Unit-1

Introduction to Software Engineering: Software evolution, Legacy software, Software myths.

A Generic View of Process: Software engineering layers, Process frame work, Capability Maturity Model Integration (CMMI).

➢ Software Engineering

Let us first understand what software engineering stands for. The term is made of two words, software, and engineering.

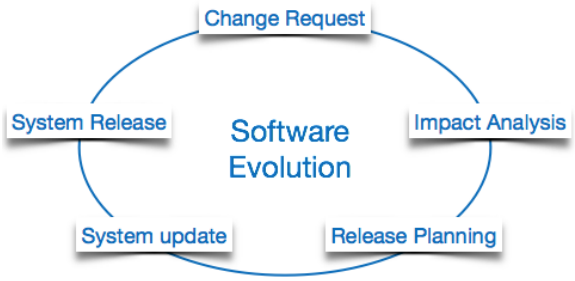
Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

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 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 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 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:

- **S-type (static-type) -** This is a software, which works strictly according to defined specifications and solutions. 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.
- **P-type (practical-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 obvious instantly. For example, gaming software.
- **E-type (embedded-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 unless work 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 systems and 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.

➢ Legacy software

Legacy software is an older version of a program or application that still functions even after newer updates or versions are available. Companies typically use legacy software when they have older technology systems. Older computers are often only compatible with software of a similar age, so it's easier for companies to use legacy software instead of purchasing new equipment. While using legacy software, IT professionals need to observe extra security protocols, perform frequent data backups, convert files to compatible formats with new software and drivers and purchase additional storage.

Who uses legacy software?

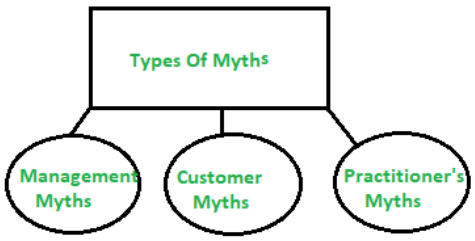
Any company that relies on software and data systems may use legacy software to maintain an archive of business operations. IT professionals often work with legacy software since they typically manage business information systems. Business administrators may also work with legacy systems to access important information and business files to update new systems.

Here are some companies that may use legacy software:

- **Background checking organizations:** Industries that perform background checks, like law enforcement and human resources (HR), may use legacy software due to the high volume of information and the complexity of the background checking system. Transferring the information to a newer system is challenging for organizations since data loss is risky.
- **Banks:** It's common for banks to use legacy software since they handle accounts and transactions over long periods. They may use outdated software to ensure their accounts and transactions remain unchanged.
- **Retail:** Companies in the retail industry may use legacy software so that they don't have to update all of their sales terminals. For example, if a grocery store uses older cash registers, it may cost less to repair them than to buy new ones for the entire store

➢ Software Myths:

Most, experienced experts have seen myths or superstitions (false beliefs or interpretations) or misleading attitudes (naked users) which creates major problems for management and technical people. The types of software-related myths are listed below.



(i) Management Myths:

Myth 1:

We have all the standards and procedures available for software development.

Fact:

- Software experts do not know all the requirements for the software development.
- And all existing processes are incomplete as new software development is based on new and different problem.

Myth 2:

The addition of the latest hardware programs will improve the software development.

Fact:

- The role of the latest hardware is not very high on standard software development; instead (CASE) Engineering tools help the computer, they are more important than hardware to produce quality and productivity.
- Hence, the hardware resources are misused.

Myth 3:

- With the addition of more people and program planners to Software development can help meet project deadlines (If lagging behind).

Fact:

- If software is late, adding more people will merely make the problem worse. This is because the people already working on the project now need to spend time educating the newcomers, and are thus taken away from their work. The newcomers are also far less productive than the existing software engineers, and so the work put into training them to work on the software does not immediately meet with an appropriate reduction in work.

(ii)Customer Myths:

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.

Myth 1:

A general statement of intent is enough to start writing plans (software development) and details of objectives can be done over time.

Fact:

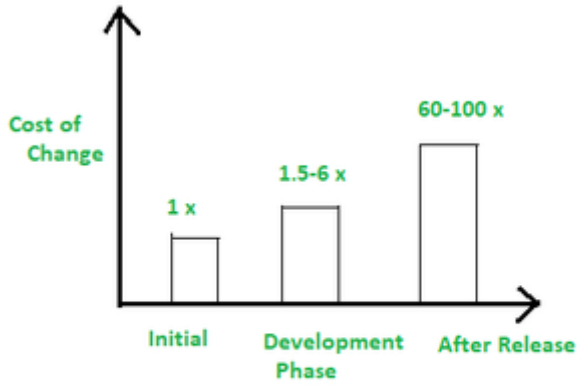
- Official and detailed description of the database function, ethical performance, communication, structural issues and the verification process are important.
- Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous
  communication between customer and developer.

Myth 2:

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

Fact:

- It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.



Cost of Change

Different Stages of Myths

(iii)Practitioner's Myths:

Myths 1:

They believe that their work has been completed with the writing of the plan.

Fact:

- It is true that every 60-80% effort goes into the maintenance phase (as of the latter software release). Efforts are required, where the product is available first delivered to customers.

Myths 2:

There is no other way to achieve system quality, until it is "running".

Fact:

- Systematic review of project technology is the quality of effective software verification method. These updates are quality filters and more accessible than test.

Myth 3:

An operating system is the only product that can be successfully exported project.

Fact:

- A working system is not enough, the right document brochures and booklets are also required to provide guidance & software support.

Myth 4:

Engineering software will enable us to build powerful and unnecessary document & always delay us.

Fact:

- Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times

➢ Layered Technology in Software Engineering

Software engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfilment of the previous layer.

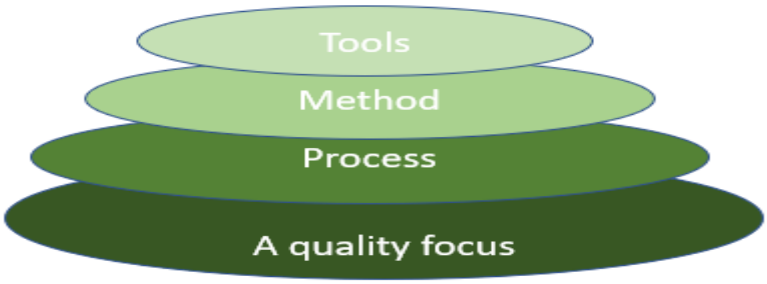
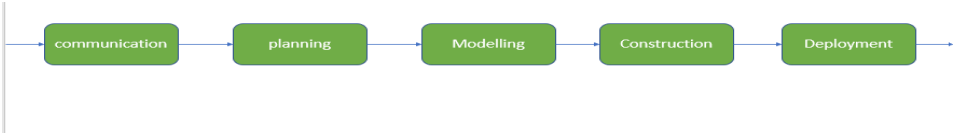


Fig: The diagram shows the layers of software development

Layered technology is divided into four parts:

1. A quality focus: It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. Process: It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development.



Process activities are listed below:-

- **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
- **Planning:** It basically means drawing a map for reduced the complication of development.
- **Modeling:** In this process, a model is created according to the client for better understanding.
- **Construction:** It includes the coding and testing of the problem.
- **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

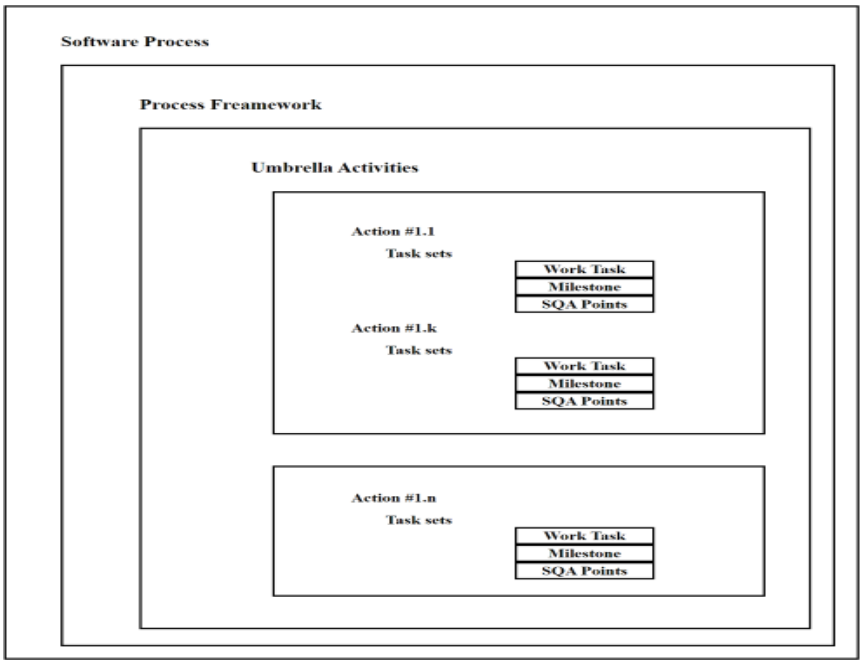
3. Method: During the process of software development the answers to all "how-to-do" questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. Tools: Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

➢ Software Process Framework

Software Process Framework is an abstraction of the software development process. It details the steps and chronological order of a process. Since it serves as a foundation for them, it is utilized in most applications. Task sets, umbrella activities, and process framework activities all define the characteristics of the software development process. Software process includes:

- Tasks – focus on a small, specific objective.
- Action – set of tasks that produce a major work product.
- Activities – group of related tasks and actions for a major objective.



Software Process Framework

Process Framework Activities:

The process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering. Communication, planning, modeling, construction, and deployment are all examples of framework activities. Each engineering action defined by a framework activity comprises a list of needed work outputs, project milestones, and software quality assurance (SQA) points.

- **Communication:** By communication, customer requirement gathering is done. Communication with consumers and stakeholders to determine the system's objectives and the software's requirements.
- **Planning:** Establish engineering work plan, describes technical risk, lists resources requirements, work produced and defines work schedule.

Modeling: Architectural models and design to better understand the problem and for work towards the best solution. The software model is prepared by:

o Analysis of requirements

o Design

Construction: Creating code, testing the system, fixing bugs, and confirming that all criteria are met. The software design is mapped into a code by:

o Code generation

o Testing

- **Deployment:** In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products.

Umbrella activities:

Umbrella Activities are that take place during a software development process for improved project management and tracking.

1. **Software project tracking and control:** This is an activity in which the team can assess progress and take corrective action to maintain the schedule. Take action to keep the project on time by comparing the project's progress against the plan.
2. **Risk management:** The risks that may affect project outcomes or quality can be analyzed. Analyze potential risks that may have an impact on the software product's quality and outcome.
3. **Software quality assurance:** These are activities required to maintain software quality. Perform actions to ensure the product's quality.
4. **Formal technical reviews:** It is required to assess engineering work products to uncover and remove errors before they propagate to the next activity. At each level of the process, errors are evaluated and fixed.
5. **Software configuration management:** Managing of configuration process when any change in the software occurs.
6. **Work product preparation and production:** The activities to create models, documents, logs, forms, and lists are carried out.
7. **Reusability management:** It defines criteria for work product reuse. Reusable work items should be backed up, and reusable software components should be achieved.
8. **Measurement:** In this activity, the process can be defined and collected. Also, project and product measures are used to assist the software team in delivering the required software.

➢ Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) is a successor of CMM and is a more evolved model that incorporates best components of individual disciplines of CMM like Software CMM, Systems Engineering CMM, People CMM, etc. Since CMM is a reference model of matured practices in a specific discipline, so it becomes difficult to integrate these disciplines as per the requirements. This is why CMMI is used as it allows the integration of multiple disciplines as and when needed.

Objectives of CMMI :

1. Fulfilling customer needs and expectations.
2. Value creation for investors/stockholders.
3. Market growth is increased.
4. Improved quality of products and services.
5. Enhanced reputation in Industry.

CMMI Representation – Staged and Continuous :

A representation allows an organization to pursue a different set of improvement objectives. There are two representations for CMMI :

- **Staged Representation :**
    - uses a pre-defined set of process areas to define improvement path.

- Provides a sequence of improvements, where each part in the sequence serves as a foundation for the next.
- An improved path is defined by maturity level.
- Maturity level describes the maturity of processes in organization.
- Staged CMMI representation allows comparison between different organizations for multiple maturity levels.

- **Continuous Representation :**
  - Allows selection of specific process areas.
  - Uses capability levels that measures improvement of an individual process area.
  - Continuous CMMI representation allows comparison between different organizations on a process-area-by-process-area basis.
  - Allows organizations to select processes which require more improvement.
  - In this representation, order of improvement of various processes can be selected which allows the organizations to meet their objectives and eliminate risks.

CMMI Model – Maturity Levels :

In CMMI with staged representation, there are five maturity levels described as follows :

1. **Maturity level 1 : Initial**
   - Processes are poorly managed or controlled.
   - Unpredictable outcomes of processes involved.
   - Ad hoc and chaotic approach used.
   - No KPAs (Key Process Areas) defined.
   - Lowest quality and highest risk.
2. **Maturity level 2 : Managed**
   - Requirements are managed.
   - Processes are planned and controlled.
   - Projects are managed and implemented according to their documented plans.
   - This risk involved is lower than Initial level, but still exists.
   - Quality is better than Initial level.
3. **Maturity level 3 : Defined**
   - Processes are well characterized and described using standards, proper procedures, and methods, tools, etc.
   - Medium quality and medium risk involved.
   - Focus is process standardization.
4. **Maturity level 4 : Quantitatively managed**
   - Quantitative objectives for process performance and quality are set.
   - Quantitative objectives are based on customer requirements, organization needs, etc.
   - Process performance measures are analyzed quantitatively.
   - Higher quality of processes is achieved.
   - lower risk
5. **Maturity level 5 : Optimizing**
   - Continuous improvement in processes and their performance.
   - Improvement has to be both incremental and innovative.
   - Highest quality of processes.

- Lowest risk in processes and their performance.

CMMI Model – Capability Levels

A capability level includes relevant specific and generic practices for a specific process area that can improve the organization's processes associated with that process area. For CMMI models with continuous representation, there are six capability levels as described below :

1. **Capability level 0 : Incomplete**
   - Incomplete process – partially or not performed.
   - One or more specific goals of process area are not met.
   - No generic goals are specified for this level.
   - This capability level is same as maturity level 1.
2. **Capability level 1 : Performed**
   - Process performance may not be stable.
   - Objectives of quality, cost and schedule may not be met.
   - A capability level 1 process is expected to perform all specific and generic practices for this level.
   - Only a start-step for process improvement.
3. **Capability level 2 : Managed**
   - Process is planned, monitored and controlled.
   - Managing the process by ensuring that objectives are achieved.
   - Objectives are model and other including cost, quality, schedule.
   - Actively managing processing with the help of metrics.
4. **Capability level 3 : Defined**
   - A defined process is managed and meets the organization's set of guidelines and standards.
   - Focus is process standardization.
5. **Capability level 4 : Quantitatively Managed**
   - Process is controlled using statistical and quantitative techniques.
   - Process performance and quality is understood in statistical terms and metrics.
   - Quantitative objectives for process quality and performance are established.
6. **Capability level 5 : Optimizing**
   - Focuses on continually improving process performance.
   - Performance is improved in both ways – incremental and innovation.
   - Emphasizes on studying the performance results across the organization to ensure that common causes or issues are identified and fixed.