Unit-1
Introduction to Software Engineering

What is Engineering?

- It is all about developing an application or a product in a well defined format structures using scientific principles of logics.
- Engineering is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.

What is a Software?

- The software is a collection of integrated programs.
- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.
- Software is a generic term used to refer to applications, scripts and programs that run on a device.
- An example of software is Excel or Windows or iTunes.

What is 'Software Engineering'?

Definition: Software engineering is a detailed study of engineering to the design, development and maintenance of software. Software engineering was introduced to address the issues of low-quality software projects. Problems arise when a software generally exceeds timelines, budgets, and reduced levels of quality. It ensures that the application is built consistently, correctly, on time and on budget and within requirements.

A software product is judged by how easily it can be used by the end-user and the features it offers to the user. An application must score in the following areas:-

1) Operational: -This tells how good a software works on operations like budget , usability, efficiency, correctness ,functionality , dependability , security and safety.

2) Transitional: - Transitional is important when an application is shifted from one platform to another. So, portability, reusability and adaptability come in this area.

3) Maintenance: - This specifies how good a software works in the changing environment. Modularity, maintainability, flexibility and scalability come in maintenance part.

Why is Software Engineering required?

Software Engineering is required due to the following reasons:

- To manage Large software.
- For more Scalability.
- Cost Management.
- To manage the dynamic nature of software.
- For better quality Management.

Nature of Software Project :

- User Friendly
- Accuracy
- Security
- Efficiency
- Quality
- Cost effectiveness
- Adaptability

Reasons for Project Success/Unsuccess:

Success:

- Long time survival
- Fulfilling the requirements
- Cost effectiveness
- Quantity
- Risk Management

Unsuccess:

- Lack of proper resources
- Less Man Power
- Unefficiency

Difference between Software & Hardware :

Hardware	Software

Hardware is a Physical part of a computer that causes processing of data.	Software is a set of instructions that cause processing of data
It is manufactured.	It is developed.
It is difficult to modify once it is manufactured.	It is easy to modify the process even after the development.
Cost is concentrated on production.	Cost is concentrated on design.
It is component built.	It is custom built.
Hardware can be touched.	Software cannot be touched.
Hardware is repaired in case of problem.	Software is debugged in case of problem.

Software Crisis

Software Crisis is a term used in computer science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to using the same workforce, same methods, same tools even though rapidly increasing in software demand, the complexity of software, and software challenges. With the increase in the complexity of software, many software problems arise because existing methods were insufficient. If we will use the same workforce, same methods, and same tools after the fast increase in software demand, software complexity, and software challenges, then there arise some problems like software budget problems, software efficiency problems, software quality problems, software managing and delivering problem, etc. This condition is called a **software crisis.**

Causes of Software Crisis:

- The cost of owning and maintaining software was as expensive as developing the software
- At that time Projects were running over-time
- At that time Software was very inefficient
- The quality of the software was low quality
- Software often did not meet user requirements
- The average software project overshoots its schedule by half
- At that time Software was never delivered
- Non-optimal resource utilization.
- Difficult to alter, debug, and enhance.
- The software complexity is harder to change.

Solution of Software Crisis:

There is no single solution to the crisis. One possible solution to a software crisis is *Software Engineering* because software engineering is a systematic, disciplined, and quantifiable approach. For preventing software crises, there are some guidelines:

- Reduction in software over budget.
- The quality of software must be high.
- Less time is needed for a software project.
- Experienced and skilled people working over the software project.
- Software must be delivered.
- Software must meet user requirements.

Characteristic of software:
There is some characteristic of software which is given below:

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

Changing Nature of Software:
Nowadays, seven broad categories of computer software present continuing challenges for software engineers .which is given below:

1. **System Software:**
   System software is a collection of programs which are written to service other programs. Some system software processes complex but determinate, information structures. Other system application process largely indeterminate data. Sometimes when, the system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.
2. **Application Software:**
   Application software is defined as programs that solve a specific business need. Application in this area process business or technical data in a way that facilitates business operation or management technical decision making. In addition to convention data processing application, application software is used to control business function in real time.
3. **Engineering and Scientific Software:**
   This software is used to facilitate the engineering function and task. however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take a real-time and even system software characteristic.
4. **Embedded Software:**
   Embedded software resides within the system or product and is used to implement and control feature and function for the end-user and for the system itself. Embedded software can perform the limited and esoteric function or provided significant function and control capability.
5. **Product-line Software:**
   Designed to provide a specific capability for use by many different customers, product line software can focus on the limited and esoteric marketplace or address the mass consumer market.
6. **WebApplication:**
   It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as e-commerce and B2B application grow in importance. Web apps are evolving into a sophisticate computing environment that not only provides a standalone feature, computing function, and content to the end user.
7. **Artificial Intelligence Software:**
   Artificial intelligence software makes use of a nonnumerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.

Software Myths :

What is meant by Myth?

- The meaning of myth is fable (wrong) stories or wrong beliefs.
- In a relation to a computer software myth is nothing but information, misunderstanding or confusion in software development field.
- Many **Software** problems arises due to myths that are formed during the initial stage of software development.

Definition of Software Myths :

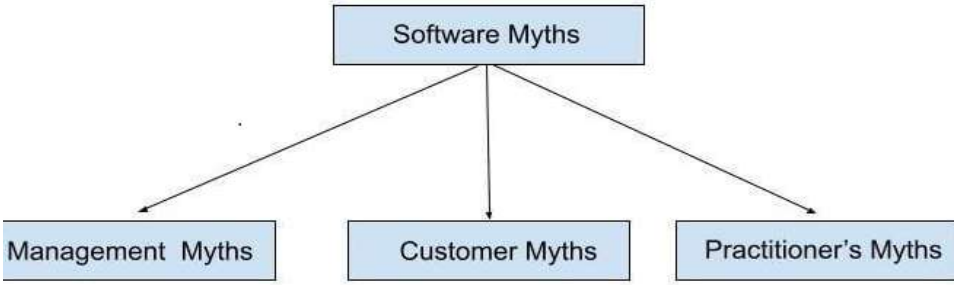
The development of software requires dedication and understanding on the developers' part. Many software problems arise due to myths that are formed during the initial stages of software development. Unlike ancient folklore that often provides valuable lessons, software myths propagate false beliefs and confusion in the minds of management, users and developers.Managers, who own software development responsibility, are often under strain and pressure to maintain a software budget, time constraints, improved quality, and many other considerations.

There are three types of Software Myths :

1.Management Myths

2.Customer Myths

3.Practitioner's Myths



Management Myths: Managers with software responsibility, like mangers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.

Myth1: Manager think "when need, we can add more programmer for faster development".

Reality:

- Training must be given to new comers.
- Spend time for training people.
- Cannot be developed fast.

Myth2: We already have a book that's full of standards and procedures for building software. It will provide the developer everything that he needs in the development process.

Reality:

- The book Exits but question arise.
- Are software developer aware about this book?
- Does it contain all modern practices?
- Does it focus quality?
- Is it actually used by developer?

Myth3: Manager think "there is no need to change approach to software development. We can develop same kind of software that we develop 10 year ago.

Reality :

- Quality of software need to improve according to customer demands.
- Customer demands change time to time.

Customer Myth: The customer can be the direct users of the software, the technical team, marketing/sales department, or other company. Customer has myths leading to false expectations (customer) & that's why you create dissatisfaction with the developer.

Myth1: Only the general statement is sufficient and no need to mention detail project requirements.

Reality:

- Other details are also essential.
- Customer must provides design details, validation criteria.
- During software development customer and developer communication is essential.

Myth2: Software requirements continually change, but change can be easily accommodated because software is flexible.

Reality:

- When requirements changes are requested early (before design or code has been started), the cost impact is relatively small.
- Otherwise cost is so high.

3.Practitioner's Myths / Developers Myths : Software practitioners are the ones who are involved in the development and maintenance of the software. Earlier developing software is considered as an art. So, the software practitioners have developed some myths regarding the software.

Myth 1: Practitioner think "Once we write the program and get it to work, our job is done".

Reality: Almost 60% to 80% Percent work is required after delivering the project to customer for first time.

Myth 2: Until I get the program "running" I have no way of assessing its quality.

Reality:

- Effective SQA can be applied during project development.
- FTR are conducted to assure quality of software.
- FTR is meeting conducted by technical staff at any stage of software development.

Software Quality Assurance (SQA) is simply **a way to assure quality in the software**. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly. Software Quality Assurance is a process which works parallel to development of software.

Formal Technical Review (FTR) is a software quality control activity performed by software engineers.

Objectives of formal technical review (FTR): Some of these are:

- Useful to uncover error in logic, function and implementation for any representation of the software.
- The purpose of FTR is to verify that the software meets specified requirements.
- To ensure that software is represented according to predefined standards.
- It helps to review the uniformity in software that is development in a uniform manner.
- To makes the project more manageable.

Myth 3: The only deliverable work product for a successful project is the working program.

Reality :

- A working program is only one part of a software configuration that includes many elements.
- A variety of work products (e.g, models, documents, plans) provide a foundation for successful engineering and
- More important, guidance for software support.

Software Development Life Cycle (SDLC) :

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. 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.

What is 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 is a graphical representation of the various stages of a typical 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.

What is a software process model?

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

A model will define the following:

- The tasks to be performed
- The input and output of each task
- The pre and post conditions for each task
- The flow and sequence of each task

Components of Software :

There are three components of the software: These are : Program, Documentation, and Operating Procedures.

- **Program –**
  A computer program is a list of instructions that tell a computer what to do.

- **Documentation –**
  Source information about the product contained in design documents, detailed code comments, etc.

- **Operating Procedures –**
  Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

There are four basic key process activities:

- **Software Specifications –**
  In this process, detailed description of a software system to be developed with its functional and non-functional requirements.

- **Software Development –**
  In this process, designing, programming, documenting, testing, and bug fixing is done.

- **Software Validation –**
  In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

- **Software Evolution –**
  It is a process of developing software initially, then timely updating it for various reasons.

There are many kinds of process models for meeting different requirements. We refer to these as **SDLC models** (Software Development Life Cycle models). The most popular and important SDLC models are as follows:

1. Waterfall model
2. Prototype model
3. Incremental model
4. Spiral model
5. Agile 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.

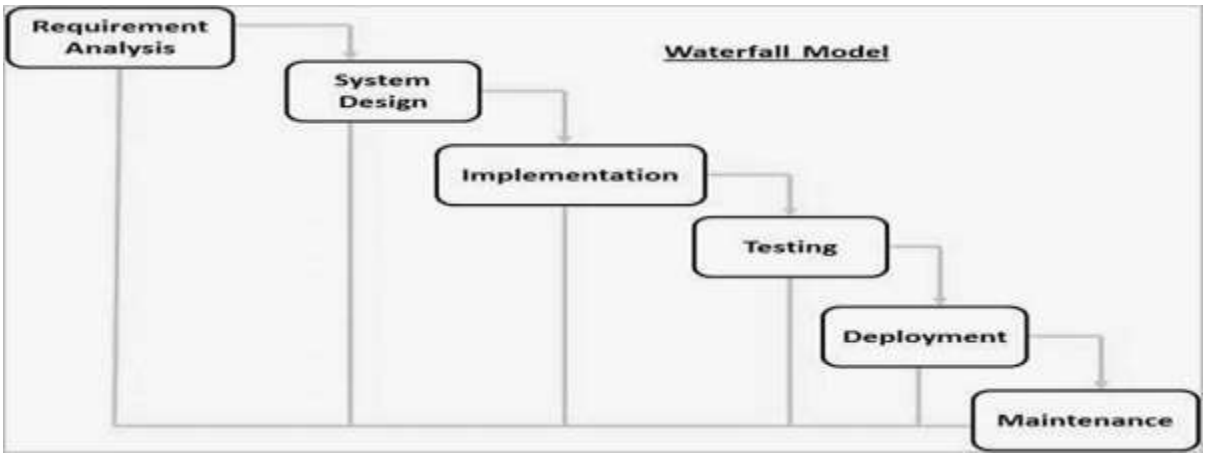
The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

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.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

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

Waterfall Model – Application :

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall model is schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.