CHAPTER

# 1 Introduction to Software Engineering

**Structure**



## 1.1 INTRODUCTION

Software Engineering is essential for understanding how to build good error free software at lesser price, in time and for evaluating the risks and opportunities that software presents in our everyday lives. Today, every sector of the economy depends on computers to a greater extent. Software is a key element of any computer-based system. Development of even a piece of software requires many activities to be performed and it is regarded as a project. Development of software requires a systematic approach. All the engineering aspects relating to software development have combined together to evolve as a discipline called software engineering.

In this unit, we are going to study about what is software engineering, software engineering principles, their characteristics and applications. We will also learn about objectives of software engineering and phases of software engineering.

### Objectives

After studying this unit, you should be able to:

- Define software engineering
- Discuss software engineering principles
- List software characteristics
- Discuss various kinds of software applications
- List the objectives of software engineering
- Explain phases of software engineering

## 1.2 BASICS OF SOFTWARE ENGINEERING

The term software engineering was first introduced in 1968 North Atlantic Treaty Organization (NATO) conference held in Germany. Software Engineering is an engineering discipline whose focus is the cost-effective development of high-quality software systems. It is a sub discipline of Computer Science that attempts to apply engineering principles to the creation, operation, modification and maintenance of the software components of various systems.

Software engineering is concerned with the practicalities of developing and delivering useful software. The cost of software engineering includes roughly 60% of development costs and 40% of testing costs. Structured approaches to software development include system models, notations, rules, design advice and process guidelines. Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software are the key challenges facing Software Engineering.

**What is an engineering?**

Engineering is an application of well-understood scientific methods to the construction, operation, modification and maintenance of useful devices and systems.

**What is software?**

Software is not just the programs but also all associated documentation and configuration data that is needed to make the programs to operate correctly. A software system usually consists of a number of separate programs, configuration files which are used to set up the programs, system documentation which describe the structure of the system and user documentation which explains how to use the system and websites for users to download the information.

**Systems**

A system is a grouping of components that interact in some manner among themselves and possibly, with the world outside the system boundary.

We understand systems by decomposing them into:

Subsystems

System components

It is very difficult to separate the software components of a system from the other components of a system.

## 1.3 PRINCIPLES OF SOFTWARE ENGINEERING

Alan Davis (1994) is one of the earlier authorities to bring forward a set of principles that underlie software engineering. Below are some principles of software engineering:

- **Give products to customers early**: In this principle, it is very difficult to completely understand and capture the user's needs during the requirement phase; thus it is more effective to give the users a prototype of the product, then gather the feedback and then go into full-scale development of the product.

- **Determine the problem before writing the requirements**: In this principle, before the software engineering rush to offer the solution, ensure that the problem is well understood. Then explore the potential solution and various alternatives.
- **Evaluate design alternatives**: In this principle, after the requirements are understood and agreed upon, explore a variety of design architecture and related algorithms. Ensure that the selected design and algorithms are the best match to satisfy the goals of the requirement.
- **Use an appropriate process model**: In this principle, since there is no universal process model that applies to all projects, each project must select a process that best fits the project based on parameters such as corporate culture, project circumstances, user expectations, requirements volatility and resource experiences.
- **Put technique before tools**: In this principle, before using the tools, the technique needs to be well understood. Otherwise, the tool just rushes us into performing the wrong thing faster.
- **Get it right before you make it faster**: In this principle, it is essential to make the software execute correctly first and then work on improving it. Don't worry about optimization for either execution speed or code during initial coding.
- **Inspect code**: In this principle, inspection as first proposed by IBM's Mike Fagan is a much better way to find errors. Some early data in inspection showed a reduction of 50% to 90% of the time-to-test.
- **Good management is more important than good technology**: In this principle, a good manager can produce extraordinary results even with limited resources.
- **People are the key to success**: In this principle, software is a labor-intensive profession and people with experience, talent and appropriate drive are the key. The right people can overcome many of the shortcomings in process, methodology or tools. There is no substitute for quality people.
- **Follow with care**: In this principle, be careful in adopting tools, process, methodology and so on. Do not follow just because someone else is doing it or using it. Run some experiments before making a major commitment.
- **Take responsibility**: In this principle, if you developed the system, then you should take responsibility to do it right. Blaming the failure or the problem on others, on the schedule or on the process is irresponsible.

## 1.4 SOFTWARE CHARACTERISTICS

Software is logical rather than a physical system element. Hence software has different characteristics than that of hardware:

(a) Software is not manufactured as in the traditional sense rather it engineered and developed.
(b) Software don't "wear down".
(c) Although the industry is moving toward component-based assembly, most software continues to be custom built.

## 1.5 SOFTWARE APPLICATIONS

Software may be applied in any situation for which a prespecified set of procedural steps that has been defined. Content and determinacy are important factors in determining the nature of a software application. Content refers to the information. Software that controls an automated machine accepts discrete data items with limited structure and produces individual machine commands in rapid succession.

Information determinacy refers to the predictability of the order and timing of information. An engineering analysis program accepts data that have a predefined order, executes the analysis algorithm without interruption and produces resultant data in report or graphical format. Such applications are determinate.

A multi-user operating system, accepts inputs that have varied content and arbitrary timing, executes algorithms that can be interrupted by external conditions, and produces output that varies as a function of environment and time. Applications with these characteristics are indeterminate.

Software applications can be compartmentalized into different categories:

**System software:** System software is a collection of programs written to service other programs. Some system software process complex information structures. Other systems applications process largely indeterminate data. It is characterized by heavy interaction with hardware, heavy usage by multiple users, concurrent operation that requires scheduling, resource sharing, and sophisticated process management, complex data structures and multiple external interfaces.

**Real time software:** Software that checks, evaluates, and manages real time events in the world as and when they occur.

**Business Software:** Business information processing is one of the major application area. Distinct applications like payroll, accounts accept/dispatch have transformed into Management Information Systems (MIS) software gets the business information by connecting few to many databases. Software helps to streamline existing data, such that it assists business development and decision making in an organization.

**Engineering and scientific software:** This software involves typically large number of arithmetic and floating point operations. Applications range from automotive stress analysis to automated manufacturing, Artificial intelligence to Data mining, whether forecast to critical applications like where human intervention is impossible.

**Embedded software:** This software is loaded merely into primary memory and used to manage systems and products for industrial and consumer markets. This software can provide partial control and capability with necessary functions.

**Personal computer software:** Day to day useful applications like word processing, spreadsheets, multimedia, database management, personal and business financial applications are some of the common examples for personal computer software.

**Web-Based Software:** This is the innovative approach of commercial software which can be accessed by a web browser through web pages. In core the internet becomes a huge computer system providing an unlimited software resource that can be

accessed by anybody with their modem connected to the internet or wireless connection to the internet.

**Artificial intelligence software:** Artificial Intelligence software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Knowledge based expert systems, robotics, optical character recognition, automatic speech recognition and personal identification systems are the various examples of applications within this category.

**Software crisis:** The set of problems that are encountered in the development of computer software is not limited to software that does not function properly rather the affliction encompasses problems associated with how we develop software, how we support a growing volume of existing software, and how we can expect to keep pace with a growing demand for more software.

## 1.6 OBJECTIVES OF SOFTWARE ENGINEERING

The software engineering is to develop high quality software, timely, within the budget and under the software production constraints. Below are some of the basic objectives of software engineering:

- **Maintainability** – The ease with which changes in a functional unit can be performed in order to meet prescribed requirements.
- **Correctness** – The extent to which software meets its specified requirements
- **Reuseability** - The extent to which a module can be used in multiple applications.
- **Testability** – The extent to which software facilities both the establishment of test criteria and the evaluation of the software with respect to those criteria.
- **Reliability** – An attribute of software quality. The extent to which a program can be expected to perform its intended function, over an arbitrary time period.
- **Portability** - The ease with which software can be transferred from one computer system or environment to another.
- **Adaptability** – The ease with which the software allows differing system constraints and user needs to be satisfied by making changes to the software.

## 1.7 PHASES OF SOFTWARE ENGINEERING

In the engineering domain, developing a solution to a given problem involves a sequence of interconnected steps, whether building a bridge or making an electronic component. These steps or phases occur in software development as well. The series of steps involved in developing the product is called the Software Development Life Cycle (SDLC). There are seven phases or stages in SDLC, which are discussed below:

- Analysis phase
- Design phase

- Development phase
- Testing phase
- Implementation phase
- Maintenance phase
- End-of-life phase/Retirement phase

**Analysis phase**

In this phase, the development team analyzes the problem in an existing application or finds new ideas for an application. After identifying the idea, the development team needs to determine the scope of the problem. After identifying the scope, the development team can easily identify the essential components necessary for the product development. The important thing for software development is the system requirement. The system requirement differs based on the software product. The development team should carefully analyze the requirements needed for the product development.

**Design phase**

This is an important phase in the product life cycle. In this phase, the development team designs the individual components and creates the blue prints. The analysis done in the analysis phase helps to create the design documents. Designs such as database design, functional specification design, and document design take place in this phase. The design needs to be done carefully because the development phase depends upon the design created during the design phase. If the design prepared is well structured, it reduces the time taken in the upcoming stages of the product life cycle.

**Development phase**

In this phase, the actual development of the product takes place according to the blue print created in the design phase. The team members start writing code for the product. The product is divided into different modules and each member is allotted a separate module to develop the product. The code is written based on the chosen technology. Generation of code takes place after the code is developed. The code is executed after it is generated.

**Testing phase**

It is essential to test each and every product before it is launched in the market. We must test the developed product to ensure that it meets the specifications stated in the design phase. In this phase, the developed product is tested and reports are prepared. The report describes the errors in the developed products. There are different methods for testing the developed product. Black box testing, unit testing, system testing and many more methods are used for testing the product. After this phase, the life cycle again moves to the development phase to correct the errors identified in this phase. After correcting the errors, the product is again tested. This process continues till the product is found to be error-free. There are some tools available to test the developed product. We can use these tools to test a product and identify the defects. These tools are used to make the testing easy and to get an error-free product.

The test scenarios are written by the testers to check the testing needs of software application. The test scenario drives the test cases which are related to the requirements and designs. Depending upon the type of requirement and design, the test scenarios are addressed as functional and structural. The test scenario should be feasible, clear, complete, and cover all requirements. The test scenario and the test case should be prioritized as per the requirements.

After completing the product testing, versions of completed product are supplied to the clients for testing onsite. The first version is the alpha release and the corrected version is the beta release. Usually the final version is the beta release.

**Implementation phase**

This is known as First Customer Ship (FCS) in software industry. After the development and testing phase the product moves to the implementation stage. In this phase, the product is taken to the end users.

**Maintenance phase**

The software maintenance phase is the longest phase in the software life cycle. This phase is distinguished in terms of costs. About 90% of the total life cycle cost of the software is consumed in the maintenance stage. The software maintenance activity is classified into four types. They are perfective, adaptive, corrective and preventive. The maintenance of the software is not performed by the person who creates the product. In this phase, the problems raised by the customers after releasing the product are rectified. The testing team tests the product again and rectifies the errors that are raised by the customers.

**End-of-life phase/Retirement phase**

The final phase of the life cycle is the retirement phase. This phase is reached after many years of service. In this phase, the product is replaced with a newer version that has enhanced features. Thus, the old product's life cycle comes to an end. This is the end phase for the old product.

## 1.8 SUMMARY

- Software Engineering is an engineering discipline whose focus is the cost-effective development of high-quality software systems.
- Engineering is an application of well-understood scientific methods to the construction, operation, modification and maintenance of useful devices and systems.
- The principles of software engineering are: give products to customers early, determine the problem before writing the requirements, evaluate design alternatives, use an appropriate process model, put technique before tools, get it right before you make it faster, inspect code, good management is more important than good technology, people are the key to success, follow with care and take responsibility.
- Software is a logical rather than a physical system element.

- Software applications can be compartmentalized into different categories like system software, real time software, business software, engineering and scientific software, embedded software, personal computer software, web-based software, artificial intelligence software, software crisis.
- Objectives of software engineering are maintainability, correctness, reuseability, testability, reliability, portability and adaptability.
- The series of steps involved in developing the product is called the Software Development Life Cycle (SDLC). There are seven phases or stages in SDLC: Analysis phase, Design phase, Development phase, Testing phase, Implementation phase, Maintenance phase and End of life stage.