Subject: Software Engineering (BIT302)

Unit: 1- Introduction

Software engineering

Software Engineering: 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.

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

- **Large software**: It is easier to build a wall than to a house or building likewise, as the size of software become large engineering has to step to give it a scientific process.
- **Scalability**: If the software process were not based on scientific and engineering 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 manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- **Dynamic Nature**: The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- **Quality Management**: Better process of software development provides better and quality software product.

Computer Science and Software Engineering

Computer science is concerned with theories and methods that underlie computers and software system. But software engineering is concerned with the practicalities of developing and delivering useful software. Some Knowledge of Computer Science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers.

Software Engineering and System Engineering

System engineering is concerned with all aspect of the development and evolution of complex systems where software plays a major role. So, it mainly concerned with, hardware development, policy and process design and system development as well as software engineering. Software engineering is only part of this process.

Introduction to software

Computer software, or only software, is a kind of program that enables a user to perform some specific task or used to operate a computer. It directs all the peripheral devices on the computer system – what to do and how to perform a task. PC Software plays the role of mediator between the user and computer hardware. Without software, a user can't perform any task on a digital computer.

A computer system can be divided into three components: the hardware, the software, and the users. Software can be further divide into mainly two parts: **Application software and System Software.**

Application Software

Applications software also called **end-user programs** or merely an application. It resides above system software. The end-user uses applications software for a specific purpose. It programmed for simple as well as complex tasks. It either be installed or access online. It can be a single program or a group of small programs that referred to as an application suite. Application software can be used by the user to complete specific tasks, such as creating word processors documents, spreadsheets, presentations, graphics, CAD/CAM, sending the email, etc.

System Software

System Software (the type of **computer program**) provides a platform to run a computer's hardware and computer application to utilize system resources and solve their computation problem. It is written in a low-level language, like assembly language, so it can easily interact with hardware with the primary level. It controls the working of peripheral devices. System software act as a scheduler for the execution of the processes and arrange the sequence according to their priority and I/O devices requirement and creation of the process. The best-known example of system software is the operating system (OS). It responsible for manages all the other programs on a computer.

Programs v/s Software

Software is a broad term that covers the programs and components that it required to run. Software consists the files, whereas a program can itself be a file. Along with these differences, there are various other comparisons between both terms.

On the basis of	Program	Software
Definition	A computer program is a set of instructions that is used as a process of creating a software program by using programming language.	Software is a set of programs that enables the hardware to perform a specific task.
Types	Programs do not have further categorization.	The software can be of three types: system software, application software, and programming software.
User Interface	A program does not have a user interface.	Every software has a user interface that may be in graphical format or in the form of a command prompt.
Size	Programs are smaller in size, and their size exists between Kilobyte (Kb) to a megabyte (Mb).	Software's are larger in size, and their size exists between megabytes (Mb) to gigabytes (Gb).
Time taken	A program takes less time to be developed.	Whereas software requires more time to be developed.
Features and functionality	A program includes fewer features and limited functionalities.	It has more features and functionalities.
Development approach	The development approach of a program is unorganized, unplanned, and unprocedural.	The development approach of software is well planned, organized, and systematic.
Documentation	There is a lack of documentation in the program.	Softwares are properly documented.

Characteristics of Software

Software is defined as a collection of computer programs, procedure, rules and data. Software Characteristics are classified into six major components:

Functionality: It refers to the suitability, accuracy, interoperability, compliance, security of software which is measured as degree of performance of the software against its intended purpose.

Reliability: Refers to the recoverability, fault tolerance, maturity of software, which is basically a capability of the software that provides required functionality under the given situations.

Efficiency: It is the ability of the software to use resources of system in the most effective and efficient manner. Software must make effective use of system storage and execute command as per required timing

Usability: It is the extent to which the software can be utilized with ease and the amount of effort or time required to learn how to use the software.

Maintainability: It is the ease with which the modifications can be made in software to extend or enhance its functionality, improve its performance, or resolve bugs.

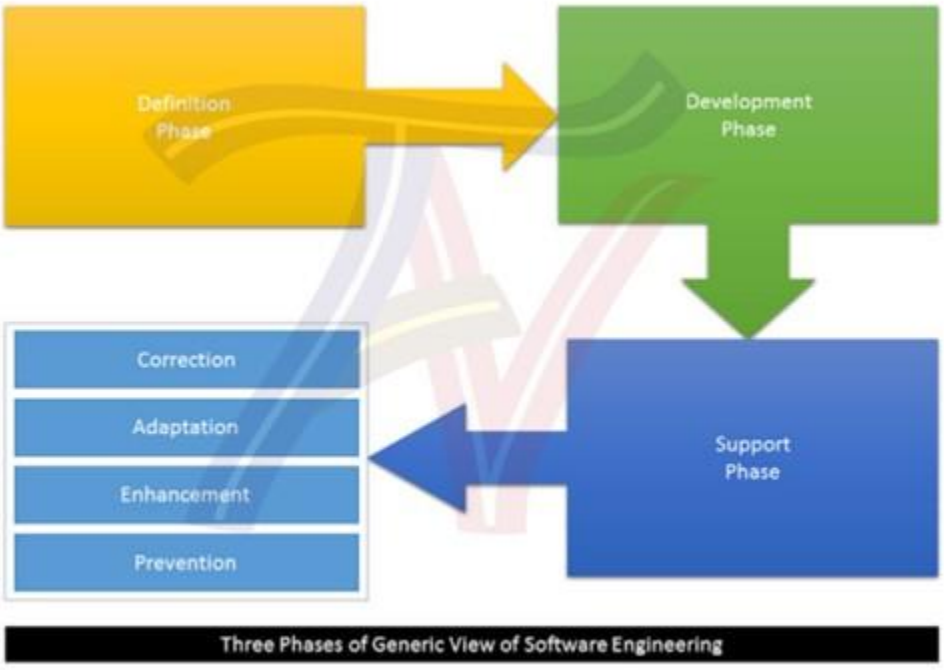
Portability:

It is the ease with which software developers can prelaunch software from one platform to another, without (or with minimum) changes. In simple terms, software must be made in way that it should be platform independent.

Apart from above mention qualities of software, there are various characteristics of software in software engineering:

- Software is developed or engineered; it is not manufactured in the classical sense
- The software doesn't "wear out."
- The software continues to be custom-built:

Generic View of Software Engineering



1. Definition Phase:

The definition phase focuses on "what". That is, during definition, the software engineer attempts to identify what information is to be processed, what function and performance are desired, what system behavior can be expected, what interfaces are to be established, what design constraints exist, and what validation criteria are required to define a successful system. During this, three major tasks will occur in some form: system or information engineering, software project planning and requirements analysis.

2. Development Phase:

The development phase focuses on "how". That is, during development a software engineer attempts to define how data are to be structured, how function is to be implemented within a software architecture, how interfaces are to be characterized, how the design will be translated into a programming language, and how testing will be performed. During this, three specific technical tasks should always occur; software design, code generation, and software testing.

3. Support Phase:

The support phase focuses on "change" associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Four types of change are encountered during the support phase

Correction: Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Adaptation: Over time, the original environment (e.g., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate changes to its external environment.

Enhancement: As software is used, the customer/user will recognize additional functions that will provide benefit. Perfective maintenance extends the software beyond its original functional requirements.

Prevention: Computer software deteriorates due to change, and because of this, preventive maintenance, often called software reengineering, must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

Professional Software Development

Software engineering is intended to support professional software development rather than individual programming. It includes techniques that support program specification, design, and evolution, none of which are normally relevant for personal software development.

Many people think that software is simply another word for computer programs. However, when we are talking about software engineering, software is not just the programs themselves but also all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful. A professionally developed software system is often more than a single program. A system may consist of several separate programs and configuration files that

are used to set up these programs. It may include system documentation, which describes the structure of the system, user documentation, which explains how to use the system, and websites for users to download recent product information.

This is one of the important differences between professional and amateur software development. If you are writing a program for yourself, no one else will use it and you don't have to worry about writing program guides, documenting the program design, and so on. However, if you are writing software that other people will use and other engineers will change, then you usually have to provide additional information as well as the code of the program.

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. However, there are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

1. **Confidentiality** You should normally respect the confidentiality of your employers or clients regardless 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) to extremely serious (dissemination of viruses or other malware).

Professional societies and institutions have an important role to play in setting ethical standards. Organizations such as the ACM, the IEEE (Institute of Electrical and Electronic Engineers), and the British Computer Society publish a code of professional conduct or code of ethics. Members of these organizations undertake to follow that code when they sign up for membership. These codes of conduct are generally concerned with fundamental ethical behavior.