Overview [week-1 theory]

Software Engineering

The term software engineering is the product of two words, software, and engineering.

- The **software** is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.
- **Engineering** is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.

Definition:

Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

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

Need of Software Engineering

The necessity of software engineering appears because of a higher rate of progress in user requirements and the environment on which the program is working.

1) **Huge Programming:** It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.

2) **Adaptability:** If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.

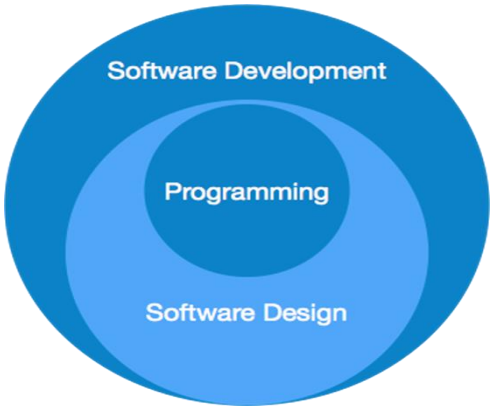
3) **Cost:** As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware. But the cost of programming remains high if the proper process is not adapted.

4) **Dynamic Nature:** The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.

5) **Quality Management:** Better procedure of software development provides a better and quality software 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 in work today, but we need to see where in the software engineering these paradigms stand.
- These can be combined into various categories, though each of them is contained in one another.



Programming Paradigm

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

- Coding
- Testing
- Integration

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 which helps the software product to build. It consists of -

- Requirement gathering
- Software design
- Programming

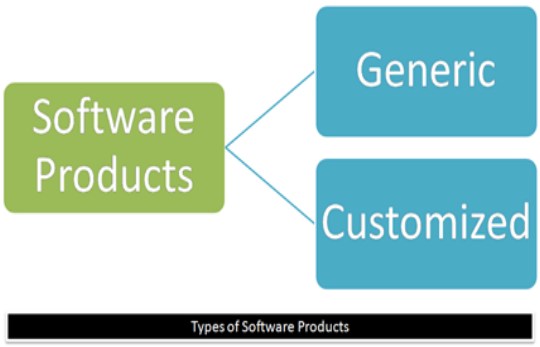
Software Design Paradigm

This paradigm is a part of Software Development and includes –

- Design
- Maintenance
- Programming

Software product types: generic, customized

Computer Software is the product that software engineers design and build. Software products are software systems delivered to a customer with the documentation which describes how to install and use the system.



Generic

These are stand-alone systems that are produced by software development companies and sold on the open market to any customer who needs them and able to buy them.

Customized

These are systems that are commissioned by a particular customer. The software is developed especially for that customer by some developer.

Differences between the generic software and custom software

The major difference between a generic software product and custom software product is the control over the development process.

Comparison parameter	Generic Software	Custom Software
Functionality	Generic software has functionality designed to solve a particular problem for many entities in that vertical.	Custom software has functionality made to solve a problem for a specific entity.
Specifications	Specifications are produced internally by the marketing department of the product company.	According to the contract between customer and developer.
Quality	Generic software product quality is not the main parameter a development company follows.	Quality is the main criterion in the customer software product.
Number of users	Large	Limited
Marketing	Marketing is required.	No marketing is required.
Needs and updates	Defined by market demand. A generic software product is made based on future updates.	Customized software is done according to the time, budget and needs defined by the customer.
Development cost	Usually not high.	High (for a single customer)
Some examples	Web browser Word processor Presentation Database applications	Banking Services Voice Recognition Delivery Services Streaming Services

Characteristics of good software

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

- Operational
- Transitional
- Maintenance

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

Operational

This tells us how well 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 a 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 concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

Challenges in software projects

Here is the list of some top challenges every Software Product Developer face:

1) Changing Requirements during the development process brings challenges for the software developers. Sometimes they won't be able to deal with changing requirements.

2) Providing complete Security to the software applications is a major challenge for developers as hackers are trying each moment there to hack the software applications and to steal the data.

3) Many times, software developers face problem during System and Application integration leading to failure of software projects also.

4) Further Maintenance and Upgradation becomes a problem for software developers for some software projects.

5) Adapting latest Technology becomes a big challenge for the software developers when they don't have sufficient experience on latest market trends.

6) Getting Defects or Errors in the product during its last stage creates an unwanted challenge for the software developers.

7) Time limitations plays a vital role in software development. When there is no sufficient time for the development sometimes the product doesn't meet the quality standards as the developers works under pressure and output decreases.

8) When a new developer lacks proper Communication and Coordination with the other developers of the same development team it creates a problem at some point.

9) It feels like a common problem when one developer Works with another developer's code This situation creates a problem for the developer as it takes lot time of the new developer to understand the code.

10) In last most of the software developers face this problem if they Don't get required support from Project Manager/Leader and sometimes it gets difficult to handle the relation between colleagues and managers which in terms decrease the productivity.

Factors that influence software development

1) **Portability:** A software product is said to be portable, if it can be easily made to work in different operating system environments, in different machines, with other software products, etc.

2) **Usability:** A software product has good usability, if different categories of users (i.e. both expert and novice users) can easily invoke the functions of the product.

3) **Reusability**: A software product has good reusability, if different modules of the product can easily be reused to develop new products.

4) **Correctness:** A software product is correct, if different requirements as specified in the SRS document have been correctly implemented.

5) **Maintainability:** A software product is maintainable, if errors can be easily corrected as and when they show up, new functions can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Understanding success Software process

Definition: A software process is a set of related activities that leads to the production of a software product.

These activities may involve the development of the software from the scratch, or, modifying an existing system.

Need of process

The software development process provides guidelines to achieve management control. SDLC is a process followed to develop a software project, within a software organization. By performing a perfect SDLC, software developers can create a functional business system.

Components of process

There are three components of the software:

- Program
- Documentation
- 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.

Process Activities

There are four basic key process activities:

1. **Software Specifications:** In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
2. **Software Development:** In this process, designing, programming, documenting, testing, and bug fixing is done.
3. **Software Validation:** In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end user's needs.
4. **Software Evolution:** It is a process of developing software initially, then timely updating it for various reasons.

Four Activities of Software Process Framework

Differentiate product, project and process

A **software process** as mentioned earlier, specifies a method of development software.
A **software project,** on the other hand is a development project in which a software process is used.
And **software products** are the outcomes of a software project.
Each **software development project** starts with some needs and (hopefully) ends with some software that satisfies those needs.
A **software process** specifies the abstract set of activities that should be performed to go from user needs to final product.

The actual act of executing the activities for some specific user needs is a **software project.** And all the outputs that are produced while the activities are being executed are the **products.**

Process assessment and improvement

The existence of a software process is no guarantee that software will be delivered on time, that it will meet the customer's needs, or that it will exhibit the technical characteristics that will lead to long-term quality characteristics.

A number of different approaches to software process assessment and improvement have been proposed over the past few decades:

1. **Standard CMMI Assessment Method for Process Improvement (SCAMPI) -** provides a five-step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting, and learning. The SCAMPI method uses the SEI CMMI as the basis for assessment.
2. **CMM-Based Appraisal for Internal Process Improvement (CBA IPI) -** provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM (Software Engineering Institute Capability Maturity Model) as the basis for the assessment.
3. **SPICE (Software Process Improvement and Capability Determination) (ISO/IEC15504) -** a standard that defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of any defined software process.
4. **ISO 9001:2000 for Software -** a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides.

Software engineering ethics

- Like other engineering disciplines, software engineering is carried out within a social and legal framework that limits the freedom of people working in that area.
- As a software engineer, you must accept that your job involves wider responsibilities than simply the application of technical skills.
- You must also behave in an ethical and morally responsible way if you are to be respected as a professional engineer. It goes without saying that you should uphold normal standards of honesty and integrity.
- You should not use your skills and abilities to behave in a dishonest way or in a way that will bring disrepute to the software engineering profession.

1) **Confidentiality:** You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
2) **Competence:** You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
3) **Intellectual property rights:** You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.
4) **Computer misuse:** You should not use your technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses or other malware).