Subject Code	**15BCA204**
Subject Name	**SOFTWARE ENGINEERING AND TESTING**
Short Name	**SET**
Total Lectures	**88**
Total Credits	**4**

Prerequisites :
- Basic knowledge in System development life cycle should be known.
- Knowledge of basic Computer software and related terms are also necessary.
- Programming constructs along with object oriented concepts must be known.

Objectives:
- To provide an insight into the process of software development.
- To understand and practice the various fields such as analysis, design, development, testing of software engineering.
- To develop skill to construct software of high quality with high reliability.
- To apply metrics and testing technique to evaluate the software.

Units	**Contents**	**Total Lectures**
I	**Software**: Definition, characteristic, myths. **Software engineering**: Definition, layer, management, and phases in software engineering software process, project, product, process component and frame work. **Software life cycle model**: Waterfall, prototyping, spiral incremental, RAD, V model.	18
II	**Software requirement**: Introduction, Types, requirement analysis: Structured, object oriented modeling, other approaches. **SRS**: Characteristic. **Basic of cost estimation**: Estimation of resources, software product cost factor COCOMO model of cost estimation.	18
III	**Software design**: Basic, data design architectural design, component level design, user interface design, object oriented design, software design notation: flowchart, DFD, structure chart. **Software coding**: Feature, programming practices: top down, bottom up, structured, information hiding.	18
IV	**Testing fundamentals**: Error, fault and failure, test cases and test criteria, **Software testing**: basic, strategies, v mode. **Level of software testing**: unit, integration, system acceptance. **Testing Technique**: white box, black box and gray box testing and their comparison.	17
V	**Software quality**: Concept, SQA group. **Quality management**: process and product quality, quality assurance and standard, quality planning & quality control. **Software Maintenance**: Basic, type, software maintenance life cycle.	17
	Text Books : 1. Rohit khurana," Software Engineering Principles and practice", Second edition , Vikas publishing house Pvt. Ltd, 2010 2. Sommerville Pearson, "Software Engineering", Eight Edition, Pearson Education, 2007 3. Pankaj Jalote, "An integrated approach to Software Engineering", Third Edition, Narosa Publishing House, 2005	
	References : 5. Roger S. Pressman ,"Software Engineering : A Practitioner Approach", Seventh edition, McGrawHill, 2010 6. Richard Fairley ,"Software Engineering Concept", Tata McGrawHill Edition 2008 7. Hans van Vliet, "Software Engineering: Principles and Practice", 3rd edition, John Wiley & Sons, 2008.	

Course Outcomes :
1. Achieve basic knowledge of software engineering principles, including models and their

applications in the different phases of software development.
2. Ability to analyze a problem, understand its requirements and use the techniques and tools necessary for software engineering practice.
3. Ability to do project management and its cost estimation.
4. Ability to do software testing using appropriate technique.
5. Ability to manage software quality and do software maintenance.

VIKAS

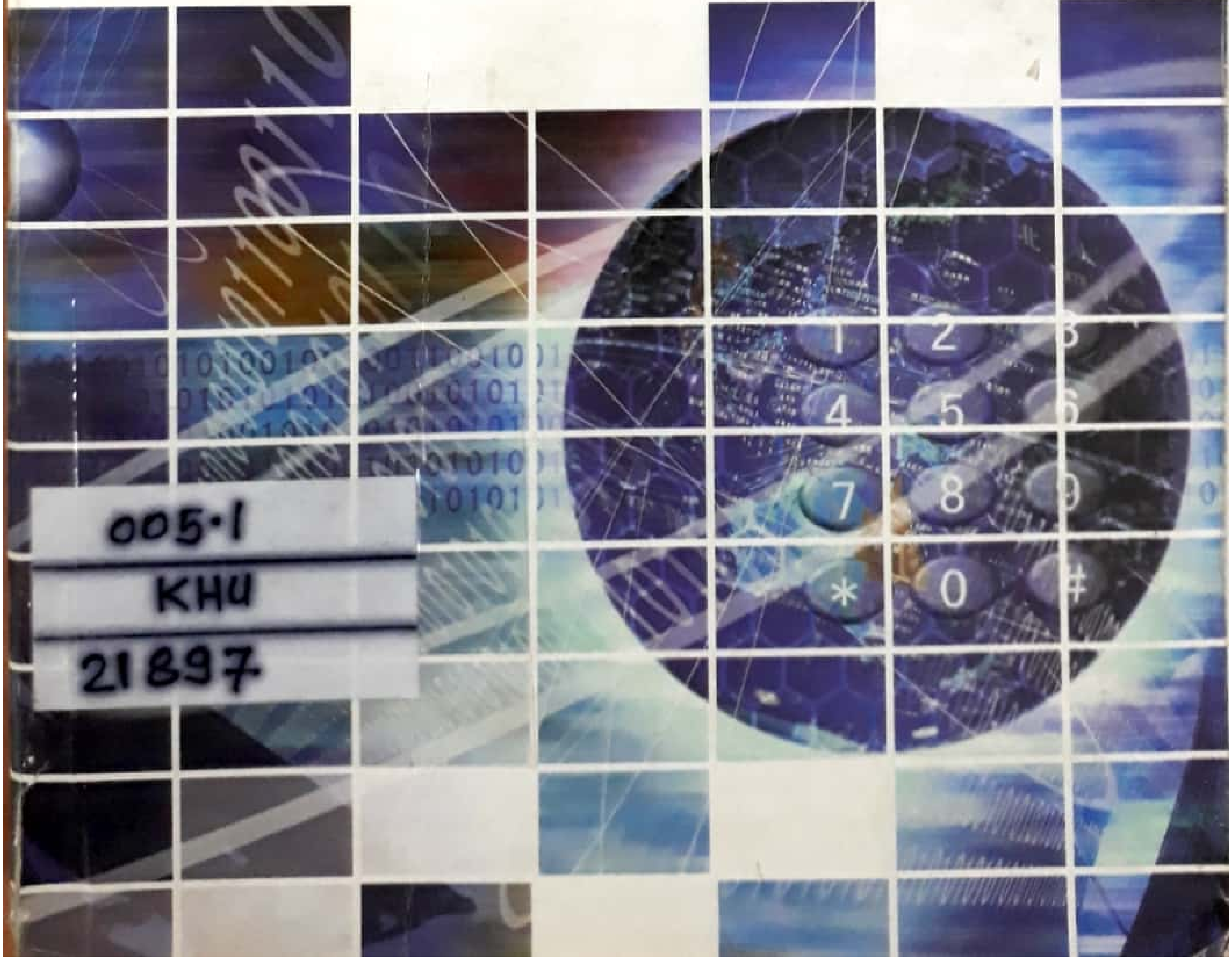
Software Engineering

Principles and Practices

Second Edition

Rohit Khurana

ITL ESL



In earlier times, software was simple in nature and hence, software development was a simple activity. However, as technology improved, software became more complex and software projects grew larger. Software development now necessitated the presence of a team, which could prepare detailed plans and designs, carry out testing, develop intuitive user interfaces, and integrate all these activities into a system. This new approach led to the emergence of a discipline known as **software engineering**.

Software engineering provides methods to handle complexities in a software system and enables the development of reliable software systems, which maximize productivity. In addition to the technical aspects of the software development, it also covers management activities which include guiding the team, budgeting, preparing schedules, etc. The notion of software engineering was first proposed in 1968. Since then, software engineering has evolved as a full-fledged engineering discipline, which is accepted as a field involving in-depth study and research. Software engineering methods and tools have been successfully implemented in various applications spread across different walks of life.

1.1 SOFTWARE

Software is defined as a collection of programs, documentation and operating procedures. The **Institute of Electrical and Electronic Engineers (IEEE)** defines software as a '*collection of computer programs, procedures, rules and associated documentation and data.*' It possesses no mass, no volume, and no colour, which makes it a non-degradable entity over a long period. Software does not wear out or get tired.

Software controls, integrates, and manages the hardware components of a computer system. It also instructs the computer what needs to be done to perform a specific task and how it is to be done. For example, software instructs the hardware how to print a document, take input from the user, and display the output.

Computer works only in response to instructions provided externally. Usually, the instructions to perform some intended tasks are organized into a program using a programming language like C, C++, Java, etc., and submitted to computer. Computer interprets and executes these instructions and provides response to the user accordingly. A set of programs intended to provide users with a set of interrelated functionalities is known as a **software package**. For example, an accounting software package such as Tally provides users the functionality to perform accounting-related activities.

1.1.1 Software Characteristics

Different individuals judge software on different basis. This is because they are involved with the software in different ways. For example, users want the software to perform according to their requirements. Similarly, developers involved in designing, coding, and maintenance of the software evaluate the software by looking at its internal characteristics, before delivering it to the user. Software characteristics are classified into six major components, which are shown in Figure 1.1.

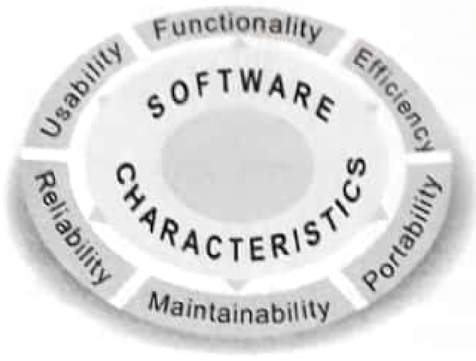


Figure 1.1 Software Characteristics

- **Functionality:** Refers to the degree of performance of the software against its intended purpose.
- **Reliability:** Refers to the ability of the software to provide desired functionality under the given conditions.
- **Usability:** Refers to the extent to which the software can be used with ease.
- **Efficiency:** Refers to the ability of the software to use system resources in the most effective and efficient manner.
- **Maintainability:** Refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.
- **Portability:** Refers to the ease with which software developers can transfer software from one platform to another, without (or with minimum) changes. In simple terms, it refers to the ability of software to function properly on different hardware and software platforms without making any changes in it.

In addition to the above mentioned characteristics, robustness and integrity are also important. **Robustness** refers to the degree to which the software can keep on functioning in spite of being provided with invalid data while **integrity** refers to the degree to which unauthorized access to the software or data can be prevented.

1.1.2 Classification of Software

Software can be applied in countless fields such as business, education, social sector, and other fields. It is designed to suit some specific goals such as data processing, information sharing, communication, and so on. It is classified according to the range of potential of applications. These classifications are listed below.

- **System software:** This class of software manages and controls the internal operations of a computer system. It is a group of programs, which is responsible for using computer resources efficiently and effectively. For example, an operating system is a system software, which controls the hardware, manages memory and multitasking functions, and acts as an interface between application programs and the computer.

- **Real-time software:** This class of software observes, analyzes, and controls real world events as they occur. Generally, a real-time system guarantees a response to an external event within a specified period of time. An example of real-time software is the software used for weather forecasting that collects and processes parameters like temperature and humidity from the external environment to forecast the weather. Most of the defence organizations all 2) over the world use real-time software to control their military hardware.

- **Business software:** This class of software is widely used in areas where management and control of financial activities is of utmost importance. The fundamental component of a business system comprises payroll, inventory, and accounting software that permit the user to access relevant data from the database. These activities are usually performed with the help of specialized 3) business software that facilitates efficient framework in business operations and in management decisions.

- **Engineering and scientific software:** This class of software has emerged as a powerful tool in the research and development of next generation technology. Applications such as the study of celestial bodies, under-surface activities, and programming of an orbital path for space shuttles are heavily dependent 4) on engineering and scientific software. This software is designed to perform precise calculations on complex numerical data that are obtained during real-time environment.

- **Artificial intelligence (AI) software:** This class of software is used where the problem-solving technique is non-algorithmic in nature. The solutions of such problems are generally non-agreeable to computation or straightforward 5) analysis. Instead, these problems require specific problem-solving strategies that include expert system, pattern recognition, and game-playing techniques. In addition, they involve different kinds of search techniques which include the use of heuristics. The role of artificial intelligence software is to add certain degrees of intelligence to the mechanical hardware in order to get the desired work done in an agile manner.

- **Web-based software:** This class of software acts as an interface between the user and the Internet. Data on the Internet is in the form of text, audio, or video format, linked with hyperlinks. Web browser is a software that retrieves 6) web pages from the Internet. The software incorporates executable instructions written in special scripting languages such as CGI or ASP. Apart from providing navigation on the Web, this software also supports additional features that are useful while surfing the Internet.

- **Personal computer (PC) software:** This class of software is used for both official and personal use. The personal computer software market has grown 7) over in the last two decades from normal text editor to word processor and from simple paintbrush to advanced image-editing software. This software is used predominantly in almost every field, whether it is database management system, financial accounting package, or multimedia-based software. It has emerged as a versatile tool for routine applications.

1.1.3 Software Myths

The development of software requires dedication and understanding on the developers' part. Many software problems arise due to myths that are formed during the initial stages of software development. ~~Unlike ancient folklore that often provides valuable lessons,~~ software myths propagate false beliefs and confusion in the minds of management, users and developers.

Managers, who own software development responsibility, are often under strain and pressure to maintain a software budget, time constraints, improved quality, and many other considerations. Common management myths are listed in Table 1.1.

Table 1.1 Management Myths

Myths	***Realities***
• The members of an organization can acquire all the information they require from a manual, which contains standards, procedures, and principles.	• Standards are often incomplete, inadaptable, and outdated. • Developers are often unaware of all the established standards. • Developers rarely follow all the known standards because not all the standards tend to decrease the delivery time of software while maintaining its quality.
• If the project is behind schedule, increasing the number of programmers can reduce the time gap.	• Adding more manpower to the project, which is already behind schedule, further delays the project. • New workers take longer to learn about the project as compared to those already working on the project.
• If the project is outsourced to a third party, the management can relax and let the other firm develop software for them.	• Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The organization invariably suffers when it outsources the software project.

In most cases, users tend to believe myths about the software because software managers and developers do not try to correct the false beliefs. These myths lead to false expectations and ultimately develop dissatisfaction among the users. Common user myths are listed in Table 1.2.

Table 1.2 User Myths

Myths	***Realities***
• Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages.	• Starting development with incomplete and ambiguous requirements often lead to software failure. Instead, a complete and formal description of requirements is essential before starting development. • Adding requirements at a later stage often requires repeating the entire development process.
• Software is flexible; hence software requirement changes can be added during any phase of the development process.	• Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require redesigning and extra resources.

In the early days of software development, programming was viewed as an art, but now software development has gradually become an engineering discipline. However, developers still believe in some myths. Some of the common developer myths are listed in Table 1.3.

Table 1.3 Developer Myths

Myths	***Realities***
• Software development is considered complete when the code is delivered.	• 50% to 70% of all the effort are expended after the software is delivered to the user.
• The success of a software project depends on the quality of the product produced.	• The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also play a crucial role.
• Software engineering requires unnecessary documentation, which slows down the project.	• Software engineering is about creating quality at every level of the software project. Proper documentation enhances quality which results in reducing the amount of rework.
• The only product that is delivered after the completion of a project is the working program(s).	• The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software.
• Software quality can be assessed only after the program is executed.	• The quality of software can be measured during any phase of development process by applying some quality assurance mechanism. One such mechanism is formal technical review that can be effectively used during each phase of development to uncover certain errors.