Chapter 1

Introduction to Software Engineering

Learning Outcomes

After reading this chapter, students will be able to

- Know about the introduction of software engineering
- Explain - what is Software?
- Describe the levels of Software
- To know about the characteristics of Software
- Explain the types of Software
- Evolving Role of Software
- To know about some software Components
- Describe Program Versus Software
- Explanation of Software Myths & Software Crisis
- What is Software Engineering?
- Definition of Software Engineering
- To know the Similarities and Differences between Conventional and Software Engineering
- Some ideas about Generic Views of Software Engineering
- To know the cost of Software Engineering
- Software process model description
- Explain of ethics of Software Engineering
- Represent the current trends and quality of Software Engineering

1.1 INTRODUCTION

The field of software engineering is related to the development of software. Large software needs systematic development unlike simple programs which can be developed in isolation and there may not be any systematic approach being followed.

In the last few decades, the computer industry has undergone revolutionary changes in hardware. That is, processor technology, memory technology and integration of devices have charged very rapidly. As software is required to maintain compatibility with hardware, the complexity of software also has changed much in the recent past.

In the year 1970s, the programs were small, simple in size and executed on a single central processing unit system. The development of software for such system was much easier. In the today world, high speed multiprocessor systems are available and the software's are required to be developed for the whole organization. Naturally the complexity of the software has increased many folds. Thus the need for the application of engineering technology approaches in their development is realized. The application of engineering approach of software development leads to the evolution of the area of Software Engineering.

1.2 WHAT IS SOFTWARE?

Software is a set of instructions, programs that enables the computer to perform specified task. In other words, software is nothing but binary code instructions, which control the hardware.

According to Webster's New Intercollegiate Dictionary in 1979, software is the entire set of programs, procedures and related documentation associated with a system and especially a computer system.

In 1981, the New Webster's Dictionary reworded the definition, orienting it completely to computers i.e. Software is the programs and programming support necessary to put a computer through its assigned tasks, as distinguished from the actual machine.

A more restrictive but functional definition has been introduced by Blum in 1992 i.e. software is the detailed instructions that control the operation of a computer system. Its functions are

a) Manage the organizational resources of the computer.
b) Provide tools for the human beings to take advantage of these resources.
c) Perform as an intermediary between stored information and organization.

1.3 LEVELS OF SOFTWARE

The level of software can be divided into eight subcategories:

1. Machine Micro logic
2. Supervisor or Executive
3. Operating System
4. Language Translators
5. Utility Programs
6. Inquiry, Files and Database Software
7. Programming and assembly language
8. 4GL languages and User programs such as SPSS, dbase and SQL etc.

Out of these points the machine micro logic is belongs to hardware logic. The supervisor or executive, operating system, language translators and utility programs are in system software group. The application software includes inquiry, files and database software as well as programming and assembly language. The 4GL languages and user programs such as SPSS, dbase and SQL etc are end user software.

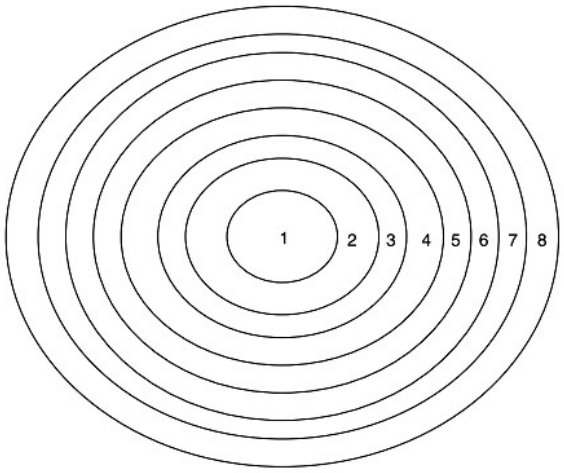

Figure 1.1: Software Levels

1.4 CHARACTERISTICS OF SOFTWARE

Software is logical more willingly than physical system elements. Therefore, the software has a distinctiveness that is significantly different from that hardware. Some of the main differences are discussed overleaf.

1.4.1 Software is developed, it is not manufactured

The concept of raw material is totally non-existent here. It is better visualized as a process than a product. In the case of software development compared to manufacturing the human element is extremely high. The development product is highly uncertain even in the case of standard products, varying greatly with the skill of developers. Each and every organization has different development tools, techniques, standards, and procedures. Qualities in software development are very different from those who are in software manufacturing units. The characteristic can be easily specified and measured in the software engineering environment.

1.4.2 Software development presents a job environment

All the products are custom built uniquely developed for every organization. It can not be assembled from any of the existing components. All the complexities of a job like problems of design, estimating of cost, and schedules exist. The most important element of the software development job is a human skill.

1.4.3 Time and effort of software development are hard to estimate

Documentation gets the least priority than interesting work. Complete the entire effort in a cleaver approach is more important than getting it ended on time and in at realistic cost-effectively. Programmers are not realistic and are optimistic; their tendency is to complete the task within their estimated time.

1.4.4 Software does not wear out

There is a well-known "bathtub curve" in reliability studies for hardware products. The curve is given in Figure 1.2. The shape of the curve is like a "bathtub" and is known as a bathtub curve.

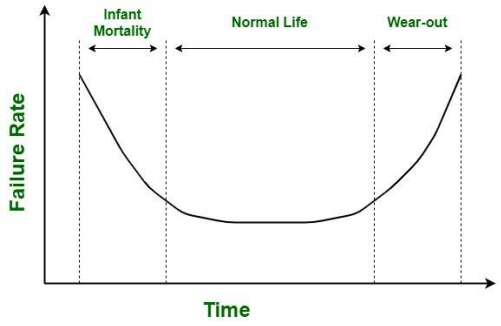

Figure 1.2: Bathtub curve

Instead, the software becomes obsolete due to the new operating environment as well as the new user environment or hardware. Hence it can only retire but not wear out. So it should follow the curve shown in Figure 1.3.

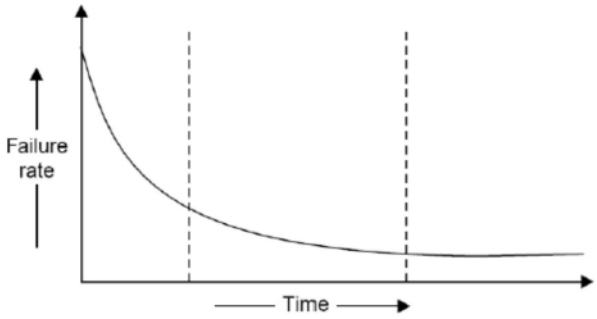

Figure 1.3: Software reliability curve

Software normally does not lose its functionality within the using stage. It may lose functionality in time when the requirements of the user get changed. When any defects are found, they are removed by rewriting the relevant code. The concept of replacing the defective code with spare code is unusual in the context of software engineering. After removing the defects, it is likely that new defects are encountered.

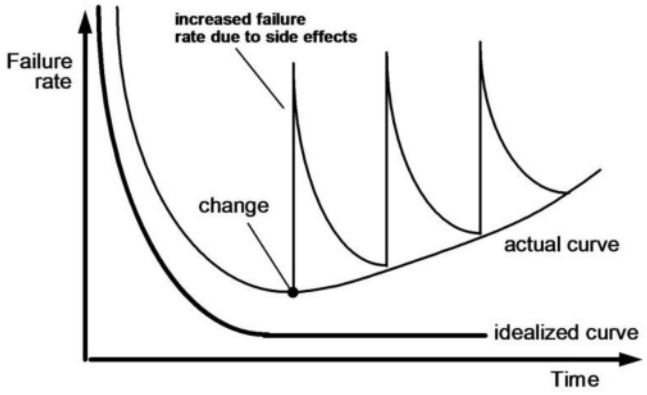

Figure 1.4: Idealized and actual failure curves for software

1.4.5 Software testing and hardware implementation

Testing of software is somehow difficult because moderate-size programming can contain enough executable paths, i.e., from the beginning to end of the program. So the process of testing of each path individually can be expensive. There are no standards or measures by which to evaluate the progress of software development. As per the

user requirements the software can be modified a number of times before implemented satisfactorily. Hardware has some physical limitations and has practically bound on complexity because every hardware design must be released as a physical implementation. The software can be highly complex while conforming to almost any set of needs. There are major differences between the management of hardware and software projects. Traditional control of hardware projects may be counterproductive in a software project.

1.5 TYPES OF SOFTWARE

In today's world we found that there are many different types of software. It can be categorized in three parts, which are custom software, embedded software, and generic software.

The specific need of some particular group of customers can develop the custom software. Although in some cases it has been found that developing custom software can raise the roof of a problem shared by several organizations. Some of the organization can develop their own in-house custom software and others can develop by consulting organizations. Some example of custom software includes websites of the organization, air or road traffic signal system for nation and software for managing some financial specialization of an organization, etc.

The hardware devices can use software called embedded software. Devices like DVD players, washing machines, microwave machines, air conditioning machines, televisions, automobiles, etc., use embedded software. Most of the cases it has been found that the embedded software cannot be upgraded or replaced. In that case, the user can replace the hardware itself.

Commercial off-the-shelf software or generic software is designed for the open market to perform functions that many people need and run on general purpose computers. The success of the generic software is the mercy of the market forces. Example of generic software includes compilers, web browser, computer games, operating system, spreadsheet, word processor, accounting package, etc.

By the characteristic of the software, it may be divided into some of the branches, they are as follows

1. **System Software:** Resource management and operating system services are included in this software.

2. **Business Software:** Existing data restructure can facilitate business management and decision-making capability.

3. **Personal Computer Software:** This category includes computer graphics, multimedia, spreadsheet, word processing, presentation making, entertainment, database management system, personal, and business management financial application.

4. **Web based Software**: Executable web based application can help the organizations world wide.

5. **Real time System:** This software can generate or manage the task within a specified time limit.

6. **Artificial Intelligence Software:** To solve complex problems, sometimes it has been used non-numerical algorithms. This analysis is included in this type of software. Example of this is expert system, pattern recognition, artificial neural networks, game playing, and theory proving etc.

7. **Engineering and Scientific Software:** Orbital dynamics, molecular biology, automotive stress analysis, astronomy, etc., are this type of software.

8. **Embedded Software:** This software can be written in read-only memory of hardware devices. This software can be used for the industrial market and the customer.

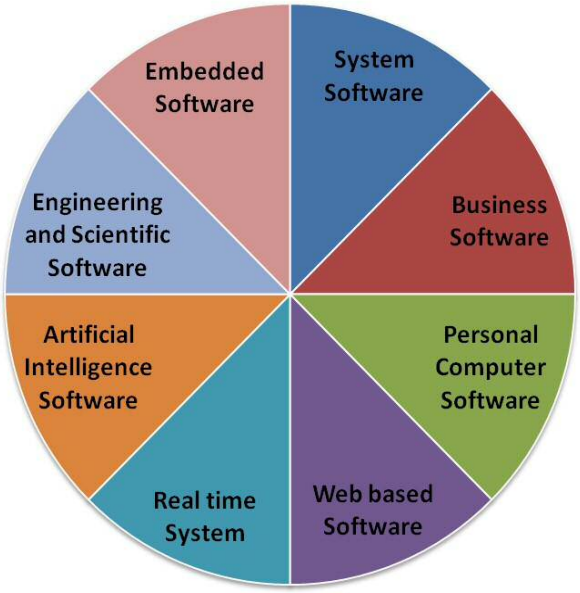

Figure 1.5: Software nature by characteristic

1.6 EVOLVING ROLE OF SOFTWARE

The software has undergone significant changes over a span of 100 years. Dramatic improvement in the computer hardware as well as reflective changes in architecture, enormous enlargement in memory size, storage capacity, and variety of input/output options have contributed to sophisticated and complex computer systems.

Software takes duel role in today's era. It is a product itself as well as a material for delivering a product. Software as a product delivers the computing potential embodied by computer hardware or more broadly computer network are accessible by the local hardware. In case of software for material use to deliver a product acts as the basis for control of the computer operating system, communication information like network and certain control of other programs by some software tools and environment.

1.7 SOFTWARE COMPONENTS

Computer software can be categorized into two basic forms – non-machine-executable components and machine-executable components. Software components are developed through a series of translations that fulfill the customer requirements to machine executable code. In 1977 Gilb defined two principal components of the software which are as follows:

Logic-ware: The logical sequence of active instructions that can control the execution sequence which has been occurred by the hardware is called logic-ware. It is the sequential procedure to process the data.

Data-ware: The physical form of passive information that appears in the hardware by result processing is called *data-ware*. The logic-ware also included in this mechanism.

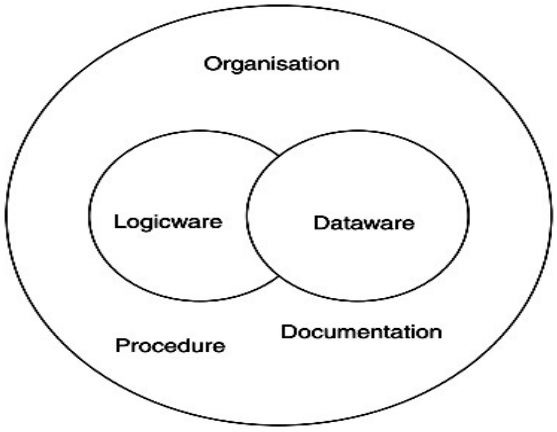

Figure 1.6: Software components

1.8 PROGRAM VERSUS SOFTWARE

Peoples compare the term software with computer program. But these two are interconnected. Software is a collection of various programs that associated with configure data or information and documentation to operate appropriately. In a software system separate program modules, configuration files, hardware integration helps to setup the programs as well as system documentations can describe the structure of the system and user documentation can deliberate the software product viz. application software or website portal information.

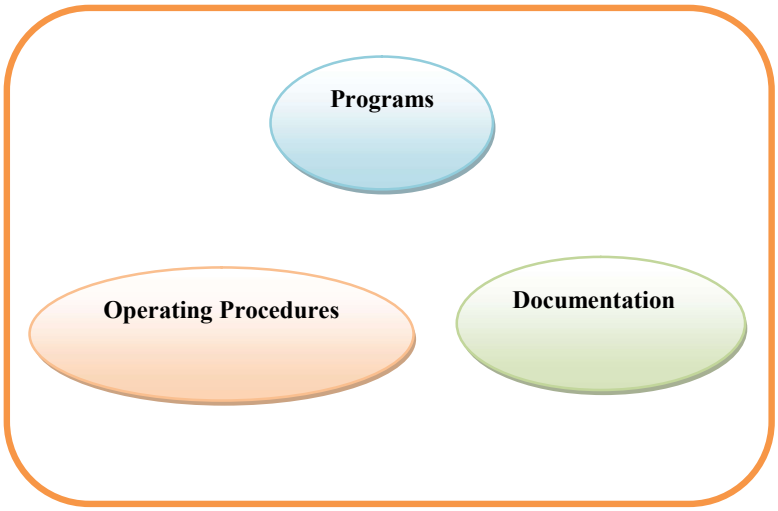

Figure 1.7: Software processes

Software consists of the following:

- Requirement analysis
- Designing of software model
- Design documentation
- Source code
- Executable code
- User manual
- Installation guide

Table 1.1: Comparison: Software vs. Program

Criteria	Software	Program
Mode of Implementation	It is developed by a group of engineers or a team.	It is developed by individuals for their personal interest.
Functionality	More	Limited
Size	Large	Small
Users	Large numbers	Single
Documentation	Good documentation support.	Lack of proper documentation.
Development	Systematic	Adhoc
User Interface	Good	Lack

1.9 SOFTWARE MYTHS & SOFTWARE CRISIS

Software development community has numbers of myths. Some of them affect the approach in which software development take place. Some discussions are given below regarding the myths applicable to standard software.

- *Software is easy to change*: The source code files are easy to change but it can not say that the software is easy to change. Source code is easy to alter and it is closely ambiguous. Making change without introducing error is enormously complicated for the organization having poor process maturity. The complete system requires re-verifying after every change. Without proper care of changing source code will be arise an extreme monotonous and expensive process.

- *Computers provide greater reliability than the devices they replace*: Traditionally the software does not fail. A given module of code can be executing number of times; it has no limit before it wears out. In any occurrence the plain expression of this myth is that our general ledgers are not absolutely correct, even through they have been computerized. Human errors were a fact of life in the olden days of manual accounting system. But now it has software errors as well.

- *Testing or providing software correct can remove all the errors*: The presence of error can be shown by the testing. Absence of error can not shown by it. The aim is to design effective test cases in order to find maximum possible errors. More testing can make more confident about the design.

- *Reusing software increases safety*: This myth is predominantly disturbing since of the false sense of security that code reuse can create. Reuse of code is very powerful tool because it can yield dramatic improvement in development efficiency, but it

still requires analysis to conclude its appropriateness and testing to conclude if it works.

- *Software can work right the first time*: An aeronautical engineer quote a price if he asked to build a fighter craft. If it is demanded he to be put in production without building the prototype then he refuse the job. Yet, software engineers are often asking to do preciously this sort of work and they often accept the job.

- *Software can be designed thoroughly enough to avoid most integration problems*: "Too bad there is no compiler for specifications" said by the software designer in the old age. This point is the fundamental difficulty of software design which detailed specification. There is no tools are there to check the consistency, so they always have inconsistency. Thereafter, special care is required to understand the specifications and if ambiguity found that should be resolved before the designing.

- *Better software means software with more features*: this is of course, almost opposite of the truth. The best most enduring programs are those which do one thing well.

- *Addition off mood software engineers will make up the delay*: in most of the cases this is not true. It may further delay the project by the process of adding more software engineers during the project. This may be true for any type of civil engineering work but in this case it does not serve any purpose here.

- *Aim is to develop working programs*: Developing working programs to good quality and maintainable programs is the today's aim. A very critical and crucial area of software engineering community is maintaining the software. This list is endless. These myths, poor software quality, increasing the cost and delay in the delivery of the software have been the driving forces behind the emergence of software engineering as a discipline. Some of the contributing factors are as follows: change in ratio of hardware to software increasing importance of maintenance, advances software techniques, increased demand for software and demand for more complex as well as larger software systems.

A crisis, it is the problems associated with software development observed by many industries. The word '*crisis*' is defined as '*a turning point in the course of anything, crucial time, stage or event*'. In terms of overall software quality as well as the speed with Computer Based systems and products are developed there have bin no turning point, no receive time. The disciplines associated with the software are slow evolutionary change and punctuated by exclusive technological changes.

Software engineering was encouraged by so called software crisis from 1960s to 1980s. With this period of time the software industry unsuccessfully attempted to build larger

and largest software system by scaling up existing development techniques. Software runaways are a number of large size projects failed because of the following reasons:

- Quality of the software is very poor
- Development teams exceeding the budget
- Delivery of software is late
- User requirements completely not supported by the software
- Maintenance is difficult
- Software is unreliable
- Cannot be transferred to machine easily and quickly
- Not always easy to use.

Statistics shows that only 2% of the projects were used as they were delivered, 3% of the projects used after modifications, 47% of the projects were never used only delivered, 19% of the projects rejected or reworked and 29% was not delivered. The problems increased due to hues dependence of business on software and lack of systematic approach to build the software. Researchers and developers realized that development of software was not an easy and straight forward task; instead it required lot of engineering principal.

In the other words, the software crisis is characterized by an inability to develop software on time, on budget and within requirements. There are many factors related to the making of present software crisis. Those factors are lack of adequate training in software engineering, major problem in sizes, increasing skill shortage and low productivity improvements. Developers and researchers found that there was a lot of scope for building quality software.

1.10 WHAT IS SOFTWARE ENGINEERING?

The phrase software engineering is called the current software crisis that should be solved by existing engineering practices for development of software. The crisis was characterized by consistent development of low quality software that increases the cost limit and development deadlines.

Software engineering is concerned with all aspects of software production. Software engineering is a engineering discipline. Depending on the problem of the organization, software engineers should adopt a systematic and organized approach to their work; also use appropriate tools and techniques for to solve the problem.

Therefore software engineering is concerned with all aspect of software production from the early stage of system specifications to maintaining system after it has started to use. In this connection there are two points should be discussed:

Engineering discipline: Engineers apply theories, methods and tools where these are appropriate, but they use them selectively and try to discover problem solutions when there are no applicable methods and theories. Engineers look for solutions within various constrains because their recognizers that engineers must work to organizational and financial constraints.

All aspects of software production: Software engineering is not a technical process of software development. Software production can be possible through some activities like software project management and with the development of tools, methods and theories.

The engineering problems, opportunities and needs associated with the development and utilization of computer software are buy software engineering.

The integration of software into computer systems, design of computers as well as the applications of software systems are controlled by software engineering. Rapid growing of software industry and increasing importance of our economy as well as standard of living need this discipline. Software engineering associated with it many industries like power generation, telecommunication, biomedical, financial sector and industrial product companies. New challenges as well as new opportunities have includes the development of powerful software. The area of software engineering applications is growing and special effects software for the movies industries with the help of software controlling devices like digital camera, robots. The software industries have significantly impacted the global economy due to the rapid growth of the software industries within a few years. Many companies are involved in the software engineering industry across the region, province and country.

Software engineering it is a layered Technology shown in below figure. Software engineering is like engineering approach that must fulfill organizational commitment to quality. Similar philosopher promotes and quality management is the continuous process improvement. This culture ultimately increases the mature approaches of software engineering.

Figure 1.8: Software engineering - layered technology

The bottom portion support software engineering is a quality focus. Next is process layer, it is the foundation for software engineering. Software engineering process is the glue that holds the technology layer together. Also enables the timely and rational development of software. Process defines the set off key process area that must be established for effective delivery of software engineering technology. Software engineering methods can provide the technical view for building software. Method is the broad area of tasks that includes requirement analysis, design of forms, program construction, testing of program and support.

Software engineering tools can provide automatic or semi-automated support for the method and for the support. When tools are integrated and another tool can use information created by one another tool, it system for the support of software development, called computer-aided software engineering (CASE). CASE combine hardware with software and software engineering database to create the engineering environment for hardware.

1.11 DEFINITIONS OF SOFTWARE ENGINEERING

In 1986, Fritz Bauer define software engineering as establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently real machines.

After that in 1990, Stephen Schach defines the software engineering as indiscipline whose aim is the production of quality software, software that delivered on time, within budget and that satisfies its requirements.

According to IEEE standard software engineering is defined as the application of a symmetric, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e. the application of engineering to software.

Software engineering is the process of solving customer's problems by the systematic development and evolution of large, high quality software systems within cost, time and other constraints.

Software engineering is the technological and managerial discipline concerned with systematic production and modified on time and in cost estimates.

Software engineering is one of the engineering disciplines, which is concerned with all aspects of software production from the early stage of system specification to maintenance the software system before it go for user stage.

1.12 SIMILARITIES AND DIFFERENCES BETWEEN CONVENTIONAL AND SOFTWARE ENGINEERING

In 1979 Jensen and Tonies declare that software engineering is related to the design of software or data processing and it is belongs to problem solving domain. They also said that drawing analogy from the methods that are generally used in engineering. According to them just like scientific method use in scientific research, the steps of engineering process problem formulation, analysis of problem, alternative search, decision, specifications and implementation.

In 1992, Pressman declared that software engineering is an outgrowth of hardware and system engineering. He said that there are three elements like methods, tools and procedures. According to Pressman, methods provide the technical part for building software; tools provide the automated or semi-automated support for methods and procedures stands for the sequence of applying the methods, the deliverables, the controls and the milestones.

A comparison of the two approaches of the software engineering shows some remarkable differences which are described below

- A move from an abstract design to a concrete product is the conventional engineering. A move from design to coding is called abstract in the software engineering.
- Software engineering problems can be anything like word processing to real time system and from games to robotics. But in other discipline of engineering, it is much wider in scope and with greater challenge.
- Computer science is concerned with the theory and methods that underlie computer and software systems, whereas software engineering is concerned with the practical problems producing software.
- Some knowledge of computer science is very much essential for software engineers just like the knowledge of Physics is essential for engineer in the electrical discipline. Software engineer's mast often uses ad hoc approaches for developing the software.

1.13 GENERIC VIEWS OF SOFTWARE ENGINEERING

The general view can be categorized into three parts that are definition phase, development phase, and supports phase.

The definition means what. In this phase the software engineers try to know what information is to be processed, what function and performance are desired, what design constraints can exist, what validation criteria are required to define a successful

system. The key requirements of the system and the software are identified. There are three major tasks can occur in some forms which are as follows:

- System or information engineering
- Software project planning
- Requirements analysis

The development phase stands for how. During the development stage, the software engineers try to define how data to be structured, how technical particulars are to be implemented, how the function is to be executed within a software construction, how interfaces are to be illustrated, how the design will be interpreted into the programming language and how testing will be carried out. There are three specific architectural tasks should be performed.

- Design of software
- Generation of code
- Testing of software

The support phase stands for change. Change occurs due to error correction as well as enhancement brought about by changing customer requirements. This phase is also can recap the steps from definition and development in the context of existing software. There are four types of changes are encountered during this phase which is listed below.

- Correction
- Adaptation
- Enhancement
- Prevention

1.14 COST OF SOFTWARE ENGINEERING

What are the costs of software engineering? This question is very interesting but the answer to this question is not simple. The distribution of fast depends upon the different activities of the software process model as well as the type of software that is being developed. For example, a web-based system needs validation testing but the real-time system required more extensive validation as well as testing.

In Figure 1.9, it has been assumed that the total cost of developing a complex software system is 100 units. Now the figure shows how these units are spent on different process activities.

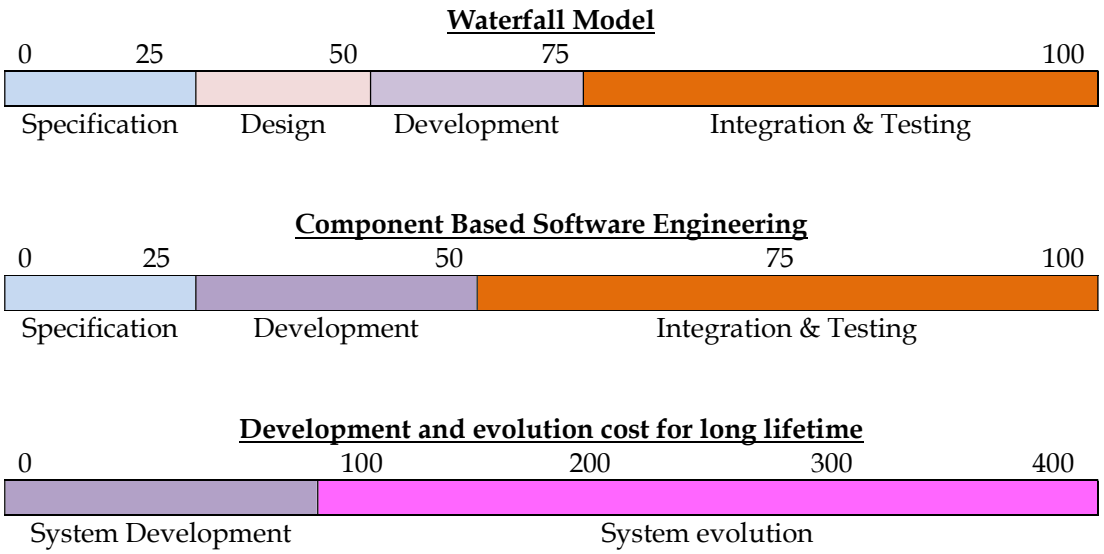

Figure 1.9: Software cost estimating model

For generic software products, the evolution cost is hard to estimate. When a specific version of the software has been released, a team of developers starts their work for the next release. The software is likely to be presented as a new product for marketing purposes rather than a modified version of the previous one which has already bought by the end user. Now in this case the evolution costs are not being calculated separately as they are customized software, but they can be calculated as the development cost for the next version.

1.15 SOFTWARE PROCESS MODEL

A set of activities and associated results that produce software is called the *software process*. There are four activities that are common to all software process models which are described below:

1. *Software Specification:* When the customer or the user identifies the problem definition and engineers define the software to be produced on its operations is called software specifications.

2. *Software Development*: After completion of the specification stage the software, it is needed to start the design and programming for the implementations.

3. *Software Validation*: In this section, it has been checked before delivery to the customer that the software has been developed and run properly as per the requirements of the user or not.

4. *Software Evolution*: As per the changing customer requirements as well as the market necessities the software is need to be modified. This modification or altering process is occurred in the software evolution section.

A software process framework has been established for a small number of activities applicable to all types of software projects, regardless of their complexity as well as the size of the software. The common process frameworks are as follows:

- Communication with customers and knowledge gathering.
- Planning of development to complete the project.
- Modeling for developer and customer to understand the requirements.
- Construction of code and testing of the design.
- Development and delivered the software for customer evolution and feedback.

Umbrella activities are those activities that are independent of any one framework activity and this will occur throughout the process. Software engineering umbrella activities are as discussed below:

- Software project tracking and control to maintain the schedule of development.
- Risk management for project outcomes or quality.
- Software quality assurance is required to maintain the quality of the software.
- Formal technical reviews to uncover and remove errors before broadcasting to the next activity.
- Software configuration management can manage all the activities of the user of the software.
- Measurement to assist the software development team to deliver the software as per the customer's need.
- Reusability management is for reusing the product.
- Work product preparation and production are used for documentation of created models, logs, forms, lists, data bases, etc.

Following steps are the software process characteristics:

- Understandability
- Visibility
- Robustness
- Accessibility
- Reliability
- Maintainability

- Rapidity
- Supportability

1.16 ETHICS OF SOFTWARE ENGINEERING

A sense of professional ethics is very important for a software engineer. People should develop the software after realizing the ethical issue that can arise. ACM or IEEE standard codes of ethics are discussed herewith.

- Consistently act as per the Public Interest.
- Act based on the interest of the client or employer, as this is consistent with the Public Interest.
- Develop as well as maintain the product to the highest standard possible.
- For making professional judgments maintain integrity and independence.
- Promote the ethical approach for the management.
- For consistent public interest advance the integrity and reputation.
- Be fair and supportive to the developers.
- Required to participate in lifelong learning.

1.17 CURRENT TRENDS AND QUALITY OF SOFTWARE ENGINEERING

Software engineering is a discipline what is young and is still developing discipline. The current trends can be categories in some parts that are described below:

- **Aspects:** This can help the software developer to deals with the quality attributes by providing tools to add or remove code functions from many areas of the entire source code. It deals with objects and functions that should behave in particular circumstances.

- **Agile:** This software development can guide the project that repeatedly changes expectations and competitive markets. Document driving processes are fading in importance.

- **Experimental:** Experimental software engineering is the branch of software engineering for the experiment of the software, data collection, maintaining rules and laws and theories of the data.

- **Model driven:** It is the process to develop textual and graphical model. Development tools are available which are used for model transformation as well as code generation for making well organized code fragments.

- **Software product lines:** It is a systematic way to produce families of software system. This method is Systematic, extensive, formal code reuse to industrialize the software development process.

Quality of the software is more important aspect in the context of software engineering, because each and everyone want high quality software. But the quality is a term which cannot be measured like beauty. Below figure shows that quality means each of the stakeholders. All of them consider software to be good quality if the outcome of the project as well as maintenance helps them to meet their objectives.

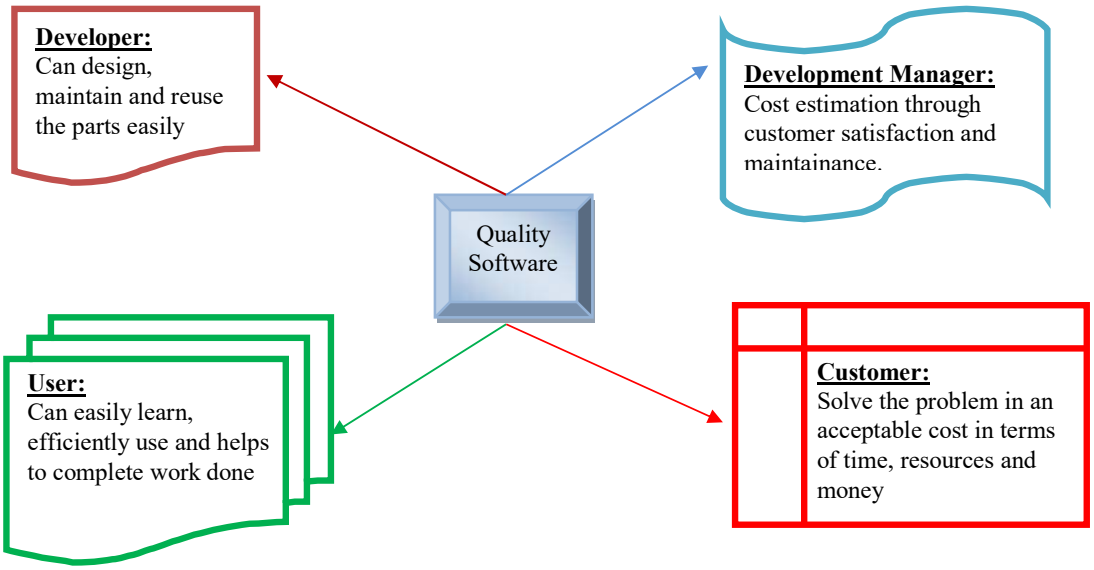

Figure 1.10: Software quality by means of the stakeholders

SUMMARY

In this chapter, we have learned that

- Different types of software are custom software, generic software and embedded software.
- Software has the key elements in the evolution of Computer Based system as well as products.
- Software is composed of programs, data or information and documents.

- Software consists of source code, executable code, user manual, requirement analysis and design document, installation guide.
- Software development is heavily dependent on four factors like people, product, process and project.
- In this domain, cost, schedule and quality are the basic forces. Therefore, tools and methods have been used for solving the problem in this domain mast ensure high quality software.
- Software development requires good managers, who cannot ensure the success of the project but can increase the probability of success.
- Software quality has many attributes like functionality, reliability, usability, efficiency, maintainability and portability.
- Software can be flexible and program can be developed to do almost everything.

EXERCISE

1. What is software?
2. What are the application areas of software?
3. Describe the role of software.
4. Discuss the characteristics of the software.
5. Describe the software crisis.
6. What are the different software components?
7. Why it is difficult to build quality software?
8. Discuss the software myths.
9. What are the essential qualities of software? Explain it.
10. What is software engineering?
11. Suppose the total cost of developing a complex software system is 100 cost units. How these are spent on (*a*) Waterfall model, (*b*) Development evolution costs for long lifetime.
12. Write the difference in between conventional engineering and software engineering.

❑ ❑ ❑