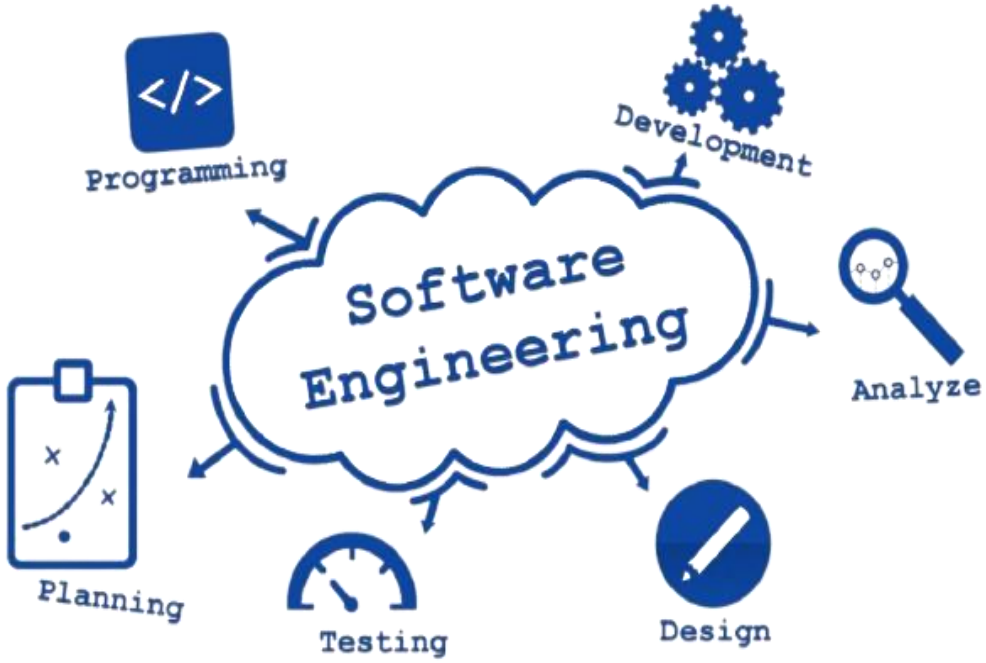
Unit – 1 Introduction to Software Engineering



1.1 Software and Software Engineering

1.2 Evolution of Software

1.3 Software Characteristics

1.4 Software Applications

1.5 Software Myths

1.6 Software Process

1.7 Software Development Life Cycle (SDLC)

Prof. A. P. Chaudhari (M.Sc., SET)
HOD, Department of Computer Science
SVS's Dadasaheb Rawal College, Dondaicha

1.1) Software and Software Engineering:

The term **Software Engineering** 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.

1.1) Software and Software Engineering:

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.

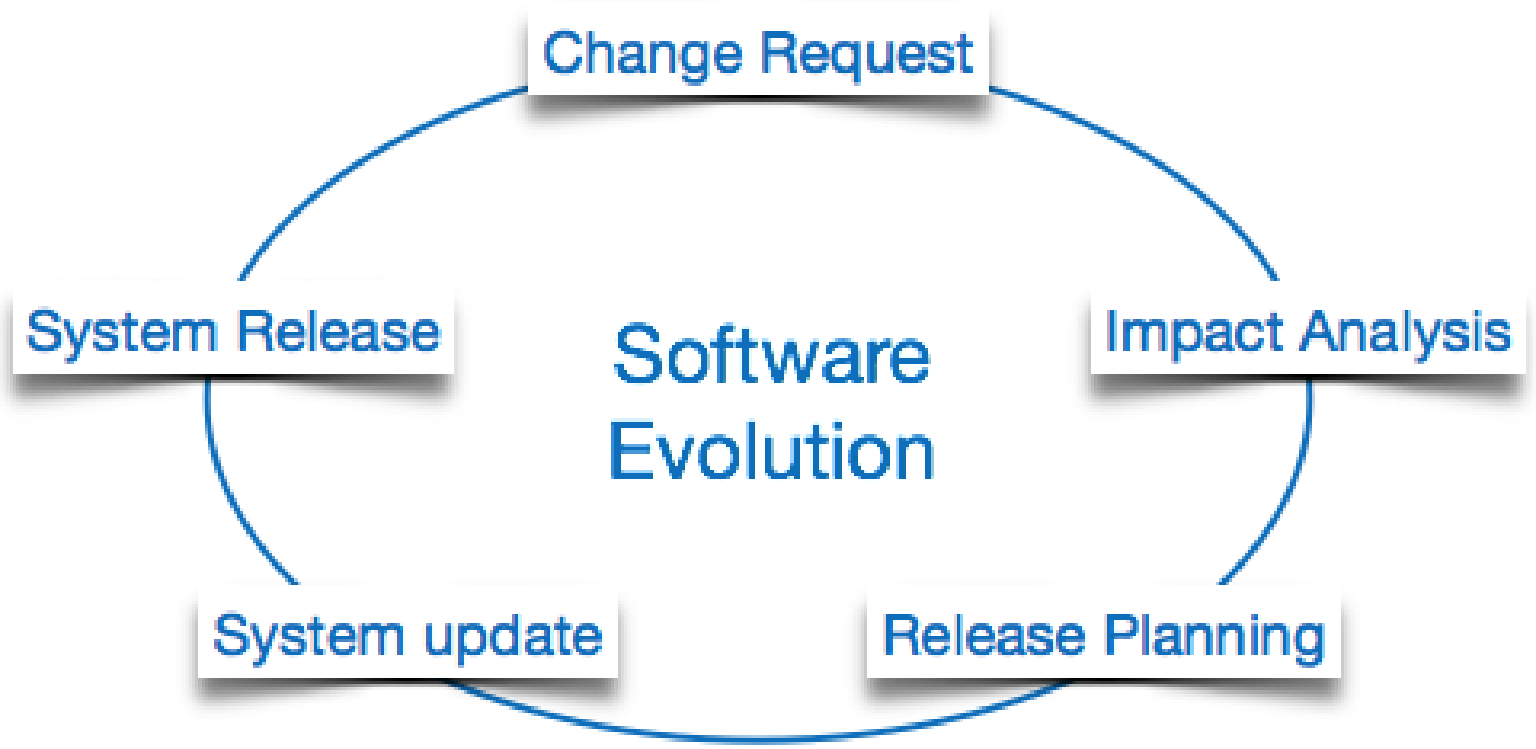
Definition:

IEEE defines software engineering as:

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

1.2) Evolution of Software:

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.



1.2) Evolution of Software:

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 the 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 the 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 the requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

1.3) Software Characteristics:

To gain an understanding of software, it is important to examine the characteristics of software that make it different from other things that human being build the human creative process is ultimately translated into physical form. Software is logical rather than physical system element. Its behavior and nature is quite different than other products of human life.

Some of the important characteristics of software are discussed below:

1) Software does not wear out:

There are three phases for the life of a hardware product. Initial phase is burn-in phase where failure intensity is high before delivery it is expected to test the product. Due to testing, failure intensity will come down initially. Second phase is useful life phase where failure is constants. Third phase is wear out phase in which again intensity will increase due to wearing out of components.

1.3) Software Characteristics:

Important point is software become reliable overtime instead of wearing out. It becomes obsolete (outdated), if the environment for which it was developed. Hence software may be retired due to environmental changes, new requirements, new expectations, etc.

2) Software is not manufactured:

Software or hardware both get manufactured in the same manner and both of them users the design model to implement the product. The only difference is in their implementation part. They both differ in their coding part so it is said that software is not manufactured but it is developed or engineered.

3) Reusability of Components:

If we have to manufacture a TV, we may purchase picture tube from one vendor, cabinet from another and other electronic components from third vendor.

1.3) Software Characteristics:

We will assemble every part and test the product thoroughly to produce a good quality TV. We may have standard quality guidelines and effective processes to produce a good quality product. Reusability is the common component which we use in our process. During software development, we don’t need to start writing the code from the beginning. We can use already developed software to develop another one.

4) Software is flexible:

A program can be developed to do almost anything. Software can be developed for solving any type of problem and can also be change if the requirement changes.

1.4) Software Applications:

Software may be applied in any situation for which a pre-specified set of procedural steps has been defined. Information content is important factors in determining the nature of a software application. Content refers to the meaning and form of incoming an outgoing information. For example many business application make use of highly structured input data and produce formatted “Reports”, Software that controls an automated machine accepts discrete data items with limited structure and produces individual machine commands in rapid succession.

It is somewhat difficult to develop meaningful generic categories for software applications. The following software areas indicate the breadth of potential applications:

1.4) Software Applications:

1) System Software: System software is a collection of programs written to service other programs. Some system software (e.g. compilers, editors and file management utilities) processes complex but determinate information structures. Other systems application (e.g. operating system components drivers telecommunications processors) process largely indeterminate data. In either case the systems software area is characterized by heavy interaction with computer hardware heavy usage by multiple users; concurrent operation that requires scheduling resource sharing and sophisticated process management ; complex data structures and multiple external interfaces.

1.4) Software Applications:

2) Business Software: Business information processing is the largest single software application area. Discrete “systems” (e.g., payroll accounts receivable/payable inventory, etc.,) have evolved into management information system (MIS) software that accesses one or more large databases containing business information. Applications in this area restructure existing data in a way that facilitates business operation or management decision making.

3) Engineering and Scientific Software: Engineering and Scientific software has been characterized by “number of” algorithms. Application range from astronomy to volcanology and from molecular biology to automated manufacturing. However new applications with the engineering/scientific area are moving away from traditional numerical algorithms. Computer aided design (CAD) system simulation and other interactive applications have begun to take on real-time and even system software characteristics.

1.4) Software Applications:

4) Embedded Software: Intelligent products have become commonplace in nearly every consumer and industrial market. Embedded software resides in read only memory and is used to control products and systems for the consumer and industrial markets. Embedded software can perform very limited and secret functions (e.g. digital functions in an automobile such as fuel control, dashboard displays, etc.).

5) Personal Computer Software: The personal computer software market has growing over the past decade. Word processing, spreadsheets, computer graphics, multimedia entertainment, database management personal and business financial applications and external network or database access are only a few of hundreds of application.

1.4) Software Applications:

6) Artificial Intelligence Software: Artificial Intelligence (AI) software makes use of non numerical logarithms to solve complex problems. An active AI area is expert systems also called knowledge-based systems. However other application areas for AI software are pattern recognition (image and voice) theorem proving and game playing. In recent years a new branch of AI software called artificial neural networks, has evolved.

7) Real-Time Software Programs that monitor/analyze/ control real world events as they occur are called real-time software. Elements of real-time software include a data gathering component that collects and formats information from an external environment an analysis component that transforms information as required by the application a control / output component that responds to the external environment so that real-time response (typically ranging from 1 millisecond to 1 minute) can be maintained.

1.5) Software Myths:

Today most knowledgeable professionals recognize myths for what they are—misleading attitudes that have caused serious problems of managers and technical people alike. However old attitudes and habits are difficult to modify software myths are still believed.

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

Myth: We already have a book that's full of standard and procedures for building software. Won't that provide my people with everything they need to know?

Reality: The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software development practice? Is it complete? In many cases the answer to all of these question is "no".

1.5) Software Myths:

Myth: My people do have state of the art software development tools, After all we buy them the newest computers.

Reality: It takes much more that the latest model mainframe workstation or PC to do high quality software development. Computer aided software engineering (CASE) tools are more important than hardware for achieving good quality and productivity yet the majority of software developers still do not use them.

Myth: If we get behind schedule we can add more programmers and catch up.

Reality: Software development is not a mechanistic process like manufacturing. In the other words, “adding people to a late software project makes it later”. However as new people are added people who were working must spend time education the newcomers hereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well coordinated manner.

1.5) Software Myths:

Consumer Myths: A customer who request computer software may be a person at the next desk, a technical group down the hall the marketing /sales department or an outside company that has requested software under contract. In many cases the customer believes myths about software because software responsible managers and practitioners do little to correct misinformation (Myths) lead to false

expectations (by the consumer) and ultimately dissatisfaction with the developer.

Myth: A general statement of objectives is sufficient to begin writing programs we can fill in details later.

Reality: poor up-front definition is the major cause of failed software efforts. A formal and detailed description of information domain, function performance interfaces, design constraints and validation criteria is essential. These characteristic can be determined only after through communication between customer and developer.

1.5) Software Myths:

Myth: Project requirements continually change, but change can be easily accommodated because software is flexible.

Reality: It is true that software requirements do change, but the impact of change varies with the time at which it is introduced. If serous attention is given to up-front definition early requests for change can be accommodated easily. The customer can review requirements and recommend modification with relatively little impact on cost. When changes are requested during software design cost impact grows rapidly. Resources have been committed and a design framework has been established. Change can cause upheaval that requires additional resources and major design modification i.e., additional cost, Changes in function, performance interfaces or other characteristics during implementation (code and test) have a severe impact on cost. Change, when requested after software is in production use can be more than the same change requested earlier.