
2. **Efficiency:** The software should not make wasteful use of computing devices such as memory, processor cycles, etc.
3. **Correctness:** A software product is correct if the different requirements specified in the SRS Document have been correctly implemented.
4. **Reusability:** A software product has good reusability if the different modules of the product can easily be reused to develop new products.
5. **Testability:** Here software facilitates both the establishment of test criteria and the evaluation of the software concerning those criteria.
6. **Reliability:** It is an attribute of software quality. The extent to which a program can be expected to perform its desired function, over an arbitrary time period.
7. **Portability:** In this case, the software can be transferred from one computer system or environment to another.
8. **Adaptability:** In this case, the software allows differing system constraints and the user needs to be satisfied by making changes to the software.

Unit -1 Introduction to Software Engineering

9. **Interoperability:** Capability of 2 or more functional units to process data cooperatively.

➢ characteristic of software engineering

Software is defined as a collection of computer programs, procedures, rules, and data. Software Characteristics are classified into six major components.
There are 6 components of Software Characteristics are discussed here:



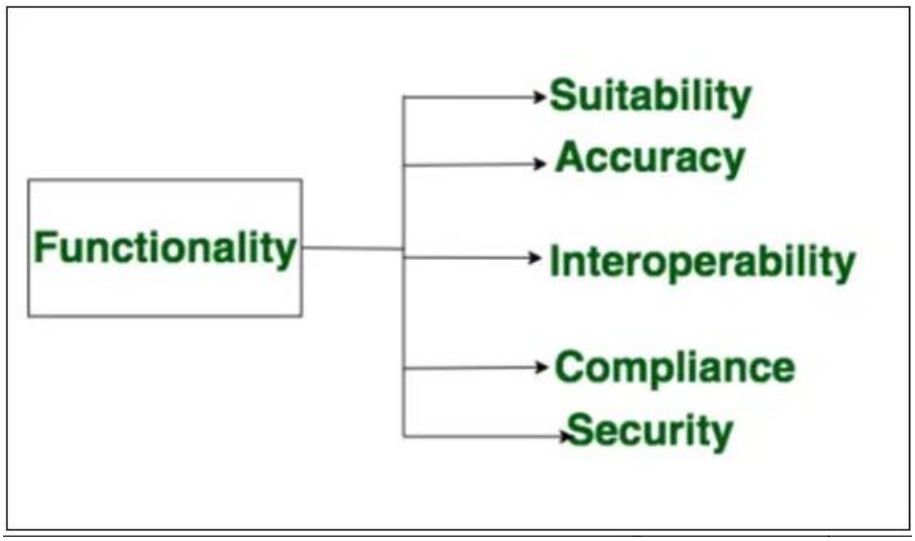
1. Functionality

Functionality refers to the set of features and capabilities that a software program or system provides to its users. It is one of the most important characteristics of software, as it determines the usefulness of the software for the intended purpose.
Examples of functionality in software include:

a. Data storage and retrieval
b. Automation and scripting
c. Communication and networking

It is important to balance the need for functionality with the need for ease of use, maintainability, and scalability.
It is important to balance the need for functionality with the need for ease of use, maintainability, and scalability.

Unit -1 Introduction to Software Engineering



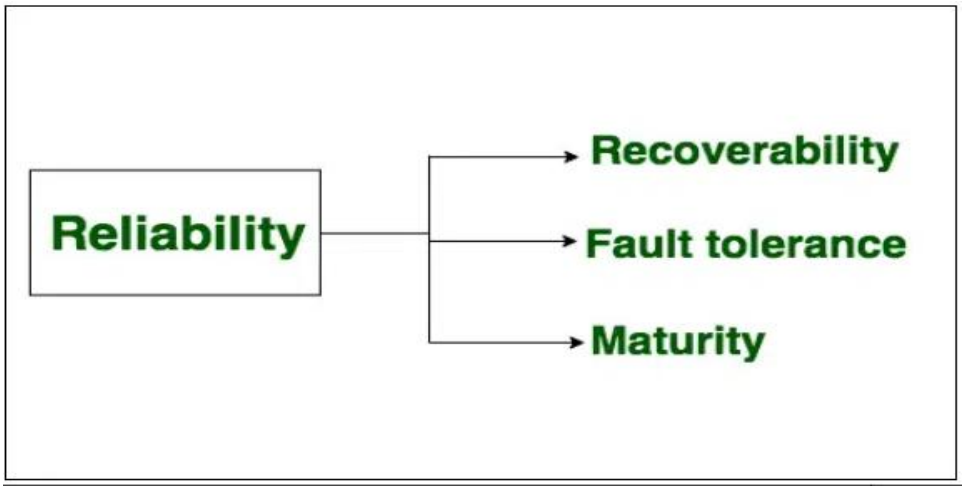
2. Reliability

A set of attributes that bears on the capability of software to maintain its level of performance under the given condition for a stated period of time.
Reliability is a characteristic of software that refers to its ability to perform its intended functions correctly and consistently over time. Reliability is an important aspect of software quality, as it helps ensure that the software will work correctly and not fail unexpectedly.

Examples of factors that can affect the reliability of software include:

a. Bugs and errors in the code
b. Lack of testing and validation
c. Poorly designed algorithms and data structures
d. Incompatibilities with other software or hardware

Required functions are:



Unit -1 Introduction to Software Engineering

3. Efficiency

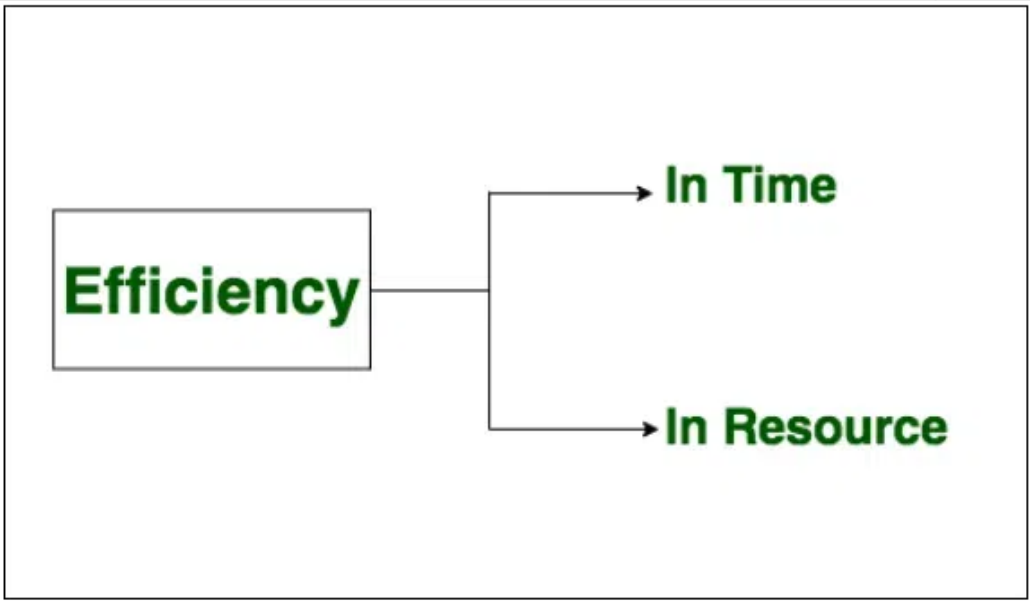
It refers to the ability of the software to use system resources in the most effective and efficient manner. The software should make effective use of storage space and executive command as per desired timing requirements. High efficiency means that a software program can perform its intended functions quickly and with minimal use of resources, while low efficiency means that a software program may be slow or consume excessive resources.

Examples of factors that can affect the efficiency of the software include:

a. UnOptimized Code
b. High Network latency or bandwidth usage
c. poorly designed algorithms and data structures.

Efficiency is important in software systems that are resource-constrained, high-performance, and real-time systems.
It is also important in systems that need to handle many users or transactions simultaneously.

Required functions are:

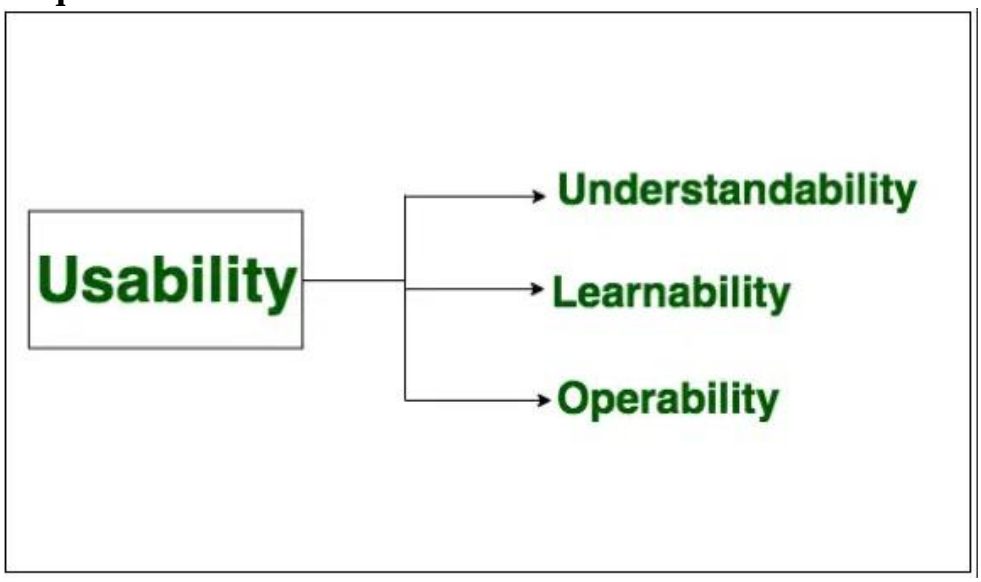


Unit -1 Introduction to Software Engineering

4. Usability

It refers to the extent to which the software can be used with ease.
The amount of effort or time required to learn how to use the software.

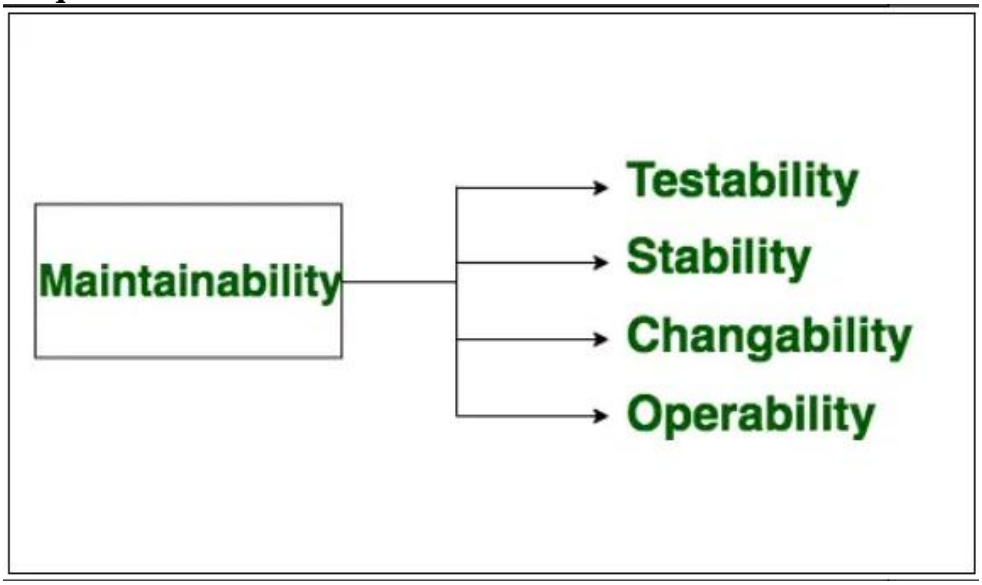
Required functions are:



5. Maintainability

It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

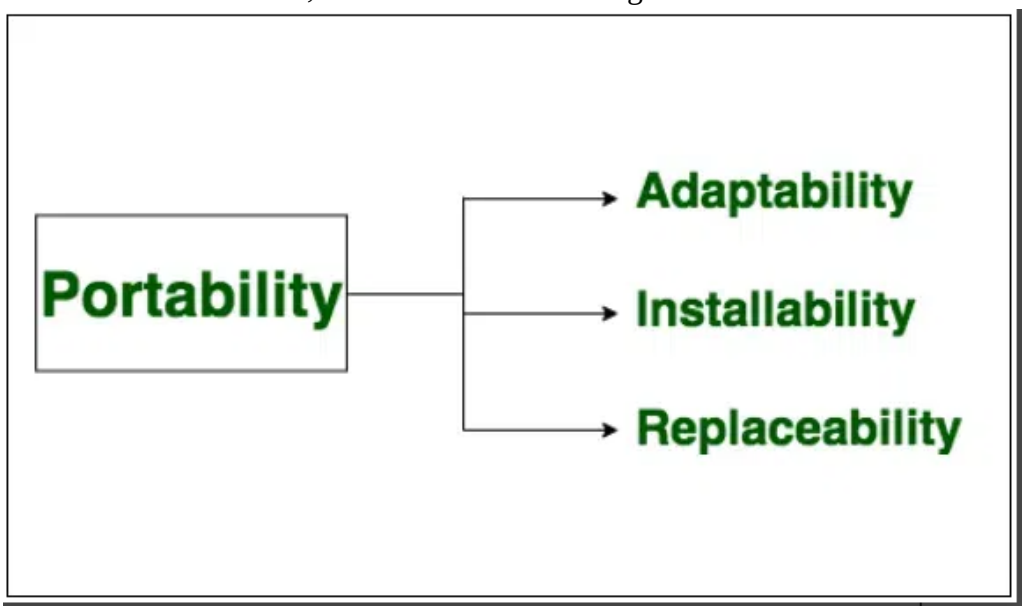
Required functions are:



Unit -1 Introduction to Software Engineering

6. Portability:

A set of attributes that bears on the ability of software to be transferred from one environment to another, without minimum changes.



Software Development Life Cycle (SDLC)

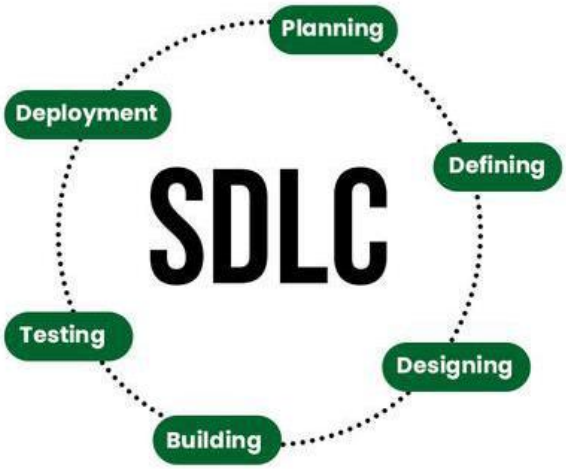
Software Development Life Cycle (SDLC) is a structured process that is used to design, develop, and test high-quality software.

SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users requirements.

➢ What is the Software Development Life Cycle (SDLC)?

SDLC is a process followed for software building within a software organization. SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software.

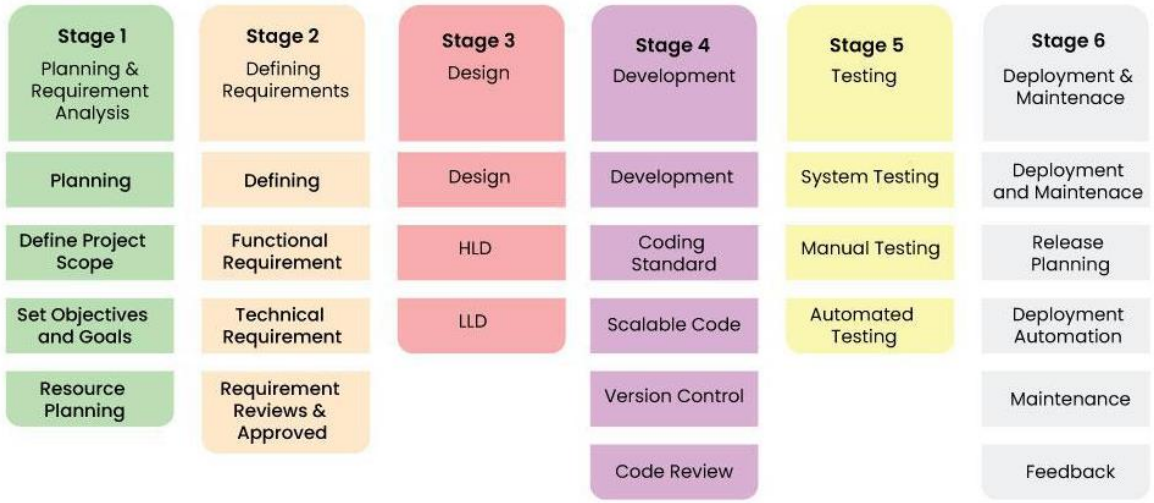
The life cycle defines a method for improving the quality of software and the all-around development process.



Unit -1 Introduction to Software Engineering

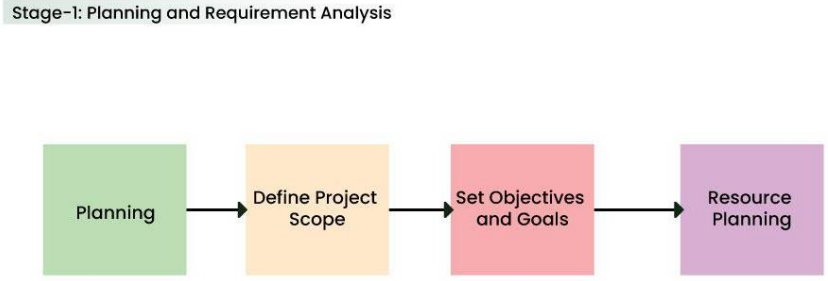
Stages of the Software Development Life Cycle:

SDLC specifies the tasks to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer's expectations and fits within the overall budget.



Stage 1: Planning and Requirement Analysis:

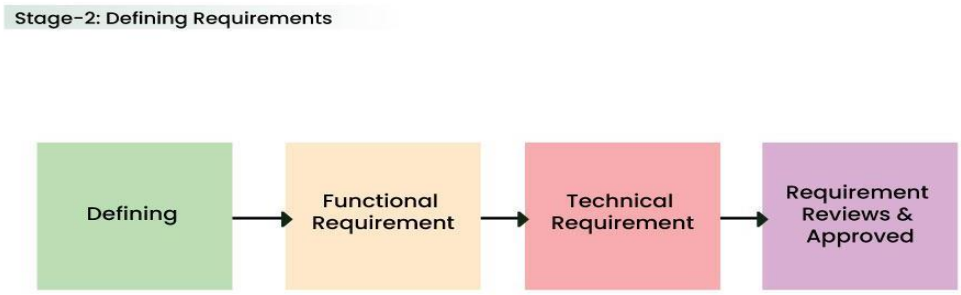
Planning is a crucial step in everything, just as in software development. In this same stage, requirement analysis is also performed by the developers of the organization. This is attained from customer inputs, and sales department/market surveys.



Unit -1 Introduction to Software Engineering

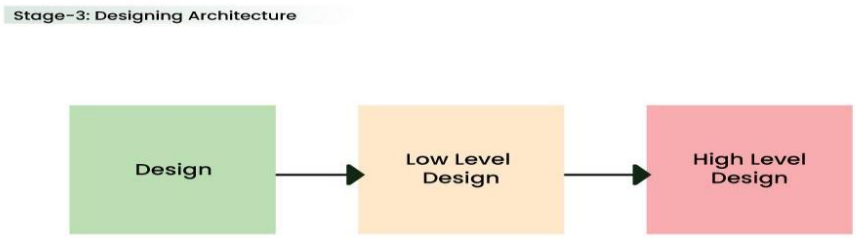
Stage 2: Defining Requirements

In this stage, all the requirements for the target software are specified. These requirements get approval from customers, market analysts, and stakeholders.
This is fulfilled by utilizing SRS (Software Requirement Specification).



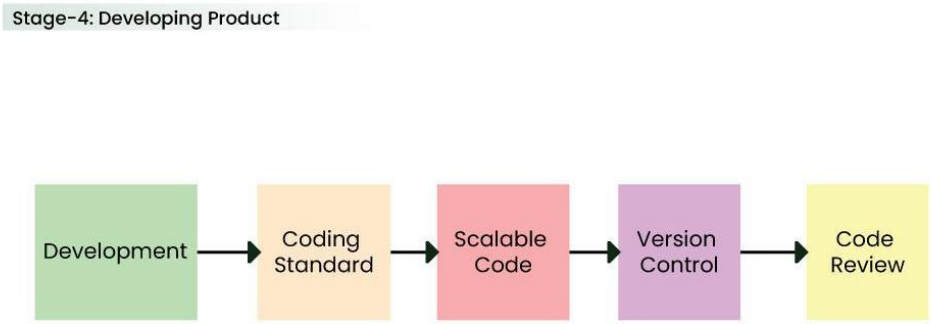
Stage 3: Designing Architecture

SRS(Software Requirements Specification) is a reference for software designers to come up with the best architecture for the software. Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the Design Document Specification (DDS).



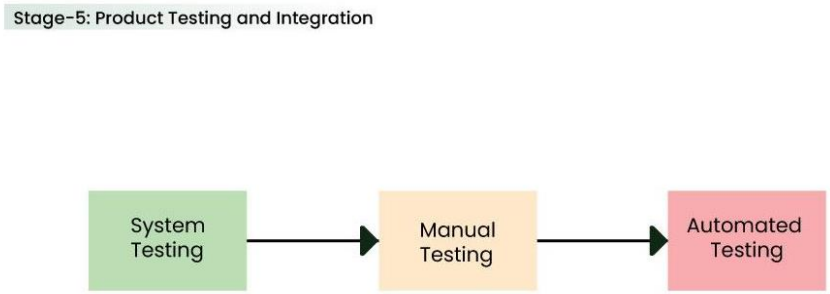
Stage 4: Developing Product

At this stage, the fundamental development of the product starts. For this, developers use a specific programming code as per the design in the DDS. Hence, it is important for the coders to follow the protocols set by the association. Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

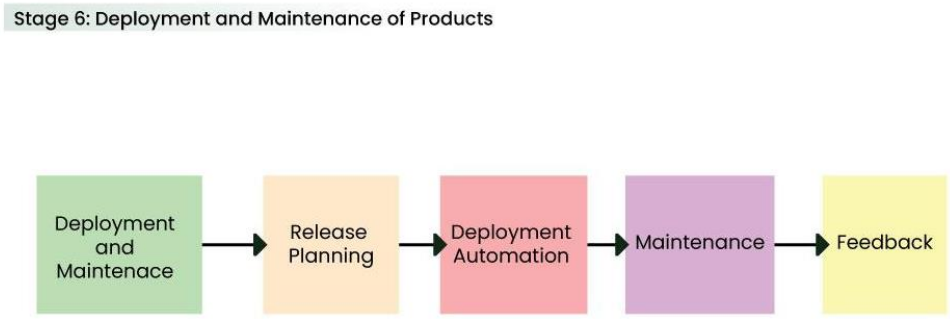


Unit -1 Introduction to Software Engineering

Stage 5: Product Testing and Integration
After the development of the product, testing of the software is necessary to ensure its smooth execution. Although, minimal testing is conducted at every stage of SDLC. Therefore, at this stage, all the probable flaws are tracked, fixed, and retested. This ensures that the product confronts the quality requirements of SRS.



Stage 6: Deployment and Maintenance of Products
After detailed testing, the conclusive product is released in phases as per the organization's strategy. Then it is tested in a real industrial environment. It is important to ensure its smooth performance. If it performs well, the organization sends out the product as a whole.



They define the **sequence of development stages**, such as requirements, design, coding, testing, and deployment.

Common Models:
- Waterfall Model
- Agile Model
- V-Model
- Spiral Model
- Incremental Model
- RAD Model

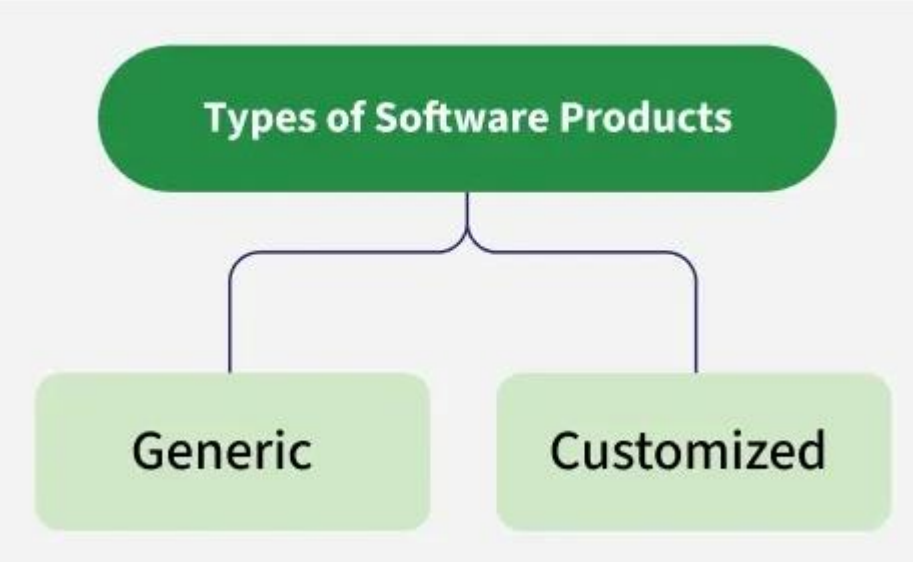
Unit -1 Introduction to Software Engineering

Software Product

What Is a Software Product?
Software Products is any Software or any system which has been developed, tested and maintained for the specific purpose to solve that problem.

Software products are produced with the help of the software process. The software process is a way in which we produce software.

Types of Software Products:
Software products fall into two broad categories:



1. Generic products: Generic products are stand-alone systems that are developed by a production unit and sold on the open market to any customer who can buy them.
Products like **Microsoft Office** or **Google Chrome**. They are ready to use right out of the box and are designed to meet the general needs of many users. Whether you're working on a document or browsing the web, these tools are flexible enough to suit a wide range of tasks.

2. Customized Products: Customized products are the systems that are commissioned by a particular customer. Some contractor develops the software for that customer.
These are designed specifically for a particular business or set of users. Imagine a software system built just for managing employee data in a company or a custom CRM system that's perfectly aligned with the way a business operates.

Unit -1 Introduction to Software Engineering

Software Engineering Principles

Software Engineering is the systems engineering approach for software product/application development. It is an engineering branch associated with analyzing user requirements, design, development, testing, and maintenance of software products.

Principles of good software engineering are –

1. One of the basic software Engineering principles is Better Requirement analysis which gives a clear vision of the project. At last, a good understanding of user requirements provides value to its users by delivering a good software product that meets users' requirements.

2. All designs and implementations should be as simple as possible mean the KISS (Keep it Simple, Stupid) principle should be followed.

3. Maintaining the vision of the project is the most important thing throughout complete development process for the success of a software project.

4. Software projects include a number of functionalities, all functionalities should be developed in a modular approach so that development will be faster and easier.

5. Another specialization of the principle of separation of concerns is Abstraction for suppressing complex things and delivering simplicity to the customer/user means it gives what the actual user needs and hides unnecessary things.

6. Think then Act is a must-required principle for software engineering means before starting developing functionality first it requires to think about application architecture, as good planning on the flow of project development produces better results.

7. When other developers work with another's code they should not be surprised and should not waste their time in getting code. So providing better Documentation at required steps is a good way of developing software projects.

8. Law of Demeter should be followed as it makes classes independent on their functionalities and reduces connections and inter dependability between classes which is called coupling.

9. The developers should develop the project in such a way that it should satisfy the principle of Generality means it should not be limited or restricted to some of cases/functions rather it should be free from unnatural restrictions and should be able to provide service to customers what actually they need or general needs in an extensive manner.

10. Principle of Consistency is important in coding style and designing GUI (Graphical User Interface) as consistent coding style gives an easier reading of code and consistency in GUI makes user learning easier in dealing with interface and in using the software.

Unit -1 Introduction to Software Engineering

11. Never waste time if anything is required and that already exists at that time take the help of Open source and fix it in your own way as per requirement.

12. To exit in current technology market trends Using modern programming practices is important to meet users' requirements in the latest and advanced way.