11.6.3(B) Bottom-up Testing 11-17
11.6.3(C) Big-Bang Approach 11-18
11.6.3(D) Sandwich Testing 11-19
11.6.4 Level 4 : System Testing 11-19
11.6.5 Level 5 : Acceptance Testing 11-22
11.6.6 Integration Testing vs. System Testing 11-23
11.7 Static Verification 11-24
11.7.1 Code Review 11-24
11.7.2 Desk Checking 11-25
11.7.3 Code Walk-Through 11-25
11.7.4 Code Inspection 11-26
✓ **Syllabus Topic :** White-box Testing / Structural Testing 11-28
11.8 White-box Testing (Structural Testing) 11-28
11.8.1 Code Coverage Testing 11-30
11.8.2 Code Complexity Testing 11-32
✓ **Syllabus Topic :** Functional / Black Box Testing 11-40
11.9 Functional (Black Box) Testing 11-40
11.9.1 Requirements based Testing 11-41
11.9.2 Positive and Negative Testing 11-41
11.9.3 Boundary Value Analysis 11-42
11.9.4 Equivalence Class Partition 11-43
11.9.5 Domain Testing 11-46
11.9.6 Cause Effect Graphing 11-46
11.10 Black Box vs. White Box 11-49
✓ **Syllabus Topic :** Test Plan 11-49
11.11 Test Plan 11-49
11.11.1 Test Plan Template 11-50
11.11.2 Test Plan Examples 11-54
✓ **Syllabus Topic :** Test-Case Design 11-60
11.12 Test Cases 11-60
11.12.1 Login Test Case 11-61
11.12.2 Email Address Test Case 11-64
11.12.3 Test Case for Calculator 11-65
11.12.4 Test Case for ATM 11-65
• **Model Question Papers .... M-1 to M-5**
• **Appendix A : Solved University Question Paper of April 2018 .... A-1 to A-4**

□□□

UNIT I

CHAPTER 1 Introduction to Software Engineering

Syllabus :
The Nature of Software, Software Engineering.

1.1 Introduction

– Computers are becoming an unavoidable necessity or say the part and parcel of our life. We do most of our day's work using computers and digital devices – to book railway tickets or movie tickets, search for some books on amazon, bank transactions, and any task you name it is done using computers. Therefore it is very important to build these computerized systems in an effective way.

– Building such systems requires certain technical as well as communication skills and capabilities to understand and follow a planned and systematic procedure.

1.2 Software

Q. Define software.

– Software is a set of instructions to acquire inputs and to process them to produce the desired output in terms of functions and performance as determined by the user of the software. It is developed to handle an *Input-Process-Output* system to achieve predetermined goals.

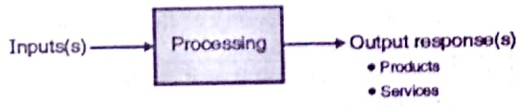

Fig. 1.2.1 : Basic diagram of a software

– **Software encompasses of**
1. Instructions (Computer programs) 2. Documents (Describes programs)
3. Architecture including Data Structures (Enables programs)

– Software includes things such as websites, programs or video games that are coded by programming languages like C or C++.

☞ **Software - a Product and a Vehicle :**

"Software plays a dual role - both as a *product* and a *vehicle* that delivers a product".

☞ **Software is a product :**

1. Delivers computing potential

2. Produces, manages, acquires, modifies, displays, or transmits information (e.g., software is in a cellular phone or in any computer)

Software is a vehicle for delivering a product :

1. Controls other programs (e.g., an operating system)
2. Effects communications (e.g., networking software)
3. Helps build other software (e.g., software tools)

1.2.1 Characteristics of Software

Q. State characteristics of software..

– Software means anything which is not hardware but which is used with hardware, such as windows OS (is system software) in computer (is hardware).

You will get more precise idea about the characteristics of software and how it differs from hardware from the below table.

Table 1.2.1 : Characteristics of software and how it differs from hardware

Sr. No.	Software	Hardware
1.	It is developed or engineered.	It is manufactured.
2.	It doesn't wear out as it is not prone to environmental problems.	It wears out as the time passes due to the affects of dust, vibration, temperature extremes and many such environmental problems.
3.	There are no software spare parts which can be used to replace the software.	When hardware fails, it can be replaced by spare parts.
4.	Software is untouchable. It is the code or instructions that tell a computer/hardware how to operate. It has no substance.	Hardware is a physical device something that you're able to touch and see.
5.	Software is usually generic but it can also be custom built (developed according to the customer specification).	It is manufactured or assembled by using the existing components.
6.	Software is invaluable as it can be installed in any hardware.	Hardware has no value without software in it. If computers (h/w) had no operating system (s/w), then no customer will buy it.
7.	Examples: Microsoft, Windows any operating system that allows you to control your computer. Example 2: Internet browsers, video games, applications like Payroll etc.	Examples: disks, disk drives, display screens, keyboards, printers and chips.

1.2.2 Classes of Software

Software is classified into two types of classes:

- Generic
- Customized

Table 1.2.2 : Difference between generic software and customized software

Sr. No.	Generic Software	Customized Software
1.	Designed for broad customer market.	Designed for specific business purposes.
2.	Is open to market and its specifications are designed by the programmer.	Is an s/w and its specifications are designed according to a particular firm or organization; it is not open for all.
3.	Examples: ERP/CRM, CAD/CAM packages, OS, System Software.	Examples: Legacy systems, Process control systems, Traffic management system and Hospital management system.
4.	Generic software development is done for general purpose audience.	Custom Software Development is done to satisfy a particular need of a particular client.
5.	Requirements and specifications of this software are managed by the developer.	Managed by the customer and influenced by the practices of that industry.
6.	General Purpose software development is tough as compared with Custom made by the design and marketing point of view.	By Buyer's point of view, he prefers to have a Custom Made application developed rather than buying a General Purpose software as he gets the application done that exactly matches his requirements.
7.	In General Purpose application design and development, you need to imagine what an end-user requires. Market Surveys and general customer demand analysis may help in such designs.	In Custom Made application, you have a specific end – user. Understanding his need and analyzing it to get the best out of it helps in such designs.

Syllabus Topic : The Nature of Software

1.3 The Nature of Software

Software is classified into 7 categories as below :

1. System software

It directly interacts with computer hardware. It is generic software.

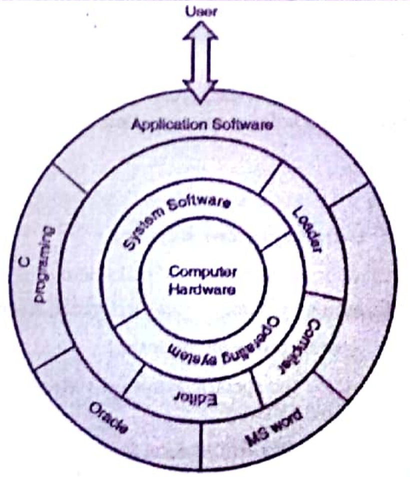

Fig. 1.3.1 : Layers of the computer software

System software is classified into three categories :

(1) Operating system
(2) System support software
(3) System development software.

(1) **Operating system :** Operating system acts as an interface between user and the hardware, and provides different services to users.

Examples : DOS, Windows-XP, LINUX etc.

(2) **System support software :** System support software manages hardware more efficiently.

Examples : Drivers of the IO devices or Antivirus software.

- Drivers of the IO device consist of interfacing programs which can interacts with IO device. Every IO device has its own driver programs.
- Antivirus programs remove viruses from the systems and some time recover systems from major data loss.

(3) **System development software :** System development software supports programming development environment to user.

Examples : Editor, Pre-Processors, Compiler, Interpreter, Loader etc.

- Editor is used to create programs as well as it is used for modification of existing programs.
- Pre-processor works before the use of translators (that means compiler or editors) to replace some segments of code with some other segment. It is also called as an expansion.
- Compilers can translate high level programs into low level programs.
- Interpreters can translate line by line high level programs into low level programs.
- Loaders are the software which can load object codes into the main memory and execute it.

2. Application software

Application software is only designed to solve user problems as per user's requirement.

Application software can generic or customized. Application software is classified into two categories :

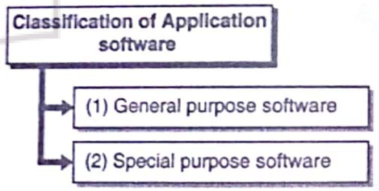
Classification of Application software
(1) General purpose software
(2) Special purpose software

Fig. C1.1 : Classification of Application software

➔ **(1) General purpose software :**

It used for much number of tasks, and provides many features. It is generic software.

Examples : Word Processor, Oracle, Excel etc.

➔ **(2) Special purpose software :**

It designed for specific purposes only. User programs come under special purpose software. It is customized software.

Examples : Pay Roll system for specific company, Tax Calculation Software etc.

Q. Differentiate between system software and application software

Table 1.3.1 : Comparison of system software with application software

Sr. No.	System software	Application software
1.	Primarily concerned with the efficient management of the computer system.	Primarily concerned with the solution of some problems, using the computer as a tool.
2.	System programs are intended to support the operation and use of the computer itself.	The focus is on the application, not on the computing system.
3.	Is usually related to the architecture of the machine on which it is to run. Thus, system programmer must have knowledge of computer architecture.	Is usually related to the detailed knowledge of the Language and environment. Thus, application programmer must have detailed knowledge of high level programming language which is used to develop application program and environment (operating system) on which they are to run.
4.	It is machine dependent.	It is machine independent.
5.	Used to develop new system programs and using Bootstrapping we can make them portable.	Used to develop new application programs and using cross compiler i.e. again system programs, we can make them portable.
6.	Examples of system software are Assembler, linker, loader, compiler etc.	Examples of Application software are Paint, Photoshop, Microsoft access etc.

3. Engineering/Scientific software

- This type of software deals with processing requirements in specific applications. These are specially used for drawing, drafting, modeling, load calculations and analyzing of engineering and statistical data for interpretation and decision making.
- **Examples :** CAD, CAM, CAE software's. Computer-aided design (CAD) software, computer-aided manufacturing (CAM) software and computer-aided engineering (CAE) software is used in mechanical, electrical or electronic design; simulation, drafting, and engineering; and analysis and manufacturing. CAD, CAM, CAE and design software runs on mainframes, general-purpose workstations, and personal computers (PCs).

4. Embedded software

- This type of software is embedded into hardware as a part of larger systems to control its various functions. This type of software is embedded in Read-Only-Memory (ROM) of the systems/products.
- **Examples :** Keypad control software embedded in a microwave oven or washing machine where there is a need to input, identify the status, decide and take action. These are also called as Intelligent Software.

5. Product-line software

- Software product lines refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production.

- A Product Line Software is a set of software products that share common features but are each different in some way e.g. they may be developed for specific customers or markets (e.g. inventory control products), or, for embedded software (word processors, spreadsheets, computer graphics, personal and business financial applications).

6. Web applications

- A web application is accessed via web browser over a network such as the Internet or an intranet. It is coded in browser-supported languages such as HTML, JavaScript, JSP ASP, PHP. .
- The first-generation web applications allowed the business to post information publicly. Thus, this information is seen to anyone with a Web browser and Internet access. The problem with the first-generation web application is that the information is static.
- The second-generation web applications allowed the users to do interactive queries against existing databases from the Web application. **"Web 2.0"** refers to the second generation of web development and web design. It allows communication, information sharing, interoperability, user-centered design and collaboration on www.
- Third-generation Web application is a powerful business tool for organizations in their Electronic Commerce (EC) efforts.

7. Artificial intelligence software

- This software makes use of non numerical algorithms which use data generated in the system to solve complex problems that are not amenable to problem solving procedures and require specific analysis and interpretation of the problem to solve it.
- **Examples:** robotics, expert systems, artificial neural networks and computer games. All these software's can run either in real-time mode or offline mode. This software's can be shared free or charged by usage. Thus, they are also called Shareware and Freeware. Some of these types of software are run on desktop computers, others on mainframe and mini computers, some are exclusively designed for web, network or the internet.

1.3.1 The Evolving Role of Software

- Software evolution is the term used in Software engineering to refer to the process of developing the software initially and then repeatedly updating it for various reasons.
- As the computer age started in 1960's, software had acquired a great importance in information technology. In the initial stage, the scientists and engineers were the only users of the computers and they only wrote the programs for developing the software. That means, user and the programmer were the same person.
- It's genuine that - as the computing technology, data storage process, communication, networking and programming languages advance, software also undergoes a change.
- The software written in 1960's are totally different from what they are now in all its specifications, such as lines of code, program structure and architecture. Today's software are shorter, smarter and more efficient compared to older versions.
- Nowadays, software includes the source code, executables, design specifications, functions and system's user manuals, installation and implementation manuals to understand the software system.
- Today's trend is to develop the software components that are platform independent.

- The advances in software came hand in hand with more advances in computer hardware. In the mid 1970s, the microcomputer was introduced. This in turn led to the now famous Personal Computer or PC and Microsoft Windows.
- The Software Development Life Cycle (SDLC) also started to appear as a 'centralized construction of software' in the mid 1980s.
- Open-source software started to appear in the early 90s in the form of Linux.
- The Internet and World Wide Web hit in the mid 90s.
- Distributed Systems gained a way to design systems.
- Programmers collaborated and wrote the Agile Manifesto that favored more light weight processes to create cheaper and timelier software.
- Now, the era is of Expert Systems, AI, Neural Networks, Parallel computing and Network computers.

The 'early era -one programmer' is replaced by teams of software engineers, each focusing on different parts of the technology required to build the complete software. And yet the questions asked to today's software programmers are same as that asked to the early era lone programmers:

- Takes more time to develop
- Development cost is high
- Defects found even after delivery
- More that estimated time and budget spent on maintaining the software

These concerned questions with the software development have led to the idea of software engineering practice.

Syllabus Topic : Software Engineering

1.4 Software Engineering

Q. Define Software engineering and its Objectives.

Definition

Software Engineering is a systematic, scientific and disciplined approach towards the development, functioning and maintenance of the software.

As described in the above definition, the systematic and scientific approach in software engineering can be defined as :

- Understanding of customer requirements both by customer and developer.
- Use of structured methodology to gather customer requirements and its analysis to arrive at the Software Requirements Specification(SRS)
- Right estimation of resource, efforts and costs.
- Use of developmental and testing methodology to ensure the quality of software.
- Ease of installation, demonstration and implementation of software by the customer and users.

Such an Engineering approach is required to ensure that the software is designed with the correct choice of technology and architecture to :

- Achieve customer satisfaction
- Ensure on-time delivery

- Be developed within the budget cost
- Provide ease of maintenance to meet changing requirements.
- Else it leads to :
- Unreliable and poor quality software development.
- Late delivery to the customer.
- Difficulty in maintenance
- Lot of expenses in development process ultimately expensive software.

Primary goal of software engineering : to provide quality software at low cost on planned delivery date. Software engineering involves various phases such as project planning, systematic analysis and design, testing and maintenance.

☞ Basic Objectives of Software Engineering are :

- Define software development plan
- Manage software development activities
- Design the proposed product
- Code/develop the product
- Test the product modules
- Integrate the modules and test the system as a whole
- Maintain the product

Software Engineering is the part of a larger subject called *Systems Engineering* which considers issues like hardware platform, operating system, and interoperability between platforms, performance, scalability and upgrades to develop the software. While considering these issues, it uses Computer Aided Software Engineering (CASE) tools (e.g. ArgoUML, Case Studio 2, MagicDraw), software quality testing tools, code generators (e.g. XMLSpy, UModel), and report writers. Thus, a good software engineer must know how to use these tools to develop quality software.

☞ Key challenges of Software Engineering :

- **Heterogeneity :** to use the development techniques that can cope with diverse platforms and execution environments.
- **Delivery :** to use development techniques that lead to faster software delivery.
- **Trust :** to use development techniques that can build trust in users' opinion.

1.5 Software Engineering – A Layered Technology

Q. Explain the layered technology of Software Engineering.

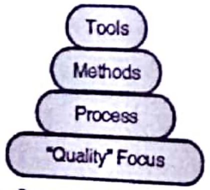

Fig. 1.5.1 : Layers of software engineering

1.5.1 Quality Focus

Software Engineering rests on an organizational commitment to *quality*, which leads to continuous improvements in the Software engineering process. *Focus on quality is always the primary goal of software engineering*. (Quality of Software is explained in section 1.4). In short, we can say that the software is qualitative if it is read and understood easily, if it is modifiable and accommodates new requirements, if it has customer satisfaction and is completed in time within budget.

- The Primary Goal of Any Software Process : High Quality
- Remember : High quality = project timeliness
- Why ? Less rework!

1.5.2 Software Process

Q. What is the need of Software Process ?

When building a product, it's important to go through some predictable steps i.e. called *process* that helps you to create a timely and high quality product. A *process* is the foundation of software engineering. It defines a framework that must be established to ensure effective delivery of Software Engineering technology. The key use of a *process* is :

- Producing models, documents, data, forms and reports.
- Establishing milestones.
- Software Quality Assurance (SQA).
- Software Configuration (change) Management (SCM).

1.5.3 Software Engineering Methods

Method is an ordered way of developing a software. This includes the suggestions for the process to be followed, the notations to be used, the rules governing the system descriptions and the design guidelines

It provides the technical "how to's" for building a software. It includes how to implement the following generic processes :

– Communication	– Requirements Analysis	– Design
– Program Construction	– Testing	– Maintenance

Methods are organized ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

1.5.4 CASE Tools

- They provide automated or semi-automated support for the process and the methods. For example; CASE tool.
- CASE represents Computer Aided Software Engineering tool used in software development process.
- A CASE tool is a computer-aided product specially designed to support the software engineering activities within a software development process.

Examples of Case tools are :

- Code generation CASE tools - Visual Studio .NET
- Code analysis CASE tools -Borland Audits
- CASE tools used for development of data models -UML editors
- Cleaning up code CASE tools -Refactoring tools
- Bug tracker CASE tool
- Version control CASE tool -CVS, etc.

Benefits

- Improves the productivity
- Generates small parts of code automatically
- Improves software quality
- Can be integrated with other tools say, with code editor to work with coding.

Review Questions

Q. 1 Define software and state its characteristics.

Q. 2 Differentiate between system software and application software

Q. 3 Define Software engineering and its Objectives.

Q. 4 Explain the layered technology of Software Engineering.

Q. 5 What is the need of Software Process ?

Q. 6 Identify the following into types of software.

Operating System, Compiler, Testing Software, Sales Analysis, 2D Design Software, Statistical Analysis, Spread Sheet Application, Graphical Analysis and Design, Expert Systems, Automatic Operations Software on ROM in Ovens/Washing Machines/Gas Stations.

❑❑❑

UNIT I

CHAPTER 2

Software Process

Syllabus

The Software Process, Generic Process Model, The Waterfall Model, Incremental Process Models, Evolutionary Process Models, Concurrent Models, Component-Based Development, The Unified Process Phases, Agile Development- Agility, Agile Process, Extreme Programming.

Syllabus Topic : The Software Process

2.1 Software Process

Q. Explain in brief the various types of Software Process.

A software process identifies a set of activities that are applicable to the development of any software project, regardless of their size or complexity.

A software process is a collection of work activities, actions and tasks that are to be performed when some software project is to be developed.

Software process can be categorized into :

1. Generic Process model – represents a framework activity populated by a set of software engineering activities. It includes :
   - Framework activities
   - Umbrella activities
2. Personal and Team Process models – This model helps in creating a software that best fits either the personal needs of the user or that meets the broader needs of a team. It includes :
   - Personal Software Process (PSP)
   - Team Software Process (TSP)
3. Prescriptive Process models – provides an ordered structure and an effective roadmap to software engineering work. It includes :
   - Waterfall model
   - Incremental model
   - RAD model
   - Evolutionary Process models
   - Prototyping
   - Spiral model
   - Concurrent Development model

Prescriptive models define a discrete set of activities and actions to accomplish all tasks of the software with milestones, which is used to develop the software. These Process models may not be perfect but they give very good guidance in software development process.